

# 3 Key Areas in Security

- Authentication
- Authorization
- Exception Handling

# User Database

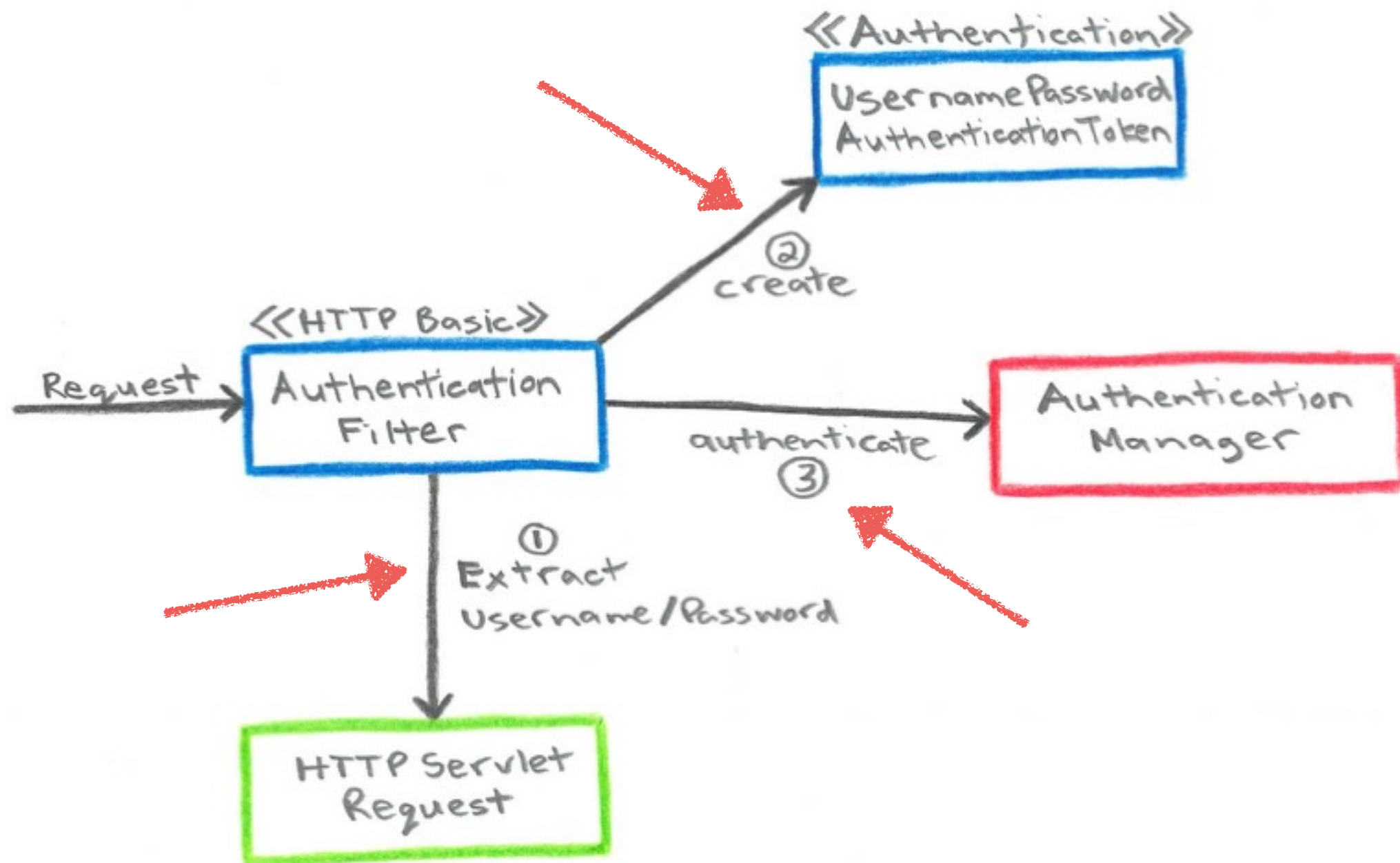
Username	Password	Authorities
<u>joe@example.com</u>	password	ROLE_USER
<u>rob@example.com</u>	password	ROLE_USER
<u>admin@example.com</u>	password	ROLE_USER, ROLE_ADMIN

**Authentication**

Authorization

Exception Handling

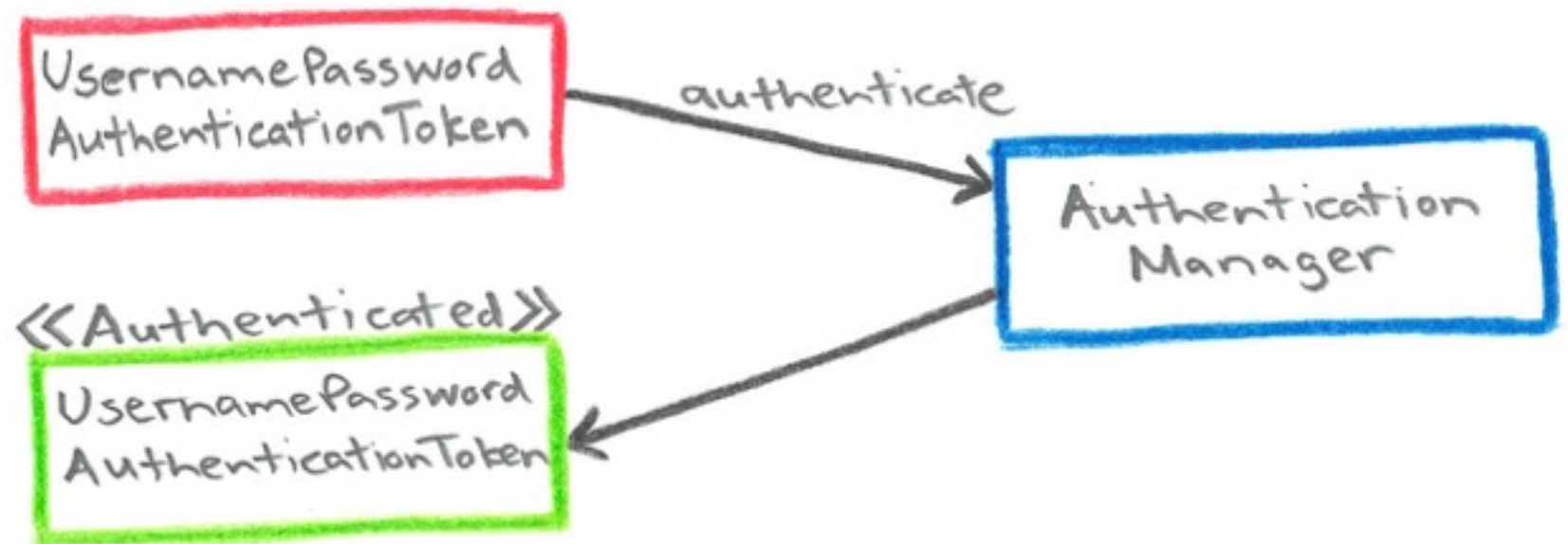
# Authentication Filter



# Authentication

Authentication	
Principal:	joe@example.com
Credentials:	password
Authorities:	—
Authenticated:	FALSE

Authentication	
Principal:	UserDetails
Credentials:	—
Authorities:	ROLE_USER
Authenticated:	TRUE



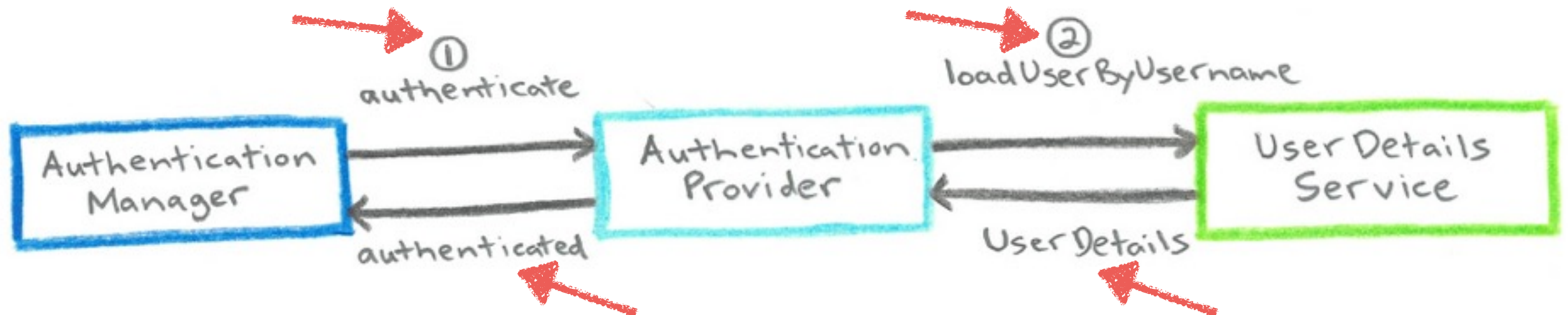
```
public interface Authentication extends Principal, Serializable {  
    Object getPrincipal();  
    Object getCredentials();  
    Collection<? extends GrantedAuthority> getAuthorities();  
    . . .  
}
```

# UserDetails / Service

```
public interface UserDetailsService {
```

```
    UserDetails loadUserByUsername(String username)  
        throws UsernameNotFoundException;
```

```
}
```



Authentication
Principal: UserDetails
Credentials: —
Authorities: ROLE_USER
Authenticated: TRUE

```
public interface UserDetails extends Serializable {
```

```
    String getUsername();
```

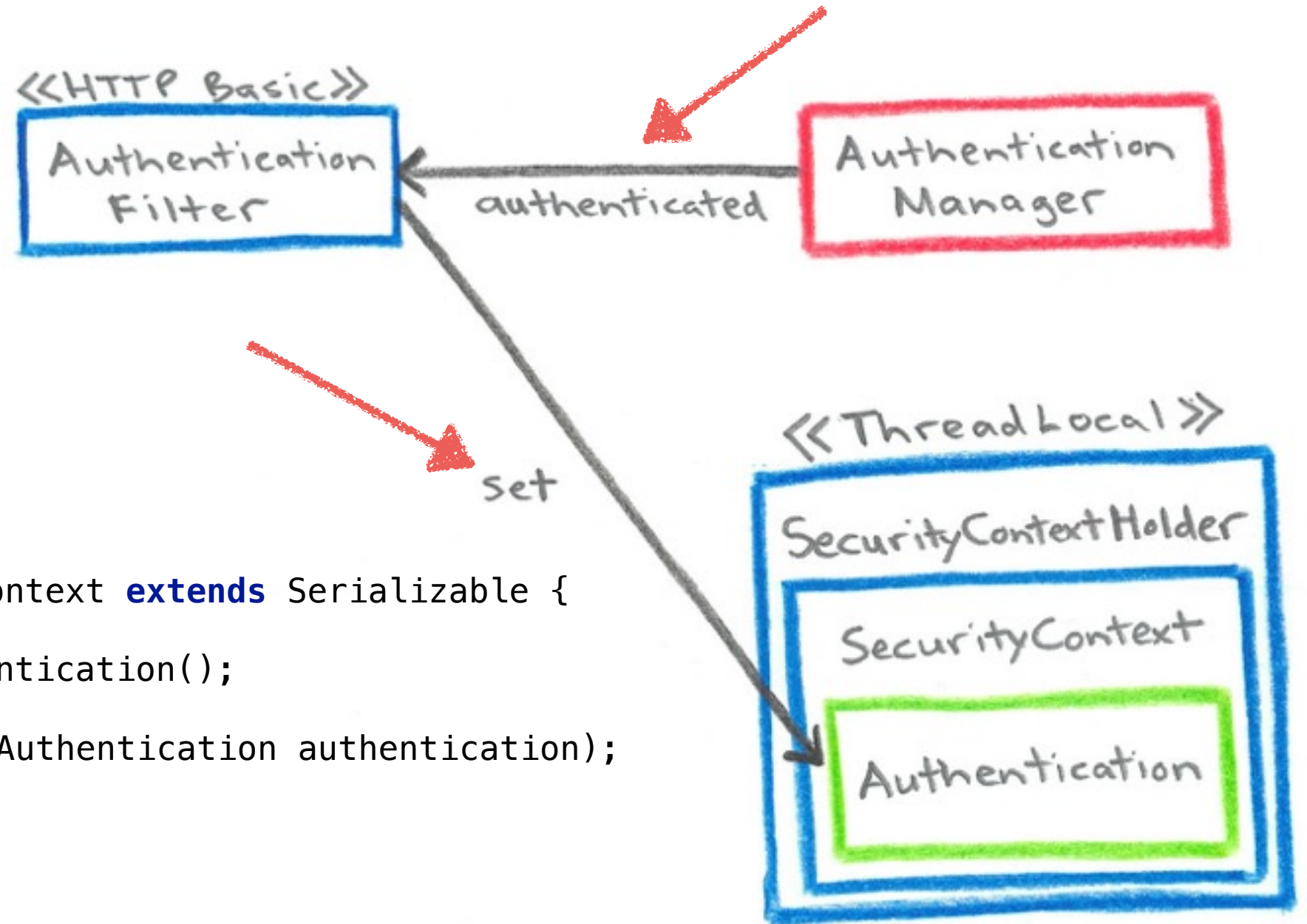
```
    String getPassword();
```

```
    Collection<? extends GrantedAuthority> getAuthorities();
```

```
    . . .
```

```
}
```

# Security Context



```
public interface SecurityContext extends Serializable {  
    Authentication getAuthentication();  
    void setAuthentication(Authentication authentication);  
}
```

```
SecurityContextHolder.getContext().setAuthentication(authenticated);
```

# Authentication Recap

- **Authentication Filter** creates an “*Authentication Request*” and passes it to the **Authentication Manager**
- Authentication Manager delegates to the **Authentication Provider**
- Authentication Provider uses a **UserDetailsService** to load the **UserDetails** and returns an “*Authenticated Principal*”
- Authentication Filter sets the **Authentication** in the **SecurityContext**

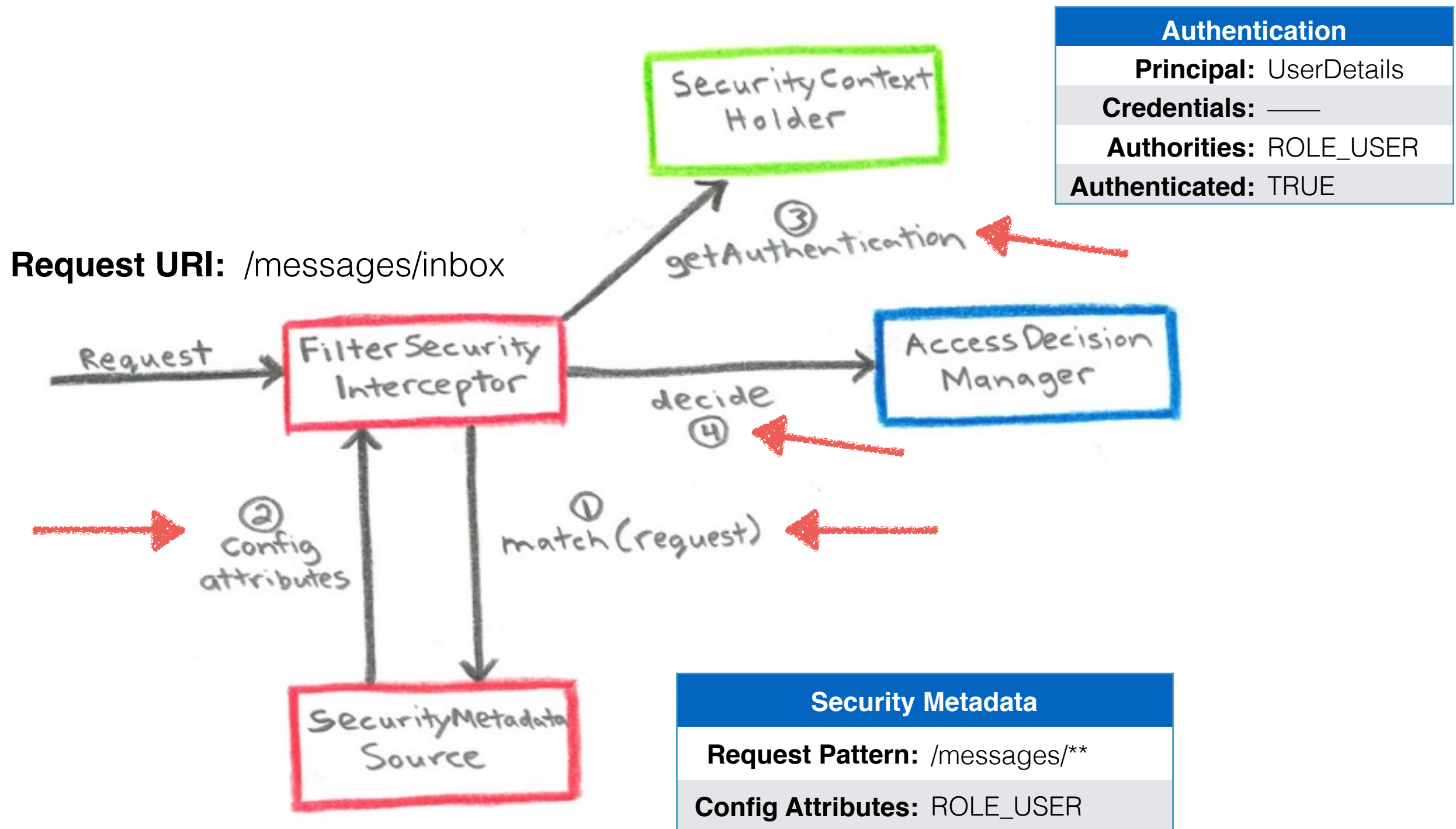


Authentication

**Authorization**

Exception Handling

# Filter Security Interceptor



# Access Decision

## Authentication

**Principal:** UserDetails

**Credentials:** —

**Authorities:** ROLE\_USER

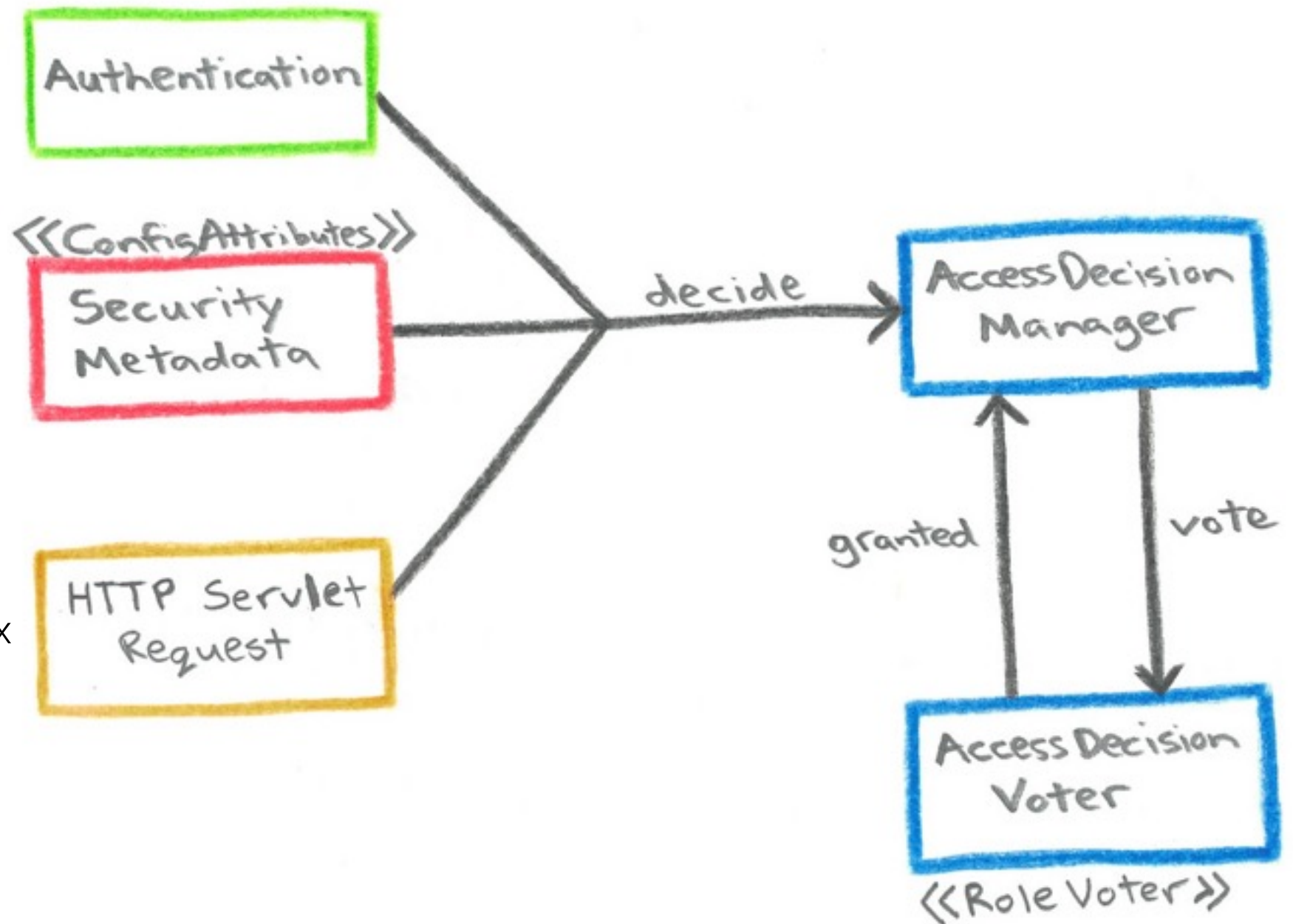
**Authenticated:** TRUE

## Security Metadata

**Request Pattern:** /messages/\*\*

**Config Attributes:** ROLE\_USER

**Request URI:** /messages/inbox



# Authorization Recap

- **FilterSecurityInterceptor** obtains the “***Security Metadata***” by matching on the current request
- FilterSecurityInterceptor gets the current **Authentication**
- The Authentication, Security Metadata and Request is passed to the **AccessDecisionManager**
- The AccessDecisionManager delegates to it's **AccessDecisionVoter(s)** for decisioning

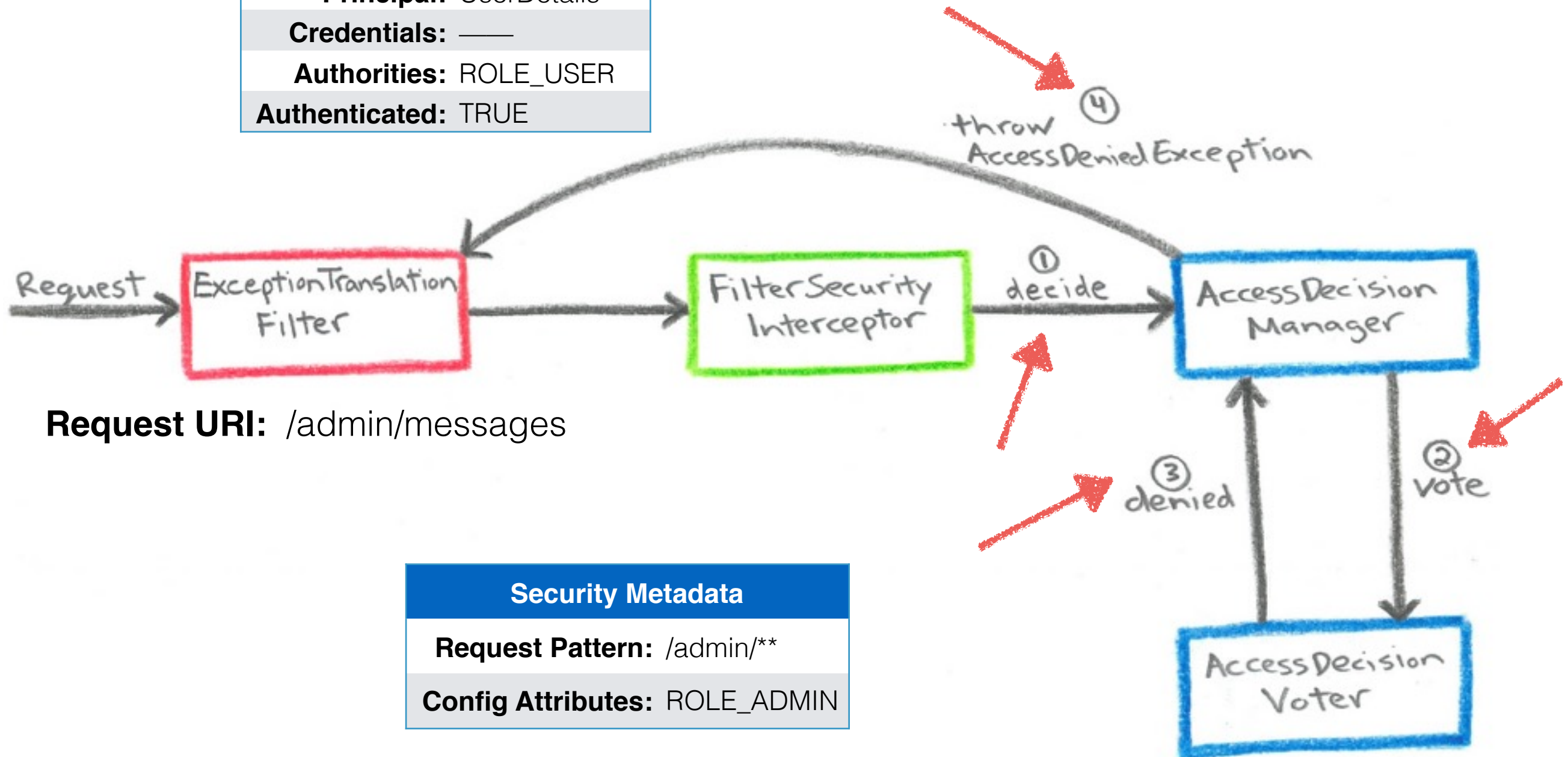
Authentication

Authorization

**Exception Handling**

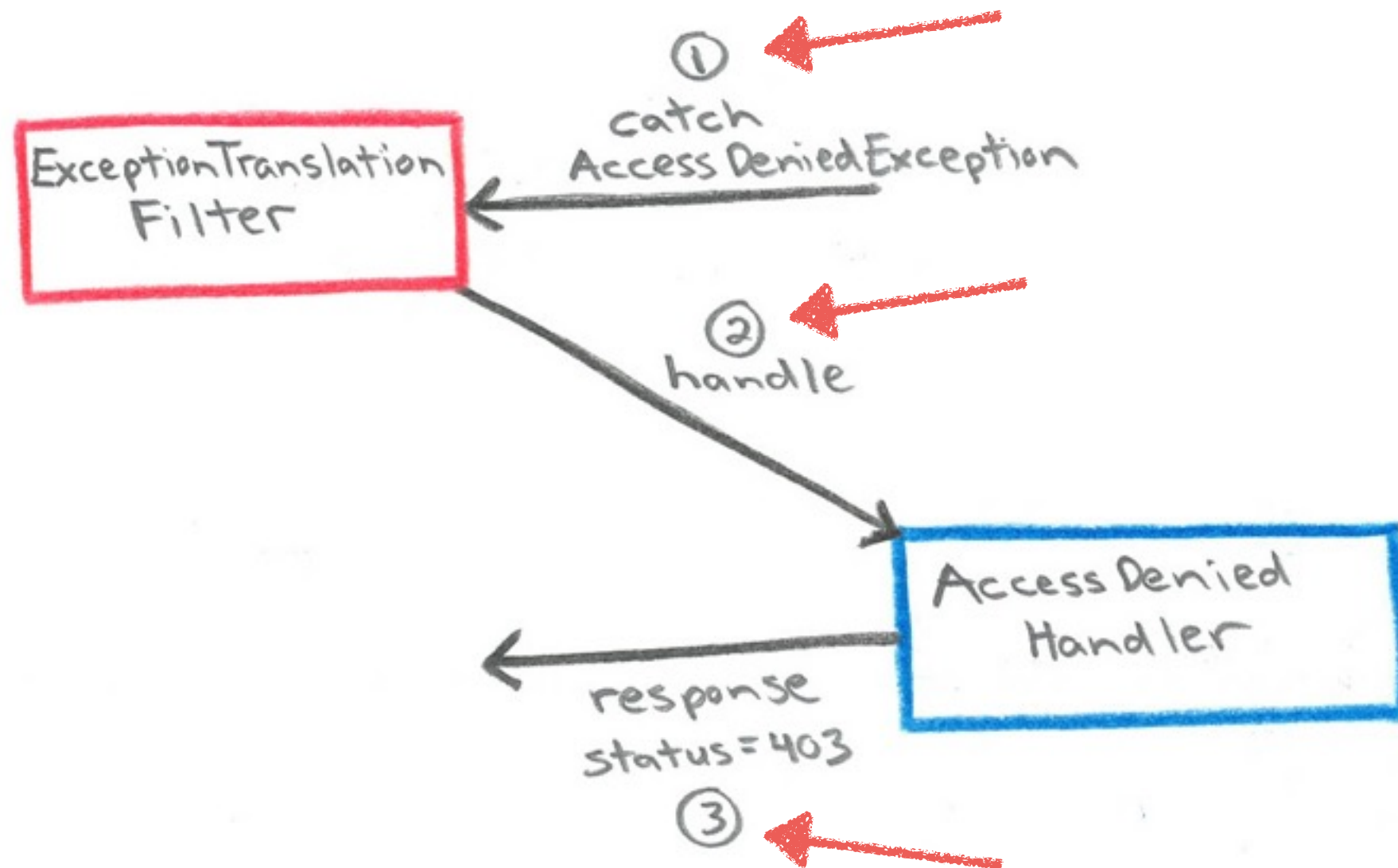
# Access Denied

Authentication	
Principal:	UserDetails
Credentials:	—
Authorities:	ROLE_USER
Authenticated:	TRUE



Security Metadata	
Request Pattern:	/admin/**
Config Attributes:	ROLE_ADMIN

# Access Denied Handler



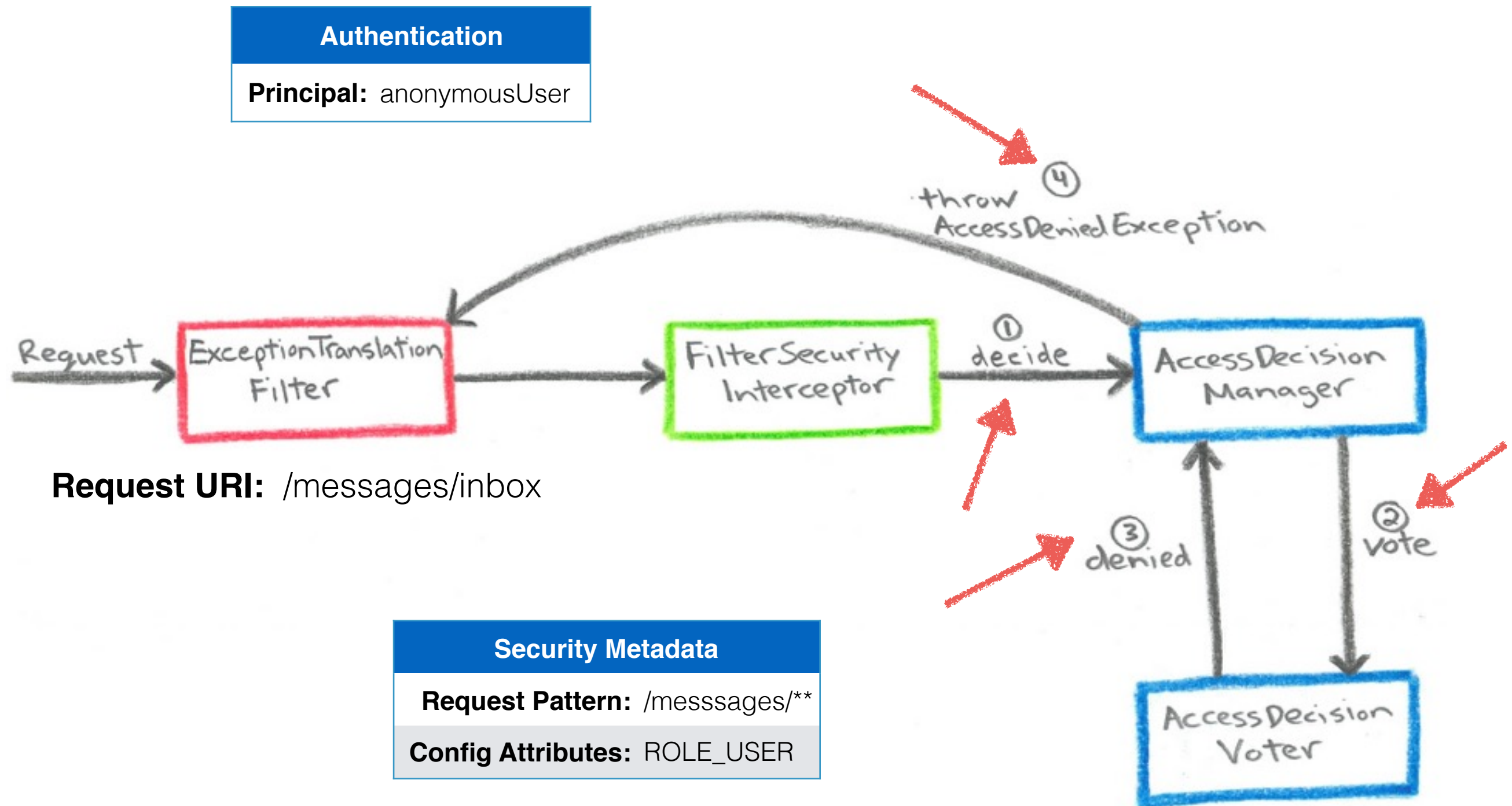
```
public interface AccessDeniedHandler {
```

```
    void handle(HttpServletRequest request, HttpServletResponse response,  
               AccessDeniedException accessDeniedException) throws IOException, ServletException;
```

```
}
```

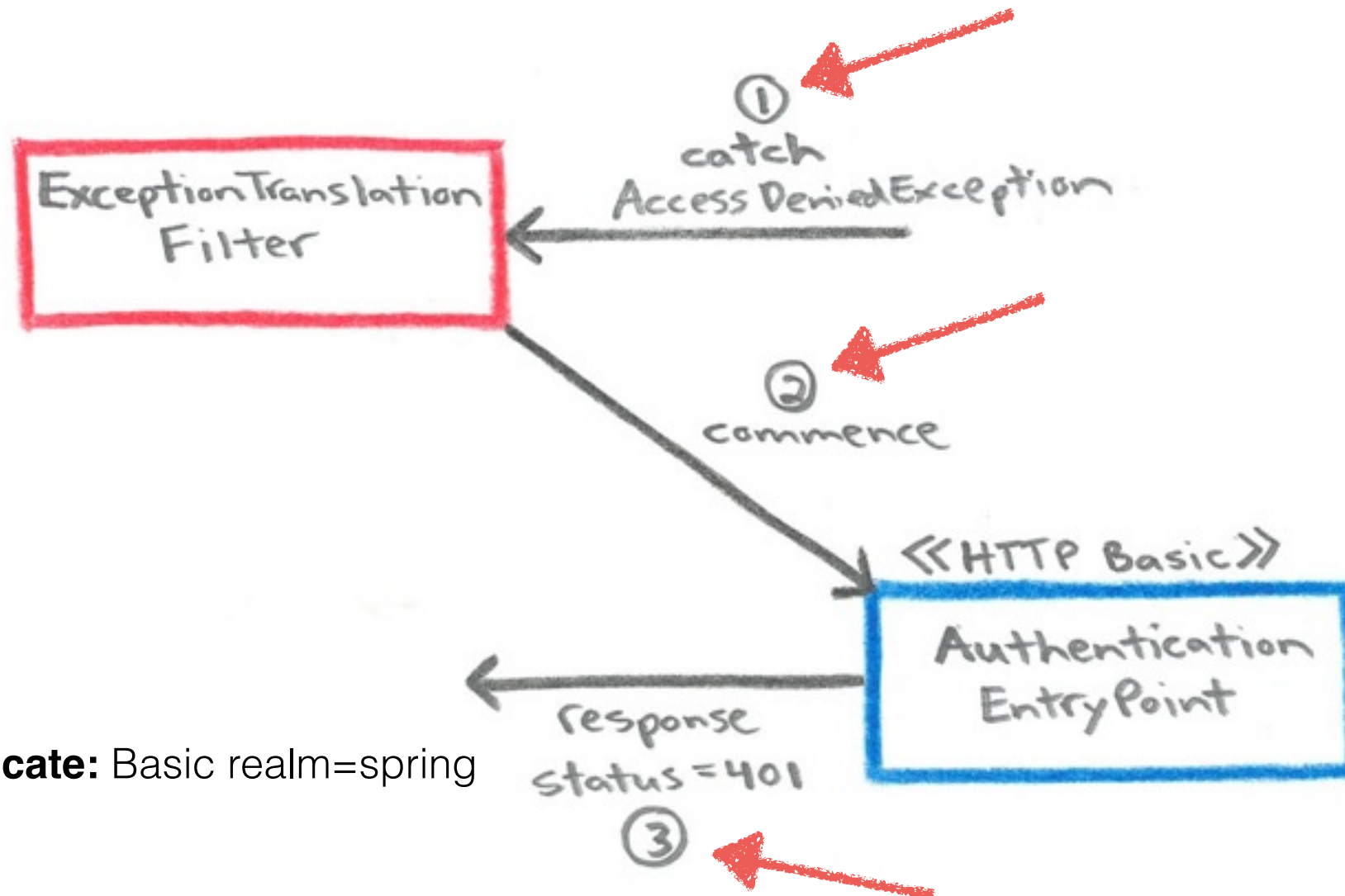


# “Unauthenticated”





# Start Authentication



**WWW-Authenticate:** Basic realm=spring

```
public interface AuthenticationEntryPoint {  
    void commence(HttpServletRequest request, HttpServletResponse response,  
        AuthenticationException authException) throws IOException, ServletException;  
}
```

# Exception Handling Recap

- When "*Access Denied*" for current Authentication, the **ExceptionTranslationFilter** delegates to the **AccessDeniedHandler**, which by default, returns a 403 Status.
- When current Authentication is "*Anonymous*", the **ExceptionTranslationFilter** delegates to the **AuthenticationEntryPoint** to start the Authentication process.

# Summary

## Authentication

## Authorization

## Exception Handling

Thank You!  
...for coming out

# Demo Sample

[github.com/jgrandja/messaging-sample](https://github.com/jgrandja/messaging-sample)

Joe Grandja  
Spring Security Team

[@jgrandja](https://github.com/jgrandja)  
[@joe\\_grandja](https://twitter.com/joe_grandja)

# Spring Security Filter Chain

