

MODERN AUTHENTICATION DEMYSTIFIED

A Deep Dive into Spring Security's Latest Innovations

OWASP Stuttgart Chapter
Stammtisch

3.6.2025

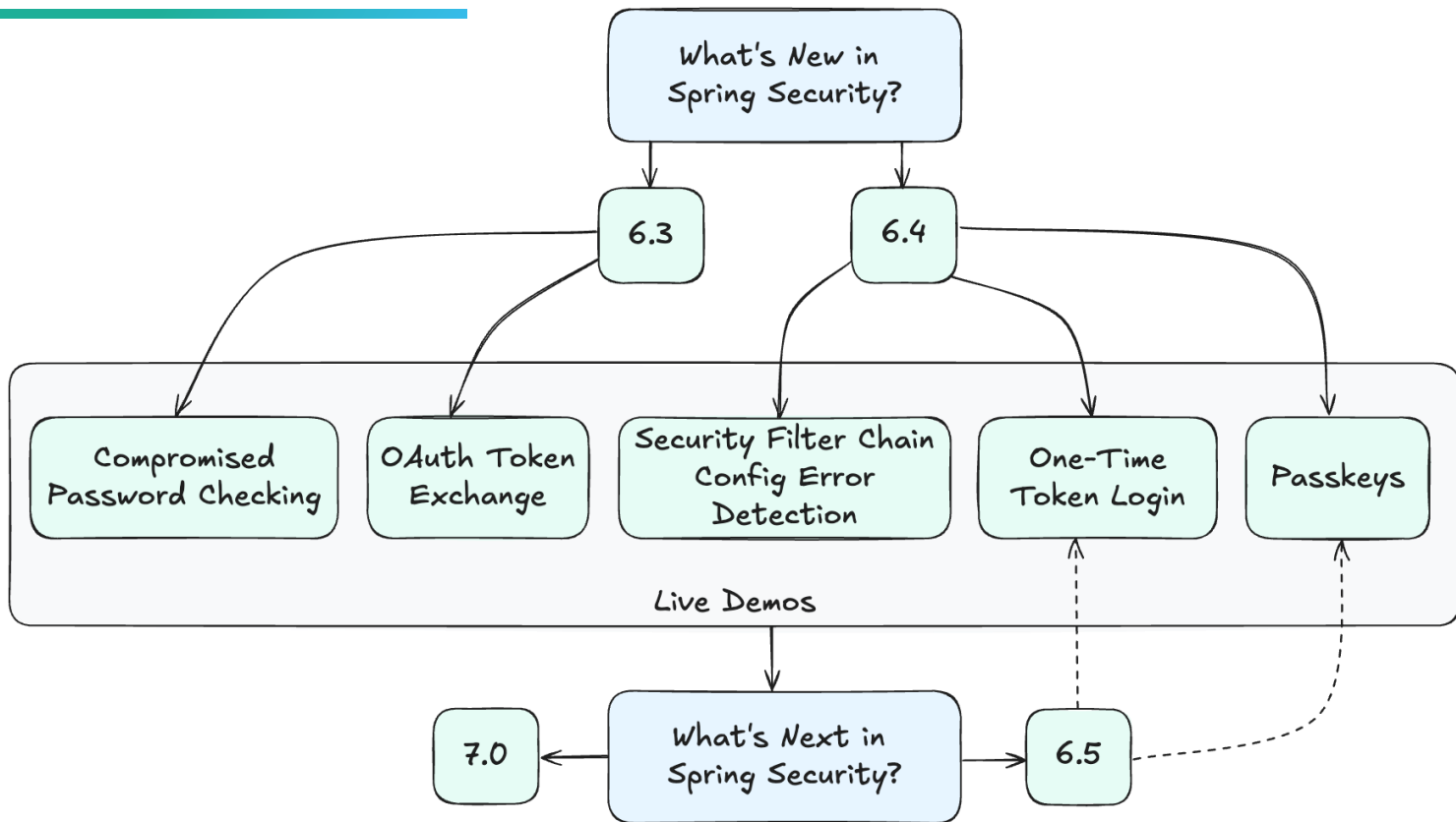


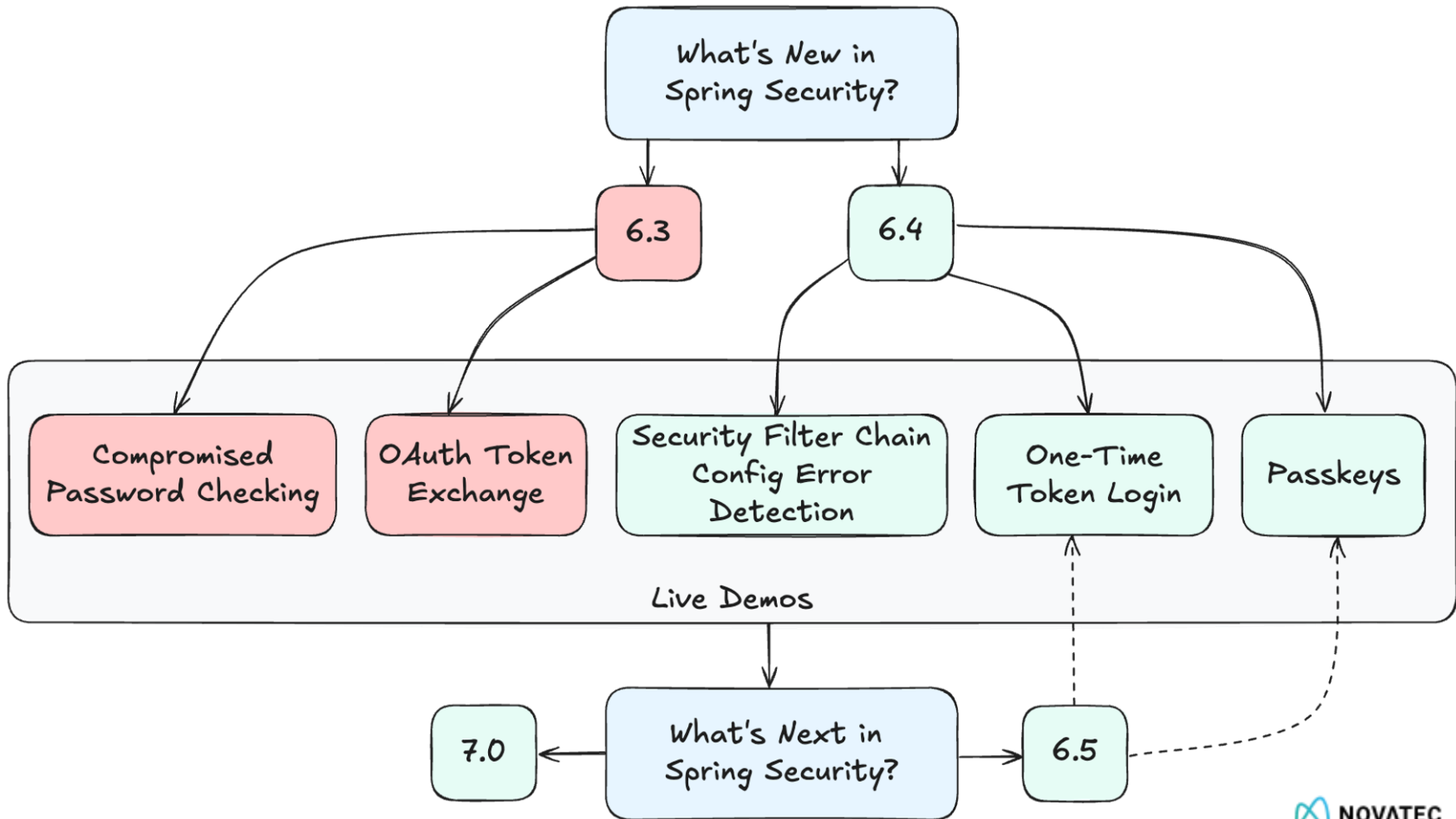
About Me



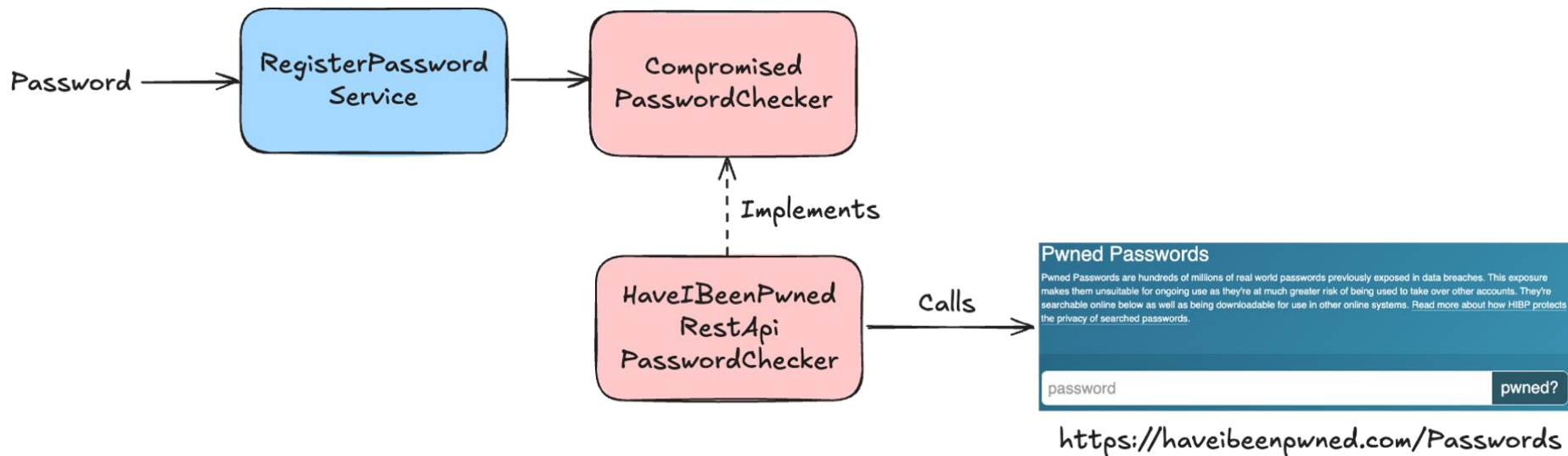
<https://www.linkedin.com/in/andifalk>

Hitchhikers Guide for this Talk





Compromised Password Checking (Spring Security 6.3)



<https://docs.spring.io/spring-security/reference/6.3/features/authentication/password-storage.html#authentication-compromised-password-check>

DEMOS



<https://github.com/andifalk/whats-new-in-spring-security>

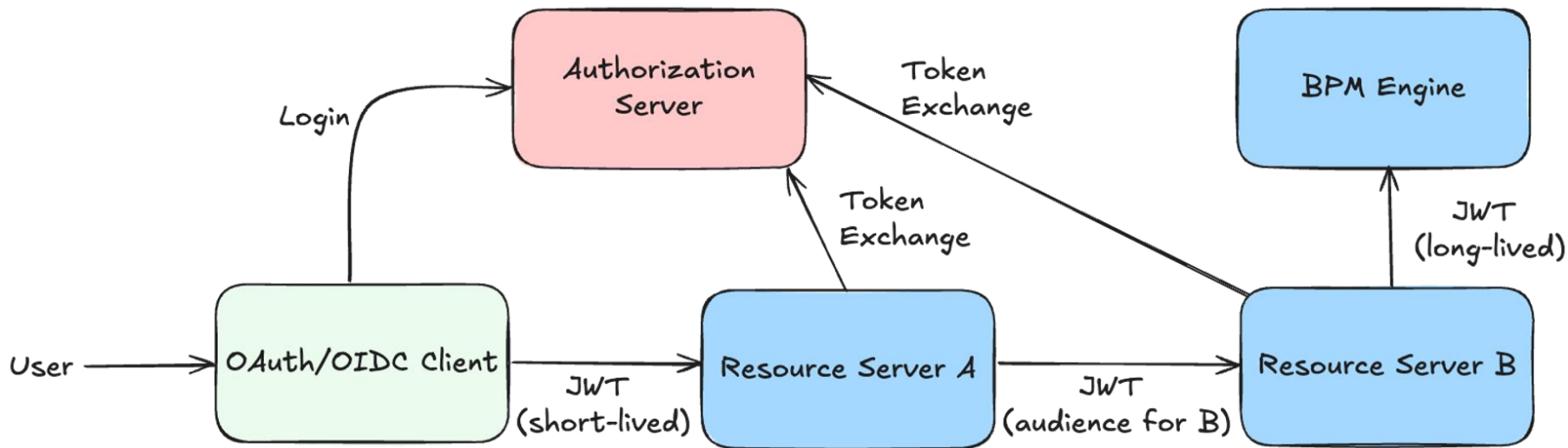
Compromised Password Checking (Spring Security 6.3)

Even Better...

Get Rid of Passwords completely 🤗

→ See **OAuth 2.1** just next & **Passkeys** a bit later...

OAuth Token Exchange (Spring Security 6.3)



https://docs.spring.io/spring-security/reference/6.3/whats-new.html#_oauth_2_0_token_exchange_grant_5199

OWASP ASVS 5: OAuth Resource Server

- ASVS V5.0 released at OWASP AppSec EU 2025 in Barcelona

V10.3 OAuth Resource Server

In the context of ASVS and this chapter, the resource server is an API. To provide secure access, the resource server must:

- Validate the access token, according to the token format and relevant protocol specifications, e.g., JWT-validation or OAuth token introspection.
- If valid, enforce authorization decisions based on the information from the access token and permissions which have been granted. For example, the resource server needs to verify that the client (acting on behalf of RO) is authorized to access the requested resource.

Therefore, the requirements listed here are OAuth or OIDC specific and should be performed after token validation and before performing authorization based on information from the token.

#	Description	Level
10.3.1	Verify that the resource server only accepts access tokens that are intended for use with that service (audience). The audience may be included in a structured access token (such as the 'aud' claim in JWT), or it can be checked using the token introspection endpoint.	2

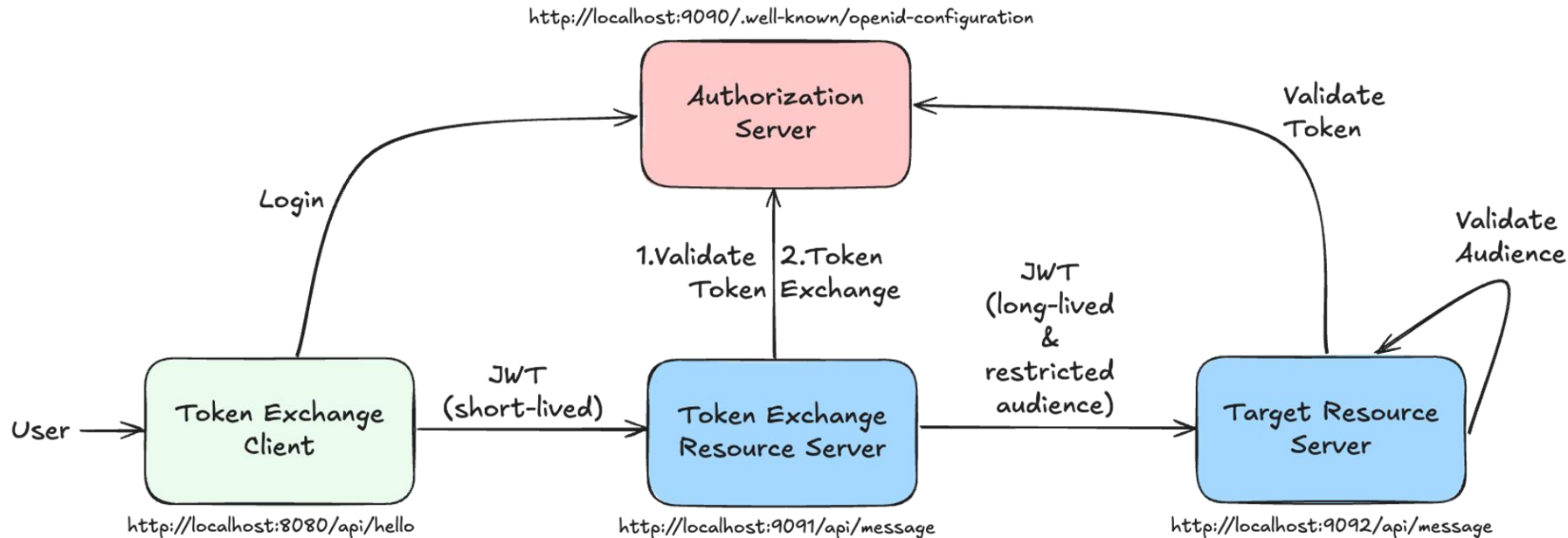
Impersonation vs. Delegation in OAuth Token Exchange

Comparison Table (OAuth Token Exchange)

Feature	Impersonation	Delegation
RFC 8693 Usage	Only subject_token	subject_token + actor_token
Identity in access token	Subject only (sub=user)	Subject + Actor (sub=user, act=caller)
Use case example	Login as user / Support admin	Microservice acting on behalf of a user
Auditability	Limited – appears as user only	Full – includes caller identity
Security risk	Higher – hides real caller	Lower – maintains trust chain

<https://www.rfc-editor.org/rfc/rfc8693.html>

Impersonation vs. Delegation in OAuth Token Exchange



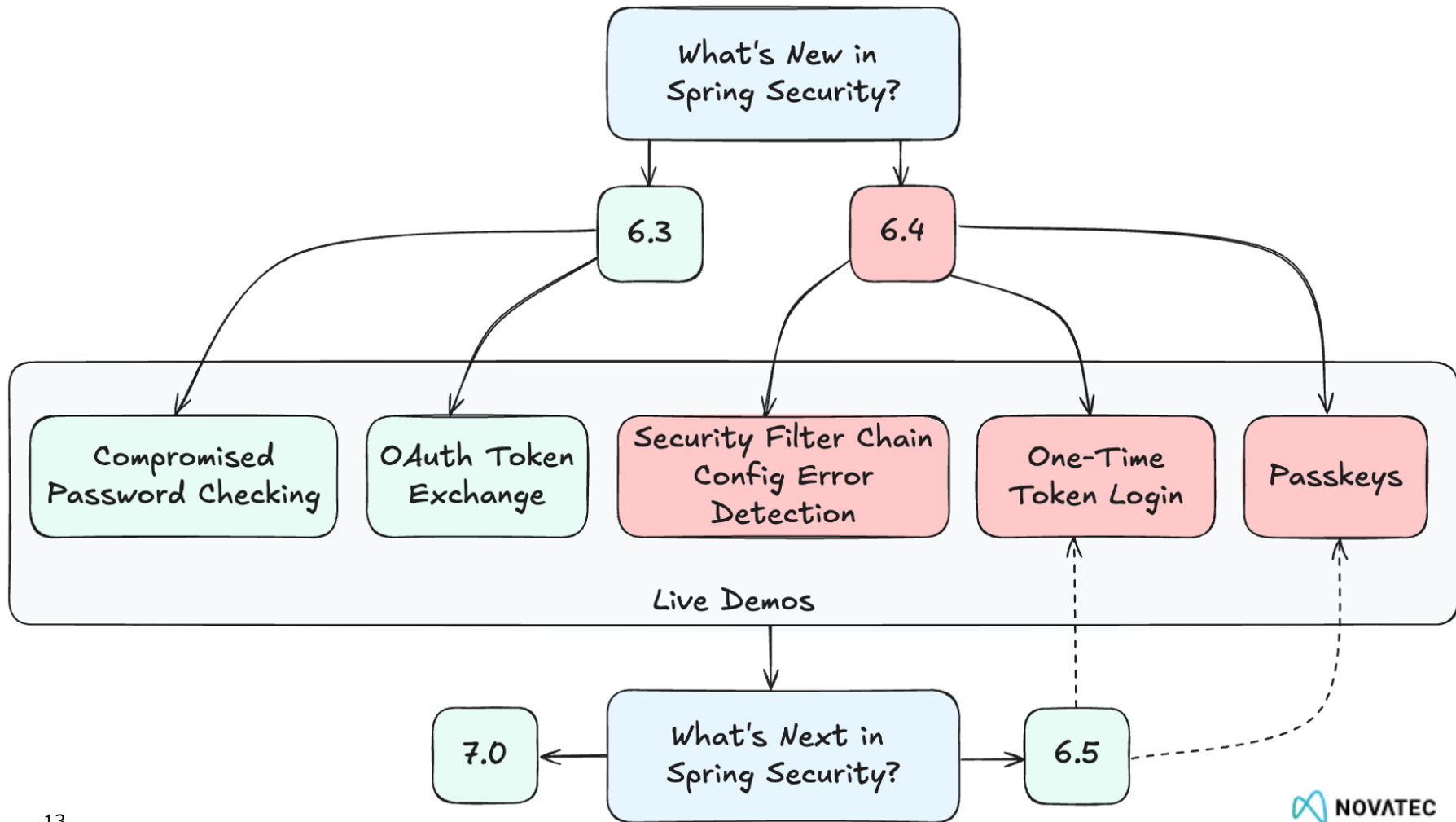
https://docs.spring.io/spring-security/reference/6.3/whats-new.html#_oauth_2_0_token_exchange_grant_5199

<https://github.com/andifalk/whats-new-in-spring-security>

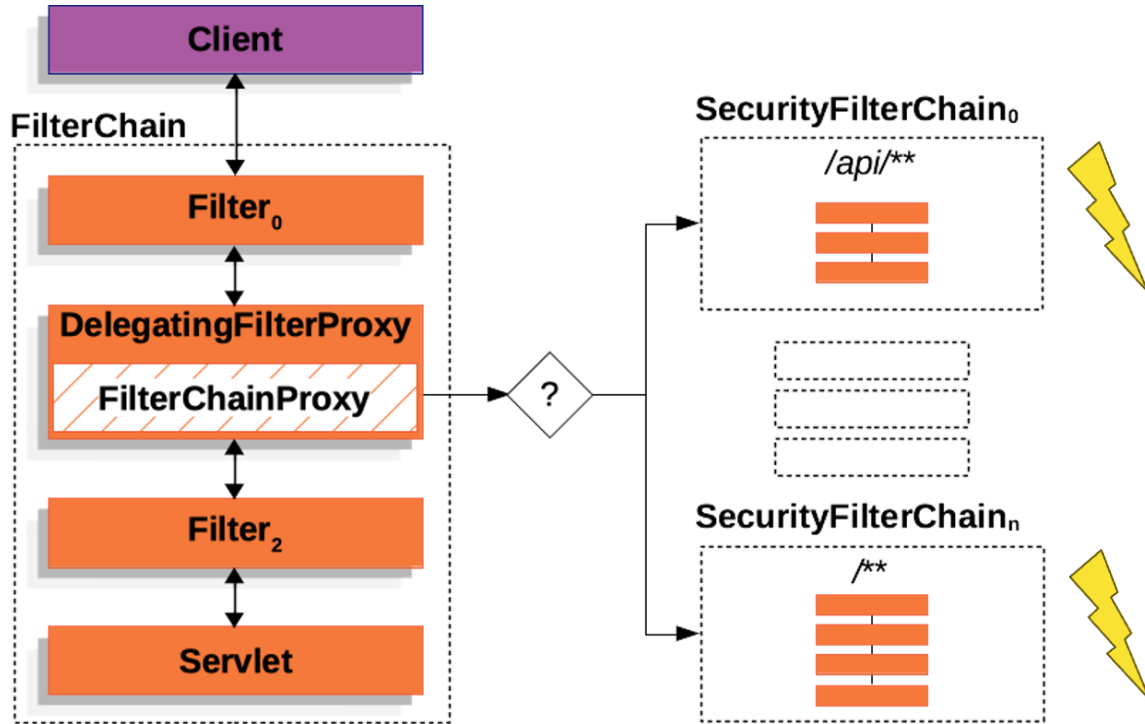
Further Improvements in Spring Security 6.3

- Authorization
 - Annotation Templates
 - Secure Return Values
 - Error Handling

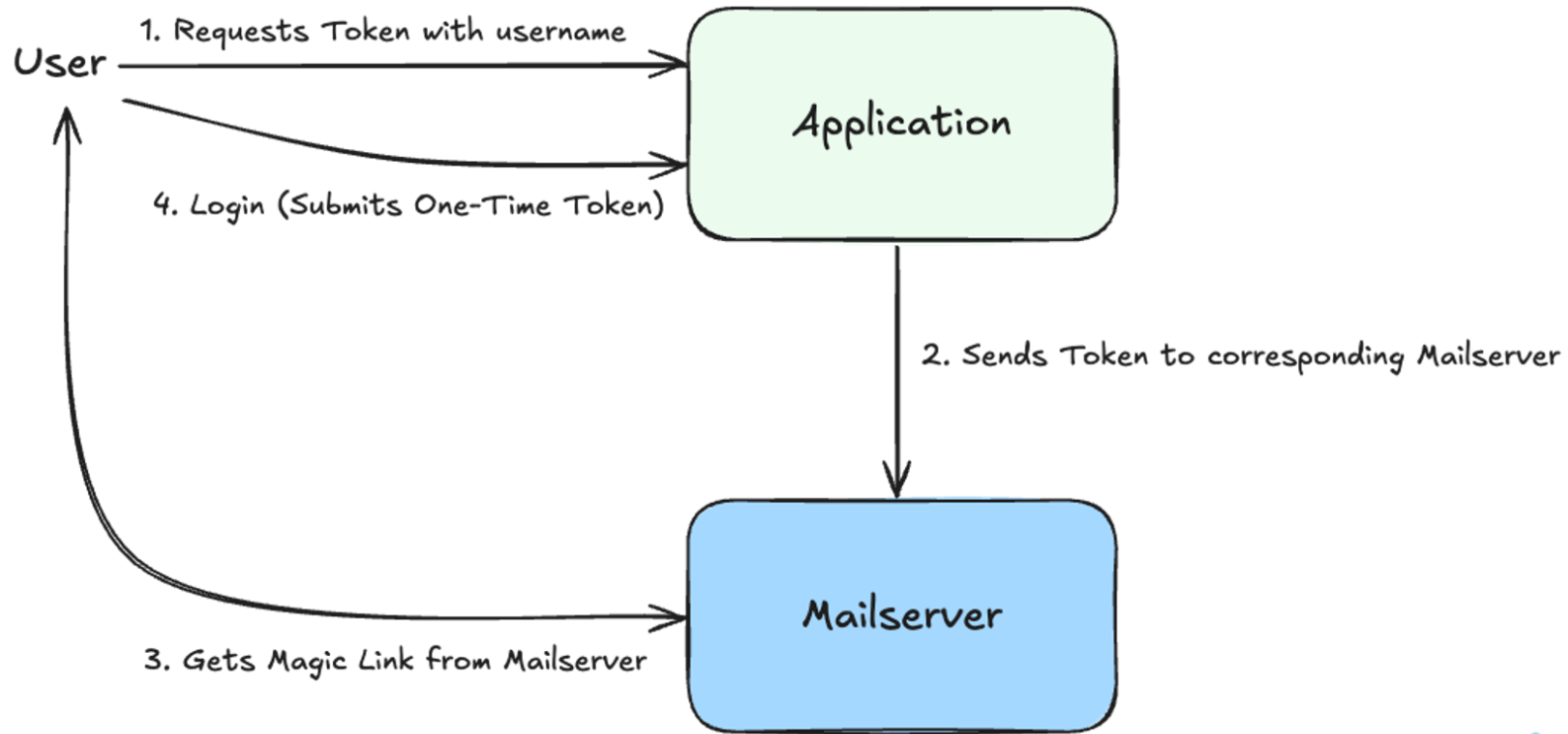
```
@Component
public class MaskMethodAuthorizationDeniedHandler implements MethodAuthorizationDeniedHandler {
    @Override
    public Object handleDeniedInvocation(
        MethodInvocation methodInvocation, AuthorizationResult authorizationResult) {
        return "*****";
    }
}
```



Security Filter Chain Error Detection (Spring Security 6.4)

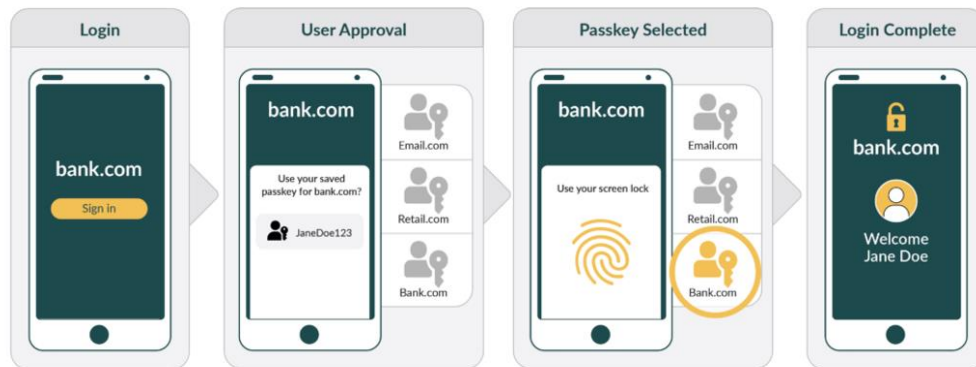
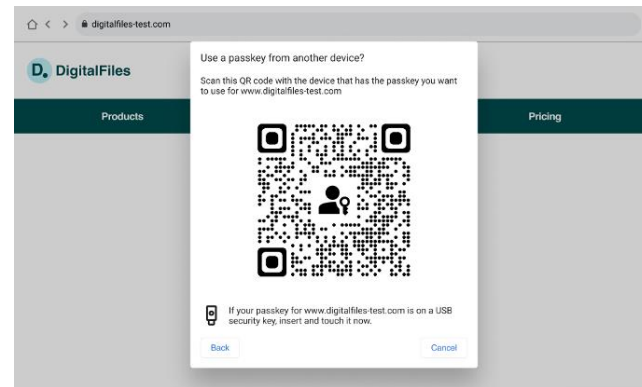


One-Time Token Login (Spring Security 6.4)




Passkey Authentication (Spring Security 6.4)

- Works across user's devices
- Strong and phishing-resistant
- Unique cryptographic public/private key pairs (passkeys) to every online service
- Replacement for Passwords



Passkeys Support and Relation to WebAuthn & FIDO2

-  WebAuthn:
The Engine
- ☐ Passkeys:
The user-friendly
car built around it
- ☐ Passkeys:
FIDO2 sign-in credentials

How Passkeys Work on Different Operating Systems

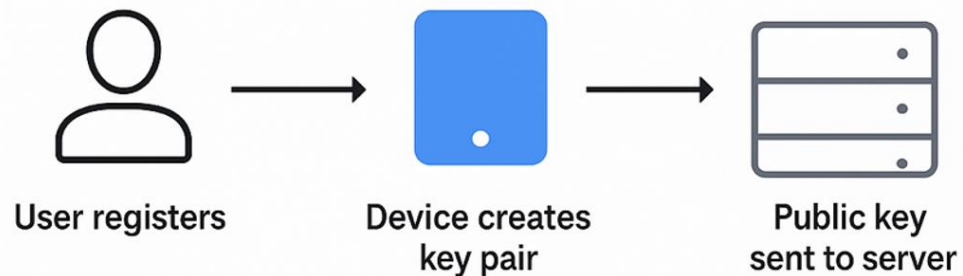
OS / Device	Where Passkeys Are Stored	Sync Across Devices?
macOS	iCloud Keychain	✓ Yes (via iCloud)
iOS	iCloud Keychain	✓ Yes
Windows 11	Windows Hello / Credential Manager	⚠ No (yet)
Android	Google Password Manager	✓ Yes (via Google Account)
Linux	Usually requires external authenticator (e.g. YubiKey)	✗ No built-in sync
Chrome OS	Google Account (like Android)	✓ Yes



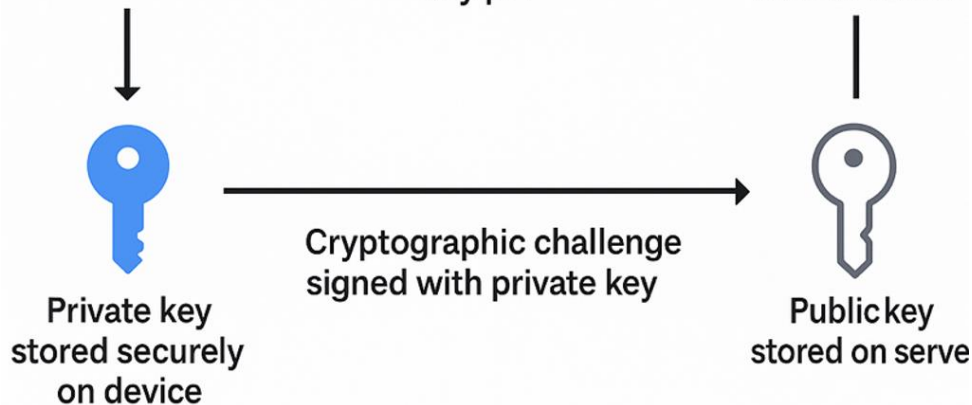
How Passkeys Work

Behind the Scenes with Key Pairs and Cryptographic Challenges

1.Register



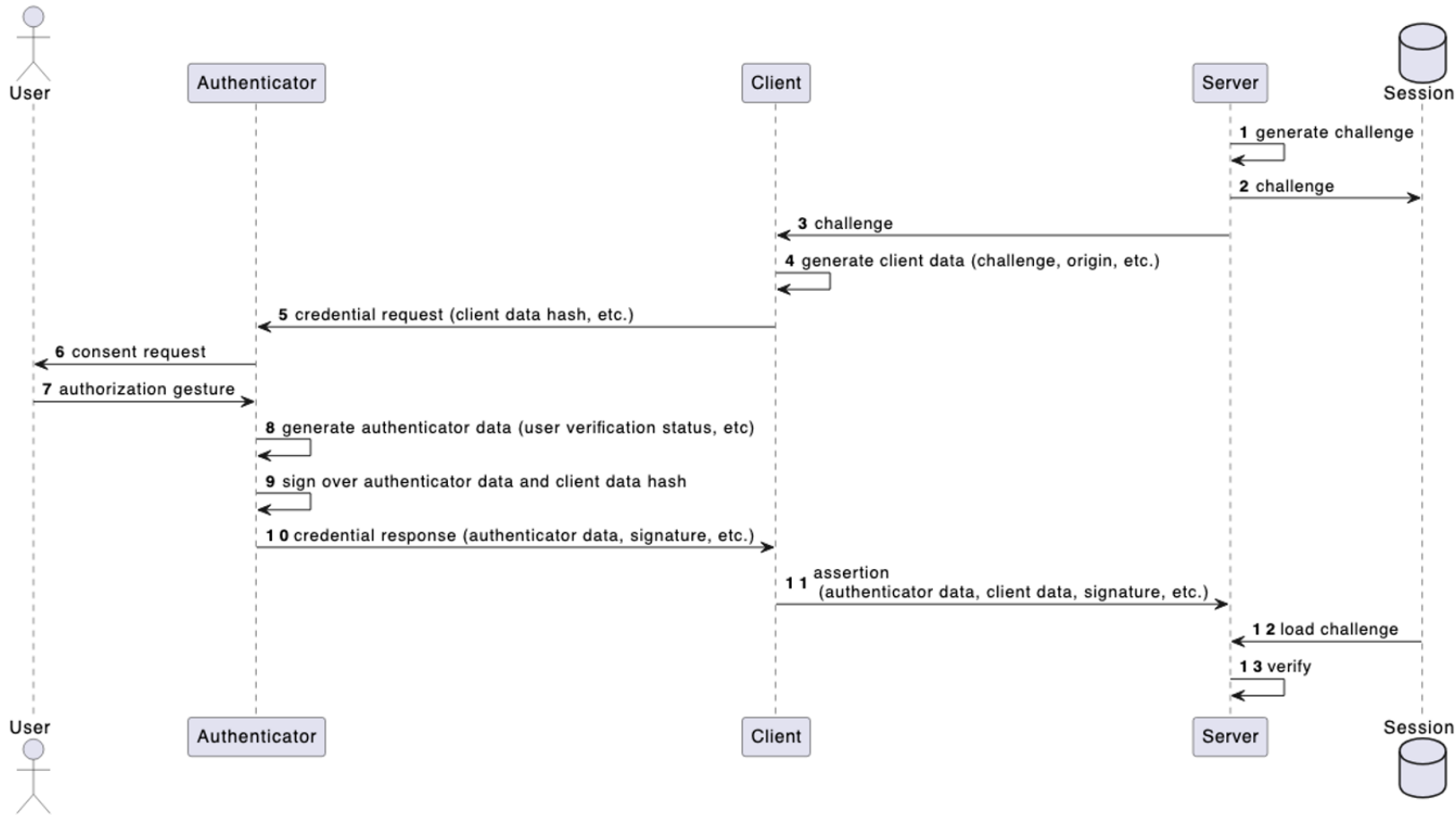
2.Authenticate



<https://www.w3.org/TR/webauthn-2>
<https://fidoalliance.org/passkeys>

Even More Details...

WebAuthn Authentication Overview



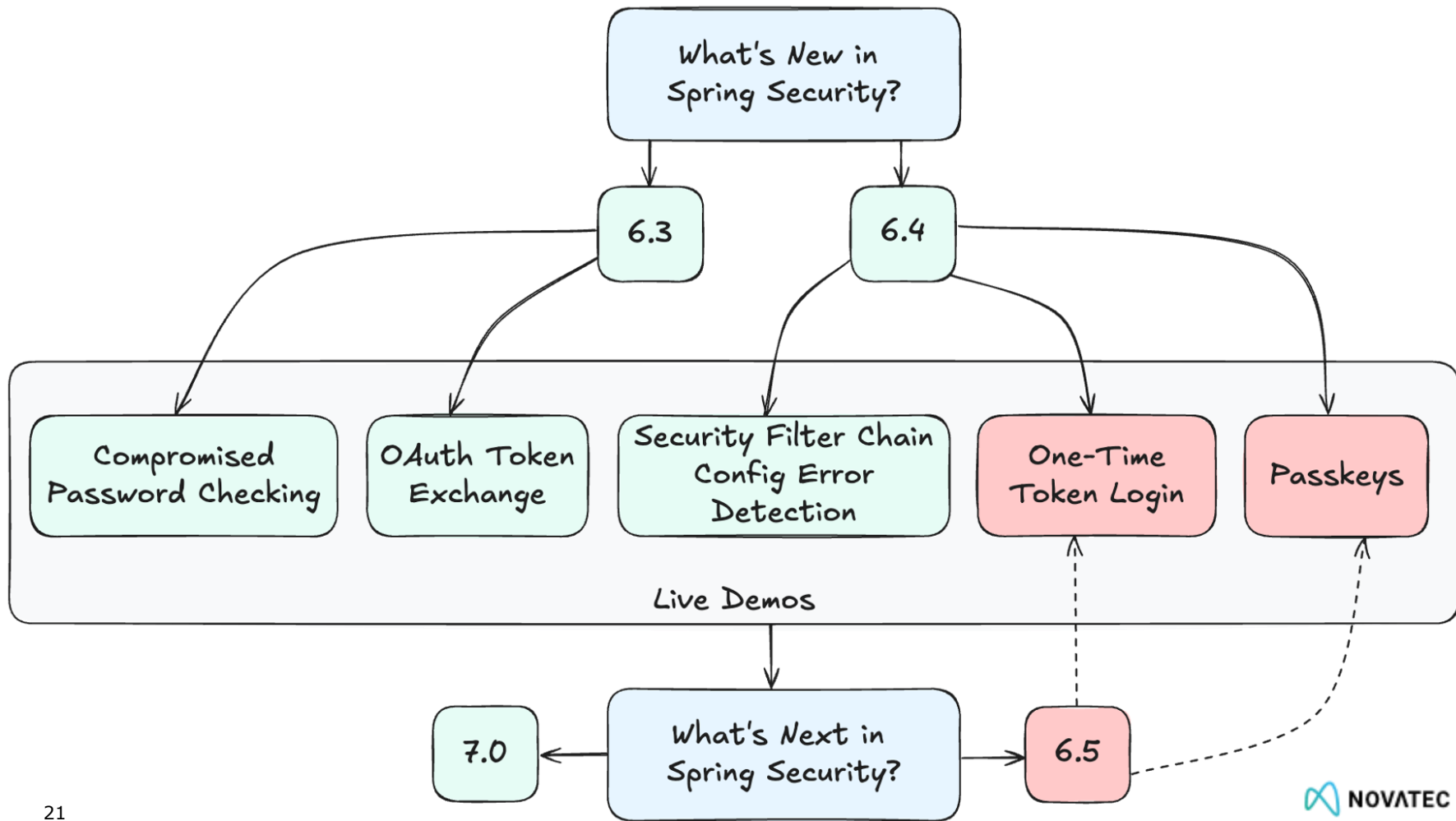
Further Improvements in Spring Security 6.4

- Authentication
 - OAuth 2.0 Support for RestClient
 - OpenSAML 5 Support
- Authorization
 - Annotation templates support for *@AuthenticationPrincipal* and *@CurrentSecurityContext*
- Improved Kotlin Support (*@PreFilter*/*@PostFilter*)

```
@Target(TargetType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@AuthenticationPrincipal("claims['{claim}']")
@interface CurrentUsername {
    String claim() default "sub";
}

// ...

@GetMapping
public String method(@CurrentUsername("username") String username) {
    // ...
}
```



What's Next in Spring Security 6.5?

Part of Spring Boot 3.5

- Support for OAuth 2.0 Demonstrating Proof of Possession (RFC-9449 - DPOP)
- JDBC Persistence for WebAuthn/Passkeys
- Customizing One-Time Token Request
- Allow **at+jwt** for bearer tokens, according to RFC-9068 (JWT Profile for OAuth 2.0 Access Tokens)

<https://docs.spring.io/spring-security/reference/6.5/whats-new.html>

<https://spring.io/blog/2025/05/19/spring-security-6-5-0-is-out>

<https://www.rfc-editor.org/rfc/rfc9449.html>

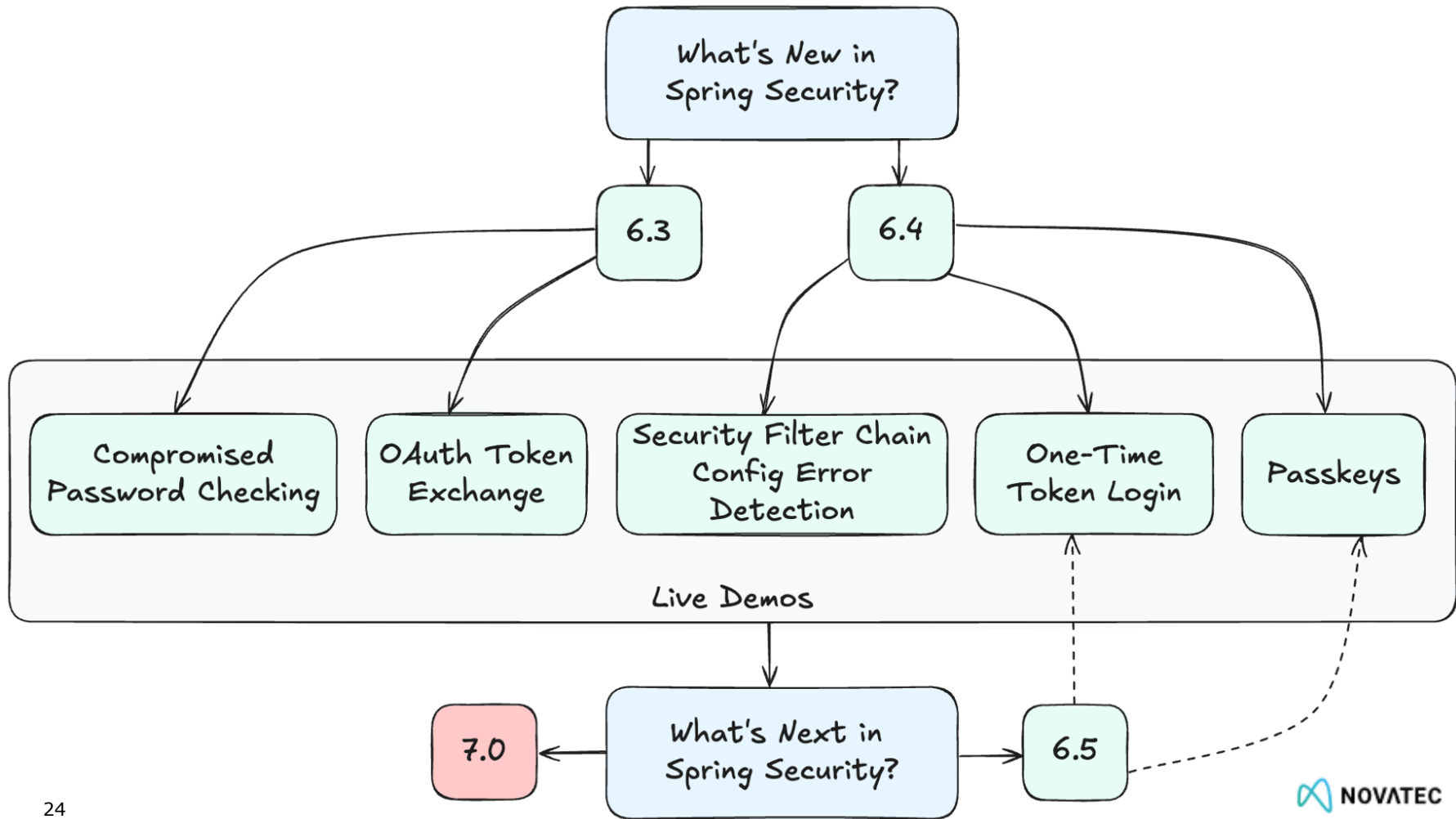
<https://www.rfc-editor.org/rfc/rfc9068.html>

RFC 9449 – DPoP (Spring Security 6.5)

- Proof-of-Possession Tokens (Token Binding) for Public Clients (i.e., a SPA)
→ Alternative to BFF pattern for SPA
- Prevent unauthorized parties from using leaked or stolen access tokens
- Requires an IdP capable of DPoP like Spring Authorization Server V.1.5+

<https://www.rfc-editor.org/rfc/rfc9449.html>

<https://www.ietf.org/archive/id/draft-ietf-oauth-browser-based-apps-24.html#name-backend-for-frontend-bff>



What's Next in Spring Security 7.0?

- Breaking Changes !!!!
 - Mandatory use of Lambda DSL Configuration
 - Removal of Deprecated Code
- Enabling PKCE for Authorization Code by Default
- ...

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests(authorize -> authorize
                .requestMatchers("/blog/**").permitAll()
                .anyRequest().authenticated()
            )
            .formLogin(formLogin -> formLogin
                .loginPage("/login")
                .permitAll()
            )
            .rememberMe(Customizer.withDefaults());

        return http.build();
    }
}
```

<https://docs.spring.io/spring-security/reference/6.5/migration-7/index.html>
<https://github.com/spring-projects/spring-security/milestone/270> (7.0.0-M1)

What's New/Next in Spring Authorization Server

V1.3.0

- Mutual-TLS Client Certificate-Bound Access Tokens
- OAuth 2.0 Token Exchange
- Multi-Tenancy (Multiple Issuer)

V1.4.0

- SPA sample using Backend For Frontend and Spring Cloud Gateway
- OpenID Connect 1.0 *prompt=none* Support

V1.5.0

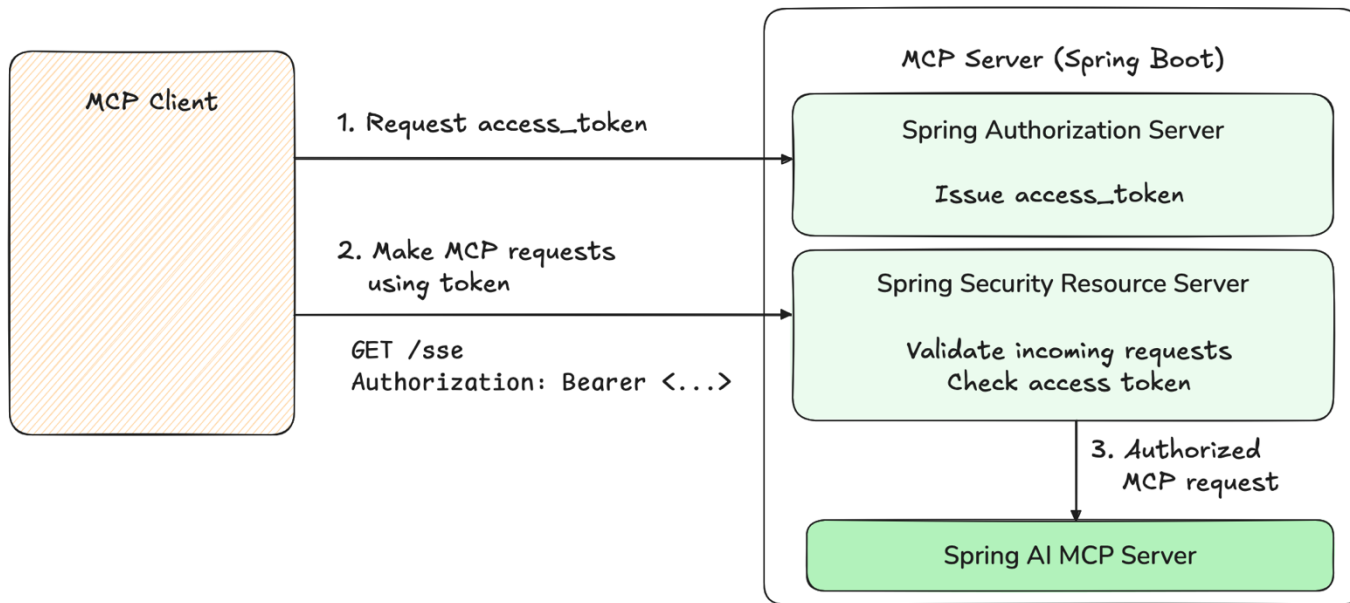
- OAuth 2.0 Pushed Authorization Requests (PAR)
- OAuth 2.0 Demonstrating Proof of Possession (DPoP)

<https://spring.io/blog/2025/05/20/spring-authorization-server-1-5-goes-ga>

<https://spring.io/blog/2024/11/19/spring-authorization-server-1-4-goes-ga>

<https://spring.io/blog/2024/05/22/spring-authorization-server-1-3-goes-ga>

AI Model Context Protocol (MCP) Security with OAuth2



<https://modelcontextprotocol.io/specification/2025-03-26>

<https://spring.io/blog/2025/04/02/mcp-server-oauth2>

<https://aaronparecki.com/2025/04/03/15/oauth-for-model-context-protocol>

Thanks! Questions?

LinkedIn



<https://www.linkedin.com/in/andifalk>



<https://github.com/andifalk/whats-new-in-spring-security>