Describe how security is implemented in OpenHDS:

In the OpenHDS , security is implemented using various features of spring security and aspect oriented programming. The OpenHDS secures the web request by restricting the user at the URL and service method level. It uses the spring security interceptor–url element to authenticate the users and grant access to the particular resource.

The spring login and logout handlers and interception rules are configured using the spring's security namespaces, tags and java annotations. Some of the security features of OpenHDS application are as follows:

Authentication:

In the OpenHDS , security starts with the authentication. A user with valid username and password can only login to the OpenHDS. The authentication is performed using the custom handler class extended from spring web security's SavedRequestAwareAuthenticationSuccessHandler. This authentication handler records the sessionID for the user. SessionID can be used to authorize the access to the subsequent requests. The *authentication flag is* set to true to indicate successful authentication. This process is useful since users have different privileges to access the resources/services.

The logout handler class clears the saved session information once the user logout the session.

Custom Filters:

OpenHDS takes the advantage of the spring web framework's DelegatingFilterProxy filter which automatically creates the filter bean SpringSecurityFilterChain and invokes chain of the required custom filters before processing the request.

Use of Interceptor:

Whitelist Authentication Interceptor authenticates the incoming Host IP addresses. It allows the registered host to access/download the data from the OpenHDS server. This handler class is extended from the spring web servlet's HandlerInterceptorAdapter. The access to certain URLs is denied if the unauthorized IP address is detected.

Use of Authorization advice using the Spring AOP:

OpenHDS uses the spring aop advices at service method calls. AOP advices are applied using BeforeAvice method and annotating the service methods with @Authorized. The AuthorizationAdvice class guarantees that the current user has the privileges to access the method. These privileges are predefined for the user. For example Admin is authorized to create, delete and view particular entity in the OpenHDS system. Whereas the data clerk can only view the entities.

Currently OpenHDS takes the advantages of various spring security features such as <global-method-security> elements, Authentication, Password encoding, spring security filter chain etc. Some of new features of Spring Security overlap with the exiting features. There are certain new

features such as AccessDeniedHandler which we can implement in the OpenHDS. We can design the custom handler error page to prevent unauthorized user accessing the sensitive information page or some functional module such as report, special study etc. This accessDenied page will appear when an unauthorized user with no privilege tries to access certain page.

The other feature which we can implement in OpenHDS is to add Remember-me authentication. Using the spring security remember-me element, the user will be identified between the sessions. Spring sends a cookie to the browser. The future sessions are authenticated automatically using the cookie. Nowadays, there are many web applications which takes advantage of spring Security's Remember-me element.

Currently, user session is not automatically logged out after certain time. It could lead to the security issue if the user forgets to logout the current session. We can configure spring to automatically detect the session timeout. It automatically log out the current session when the user forgets to logout the session. In this case, we can configure the invalid-session-url using spring security's session-management element if someone tries to access expired session. Alternatively we can store the session information and redirect user to login page.