1. أمان الخادم (Spring Security)

عبارة عن إطار مصادقة قوي وقابل للتخصيص بدرجة كبيرة وإطار عمل للتحكم في الوصول.

إنه المعيار الواقعي لتأمين التطبيقات القائمة على Spring.

و هو إطار عمل يركز على توفير كل من المصادقة (authentication) وتعني أنه يمكن للمستخدم الوصول إلى النظام الحالي، والترخيص (authorization) و يعني تحديد الحقوق الوظيفية التي يتمتع بها المستخدم.

مثل جميع مشاريع Spring ، تكمن القوة الحقيقية لـ Spring Security في مدى سهولة توسيعها لتلبية المتطلبات المخصصة.

1.1 مميزاته :

- دعم شامل وقابل للتوسيع لكل من المصادقة والترخيص
- لحماية من الهجمات مثل session fixation ، clickjacking، cross site request forgery، وما إلى ذلك
- تكاملServlet API
- تكامل اختياري معSpring Web MVC
- حمايةCSRF
- دعم تكوين جافا

1.2 المصادقة (Authentication) :

المصادقة هي عملية تتحقق من أن شخصًا ما هو ما يقوله. تستخدم أنظمة التكنولوجيا عادةً شكلاً من أشكال المصادقة لتأمين الوصول إلى تطبيق أو بياناته ، وعادةً ما يتم ذلك من خلال التحقق من اسم المستخدم وكلمة المرور. على سبيل المثال ، عندما تحتاج إلى الوصول إلى موقع أو خدمة عبر الإنترنت ، فعادة ما يتعين عليك إدخال اسم المستخدم وكلمة المرور الخاصين بك. ثم تتم مقارنة اسم المستخدم وكلمة المرور اللذين أدخلتهما مع السجل الموجود في قاعدة البيانات الخاصة به. إذا كانت المعلومات التي أرسلتها متطابقة ، يفترض النظام أنك مستخدم صالح ويمنحك حق الوصول.

1.3 الترخيص (Authorization) :

الترخيص هو عملية الأمان التي تحدد مستوى وصول المستخدم أو الخدمة. في التكنولوجيا ، نستخدم الترخيص لمنح المستخدمين أو الخدمات إذنًا للوصول إلى بعض البيانات أو تنفيذ إجراء معين. على سبيل المثال ، العملاء الذين لديهم حق الوصول إلى الواجهة الأمامية للموقع الخاص بك ، والمسؤولين الذين لديهم حق الوصول إلى منطقة إدارة منفصلة. يحتاج كلا النوعين من المستخدمين إلى تسجيل الدخول ، ولكن مجرد المصادقة لا توضح شيئًا عما يُسمح لهم بالقيام به في نظامك. وبالتالي ، تحتاج أيضًا إلى التحقق من أذونات المستخدم المصادق عليه ، أي أنك تحتاج إلى تخويل المستخدم.

من الضروري ملاحظة الاختلاف هنا بين المصادقة والترخيص. تتحقق المصادقة من المستخدم قبل السماح له بالوصول ، ويحدد الترخيص ما يمكنه فعله بمجرد أن يمنحه النظام حق الوصول.

ولكي يستطيع spring تحديد الترخيص لمسجل الدخول لابد من إضافة التعليق التوضيحي @AthanticationPrincipel في بوابة الطلب

كمثال

```
@PostMapping()
public ResponseEntity <ApiResponse> addTodos(@AuthenticationPrincipal
MyUser myUser, @RequestBody Todo todo){
    todoService.addTodo(myUser.getId(),todo);
    return ResponseEntity.status(HttpStatus.OK).body(new
ApiResponse("New Todo added !",201));
}
```

1.4 طريقة إضافة Spring Security إلى المشروع :

1. إضافةSpring Security Dependency

2. انشاء مجلدconfig

3. انشاء ملف SecurityConfig الذي يحتوي على جميع التكوينات

4. اضافة @Configuration وهو التعليق التوضيحي الذي يشير إلى أن الفئة لديها أساليب تعريف .Bean لذلك يمكن أن تقوم حاوية Spring بمعالجة الفصل وإنشاء Spring Beans لاستخدامها في التطبيق. هذا التعليق التوضيحي جزء من إطار العمل الأساسيSpring

5. اضافة @EnableWebSecurity وهو التعليق التوضيحي الذي يؤدي إلى تطبيق تكوين الأمان الافتراضي لـ Spring

6. إعادة تعريف دالةauthenticationProvider

```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {

    private final MyUserDetailsService myUserDetailsService;

    @Bean
    public DaoAuthenticationProvider authenticationProvider(){
        DaoAuthenticationProvider authenticationProvider=new
DaoAuthenticationProvider();

authenticationProvider.setUserDetailsService(myUserDetailsService);
        authenticationProvider.setPasswordEncoder(new
BCryptPasswordEncoder());
        return authenticationProvider;
    }
```

7. انشاء MyUserDetailsService الذي يقوم بتنفيذ واجهة UserDetailsService

8. تعريف دالة loadUserByUsername

```
@Service
@RequiredArgsConstructor
public class MyUserDetailsService  implements UserDetailsService {
```

```
        private final AuthRepository authRepository;
        @Override
        public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
    MyUser myUser=authRepository.findMyUserByUsername(username);
    if(myUser==null){
     throw new UsernameNotFoundException("Wrong username or
password");
            }
            return myUser;
        }
```

9. جعل MyUser تنفذ واجهة UserDetails وتعريف جميع الدوال الخاصة بالواجهة.

```
@Entity
@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class MyUser implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String username;
    private String password;
    private String role;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
   return Collections.singleton(new
SimpleGrantedAuthority(this.role));
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

10. إعادة تعريف دالة securityFilterChain(HttpSecurity http) داخل ملف SecurityConfig

```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {

    private final MyUserDetailsService myUserDetailsService;

    @Bean
    public DaoAuthenticationProvider authenticationProvider(){
        DaoAuthenticationProvider authenticationProvider=new
DaoAuthenticationProvider();

authenticationProvider.setUserDetailsService(myUserDetailsService);
        authenticationProvider.setPasswordEncoder(new
BCryptPasswordEncoder());
        return authenticationProvider;
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http)
throws Exception{
        http.csrf().disable()
                .sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.IF_REQUIRED)
                .and()
                .authenticationProvider(authenticationProvider())
                .authorizeHttpRequests()

.requestMatchers(HttpMethod.POST,"/api/v1/auth/register").permitAll()

.requestMatchers("/api/v1/auth/admin").hasAuthority("ADMIN")
                .anyRequest().authenticated()
                .and()
                .logout().logoutUrl("/api/v1/auth/logout")
                .deleteCookies("JSESSIONID")
                .invalidateHttpSession(true)
                .and()
                .httpBasic();
        return http.build();
    }


}
```