Odpowiedzi na pytania:
P: W ktorym miejscu pobrałbyś prawdziwe ROLE dla usera? Teraz wpisales do tokenu claim recznie ale w realnym systemie pobrałbys to np. z bazy.
O: Do **AuthenticationSuccessHandler** jako patametr do metody onSuccess przychodzi między innymi uwierzytelnione **Authentication**. Wszystkie role i uprawnienia dostępne dla danego użytkownika powinny być dostępne jako kolekcja **GrantedAuthority**, którą mogę wyciągnąć z **Authentication** metodą **getGrantedAuthentication()**. W prawdziwym systemie zamiast wpisywać rolę ręcznie posłużyłbym się właśnie tą metodą.

P: Czy secret może być w jakiś sposób zmienny aby uniknąć jego wykradnięcia?
O: Tak, w systemach produkcyjnych sekret może być dostarczony do aplikacji na różne sposoby, np. poprzez plik application.properties. Dobrą praktyką jest, żeby takie sekrety regularnie zmieniać.

P: Czy będą jakieś automatyczne authentication providery dodane przez springa?
O: Tak, Spring dodaje **AuthenticationProvider**, zależnie od funkcjonalności, które mu skonfigurujemy. Jeżeli np. skonfigurujemy tzw. remember me poprzez użycie **http.rememberMe()** do listy providerów zostanie automatycznie dodany **RememberMeAuthenticationProvider.**

P: W tokenie możemy dodać czas życia tokenu - expirationDate. To oznacza że u`ytkownik może uwierzytelnić się kilkoma tokenami, które sobie wygenerował dopóki te tokeny nie stracą ważności - czy to jest ogolnie przyjete i uznawane za bezpieczne?
O: Jest to jeden z głównych problemów na który ludzie zajmujący się bezpieczeństwem zwracają uwagę. To co robią niektóre systemy to do JWT dokładają kolejny claim pochodzący ze standardu: **jti** - który przechowuje unikalny identyfikator danego tokena i utrzymują mapę pomiędzy zalogowanymi aktualnie użytkownikami i id tokena. Wówczas oprócz weryfikacji sygnatury dochodzi jeszcze jedno sprawdzenie -> czy id tokena się zgadza, jeżeli nie to taki request jest odrzucany. W momencie gdy użytkownik próbuje zalogować się kilkukrotnie stare id zastępowane jest nowym.