CS544 EA

Applications

Spring Security: Authentication Providers

Plain Text

- So far we've used plain text: **bad for security**

```
@Bean
public UserDetailsService userDetailsService() {
    UserDetails user = User.withDefaultPasswordEncoder()
            .username("user").password("pass").roles("USER").build();
    UserDetails admin = User.withDefaultPasswordEncoder()
            .username("admin").password("admin").roles("ADMIN", "USER").build();
    return new InMemoryUserDetailsManager(user, admin);
}
```

```
<authentication-manager>
    <authentication-provider>
        <user-service>
            <user name="test" password="{noop}123" authorities="ROLE_USER, ROLE_ADMIN" />
            <user name="bob" password="{noop}bobiscool" authorities="ROLE_USER" />
        </user-service>
    </authentication-provider>
</authentication-manager>
```

Password Encoder

- Important: **Never store plain text**
  - Basic hashing isn't that great either

```java
@Bean
public BCryptPasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
@Bean
public UserDetailsService users() {
    UserDetails user = User.withUsername("user")
        .password("{bcrypt}$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0FxO/BTk76klW").roles("USER").build();
    UserDetails admin = User.withUsername("admin")
        .password("{bcrypt}$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0FxO/BTk76klW").roles("USER", "ADMIN").build();
    return new InMemoryUserDetailsManager(user, admin);
}
```

```xml
<sec:authentication-manager>
  <sec:authentication-provider>
    <sec:password-encoder hash="bcrypt"/>
    <sec:user-service>
      <sec:user name="jimi" password="{bcrypt}d7e6351eaa13189a5a3641bab846c8e8c69ba39f" authorities="ROLE_USER, ROLE_ADMIN" />
      <sec:user name="bob" password="{bcrypt}4e7421b1b8765d8f9406d87e7cc6aa784c4ab97f" authorities="ROLE_USER" />
    </sec:user-service>
  </sec:authentication-provider>
</sec:authentication-manager>
```

Can be: md4, md5, sha, sha-256, bcrypt

Bcrypt is recommended as it also auto-salts

JDBC Authenticator

For demo purposes these users are created here.
Normally they'd just be in the DB

```
@Bean
public UserDetailsManager users(DataSource dataSource) {
    UserDetails user = User.withUsername("user")
            .password("{bcrypt}$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0FxO/BTk76klW")
            .roles("USER").build();
        UserDetails admin = User.withUsername("admin")
            .password("{bcrypt}$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0FxO/BTk76klW")
            .roles("USER", "ADMIN").build();
    JdbcUserDetailsManager users = new JdbcUserDetailsManager(dataSource);
    users.createUser(user);
    users.createUser(admin);
    return users;
}
```

```
<sec:authentication-manager>
  <sec:authentication-provider>
    <sec:jdbc-user-service data-source-ref="dataSource" >
  <sec:user name="jimi" password="{bcrypt}d7e6351eaa13189a5a3641bab846c8e8c69ba39f" authorities="ROLE_USER, ROLE_ADMIN" />
  <sec:user name="bob" password="{bcrypt}4e7421b1b8765d8f9406d87e7cc6aa784c4ab97f" authorities="ROLE_USER" />
    </sec:jdbc-user-service>
  </sec:authentication-provider>
</sec:authentication-manager>

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name="driverClassName" value="com.mysql.jc.jdbc.Driver"/>
  <property name="url" value="jdbc:mysql://localhost/cs544"/>
  <property name="username" value="root"/>
  <property name="password" value="root"/>
</bean>
```

Standard Authentication Tables

JDBC authentication expects the following tables:

```
create table users(
    username varchar_ignorecase(50) not null primary key,
    password varchar_ignorecase(50) not null,
    enabled boolean not null
);
create table authorities (
    username varchar_ignorecase(50) not null,
    authority varchar_ignorecase(50) not null,
    constraint fk_authorities_users foreign key(username) references users(username)
);
create unique index ix_auth_username on authorities (username,authority);
```

Values could be inserted like so:

```
Insert into users values(“test”, “{bcrypt}d7e6351eaa13189a5a3641bab846c8e8c69ba39f”, 1”);
Insert into users values(“bob”, “{bcrypt}4e7421b1b8765d8f9406d87e7cc6aa784c4ab97f”, 1”);
Insert into authorities values(“test”, “ROLE_USER”);
Insert into authorities values(“test”, “ROLE_ADMIN”);
Insert into authorities values(“bob”, “ROLE_USER”);
```

Custom UserDetailService

```
@Service
public class CustomUserDetailService implements UserDetailsService {
    @Autowired
    private UserDao userDao;

    @Override
    public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
        final cs544.domain.User customer = userDao.findByEmail(email);
        if (customer == null) {
            throw new UsernameNotFoundException(email);
        }
        UserDetails user = User.withUsername(customer.getEmail())
                               .password(customer.getPassword())
                               .authorities("USER").build();
        return user;
    }
}
```

DaoAuthenticationProvider For Custom Authentication

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig {
    @Autowired
    private CustomUserDetailService myUserDetailsService; // see next slide

    @Bean
    public DaoAuthenticationProvider authProvider() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(this.myUserDetailsService);
        authProvider.setPasswordEncoder(passwordEncoder());
        return authProvider;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
```

Multiple Authentication Providers

- Spring will try each one, in the order found

```
<sec:authentication-manager>

    <sec:authentication-provider>
        <sec:user-service>
            <sec:user name="test" password="{noop}123" authorities="ROLE_USER, ROLE_ADMIN" />
            <sec:user name="bob" password="{noop}bobiscool" authorities="ROLE_USER" />
        </sec:user-service>
    </sec:authentication-provider>

    <sec:authentication-provider>
        <sec:jdbc-user-service data-source-ref="dataSource" />
    </sec:authentication-provider>

    <sec:authentication-provider ref="customAuthenticationProvider" />
</sec:authentication-manager>
```

I have not tested this yet for Java Config


MAHARISHI
HIGHER EDUCATION
FOR HIGHER CONSCIOUSNESS
INTERNATIONAL UNIVERSITY