Rozdział 9
Security Patterns

Artur Michura
Krzysztof Michalski
Piotr Sularz
Iwona Poręba
Konrad Konicki
Tomasz Mysłajek
Jan Wiśniewski

Zagadnienia

- Wyjaśnienie pojęcia wzorców bezpieczeństwa
- Wyjaśnienie pojęcia wzorca single-sign-on
- Implementacja wzorca single-sign-on
- Wyjaśnienie mechanizmu uwierzytelniania
- Implementacja mechanizmu uwierzytelniania
- Wyjaśnienie authentication interceptor

Cel

Po zapoznaniu się z tym rozdziałem będziemy mogli stworzyć aplikację zabezpieczającą i zaimplementować ją za pomocą Java EE 8.

Security Patterns

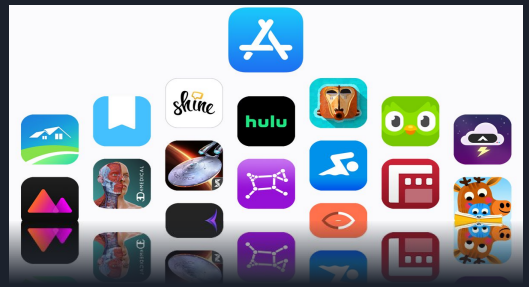
Security Patterns to zestaw rozwiązań typowych problemów związanych z bezpieczeństwem, które pojawiają się wielokrotnie. Duża część tych wzorców bezpieczeństwa służy do rozwiązywania problemów z uwierzytelnianiem, które jest związane z zasadami poufności i integralności. Dzięki wzorcom zabezpieczeń programista może tworzyć oprogramowanie o wysokim poziomie zabezpieczeń, które umożliwia rozwiązywanie znanych problemów przy użyciu przetestowanych i zweryfikowanych rozwiązań.

Dlaczego warto?

Aplikacje mają ścisły związek z danymi i ich przechowywaniem. To dlatego, że aplikacje zasadniczo polegają na zarządzaniu danymi w celu umożliwienia optymalizacji biznesu przy użyciu zadań automatyzacji, pomocy w podejmowaniu decyzji, organizowaniu zadań.

Wiele firm musi przechowywać wrażliwe dane i weryfikować dostęp do nich.

Zapotrzebowanie na oprogramowanie zabezpieczające znacznie wzrosło, a wiele firmy coraz więcej inwestuje w tworzenie bezpiecznych aplikacji.



Zasady bezpieczeństwa

Integralnym elementem bezpieczeństwa jest bezpieczeństwo informacji, które musi być zgodne podstawowymi zasadami:

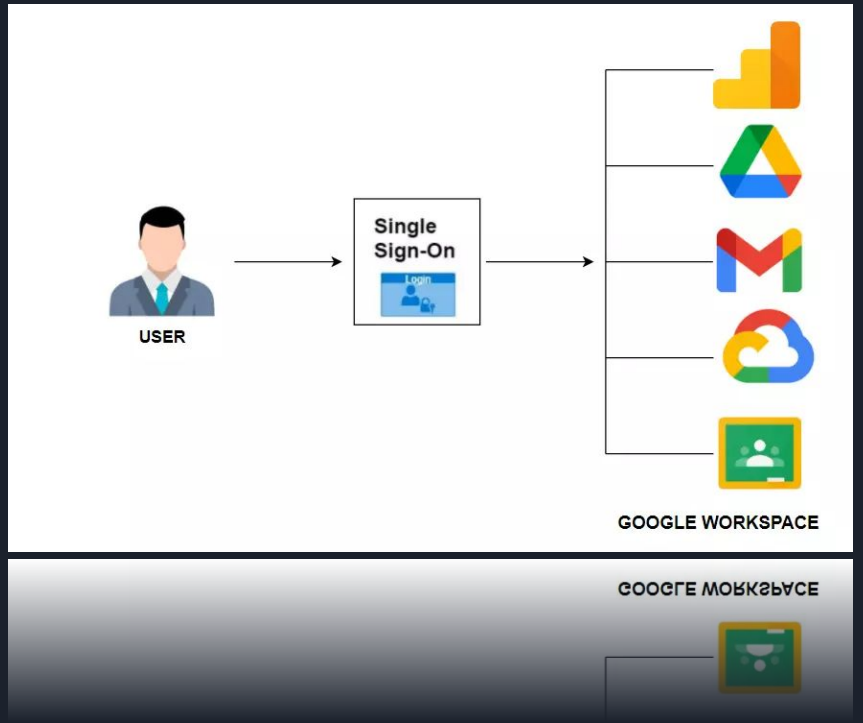
- Poufność: dane nie powinny być dostępne dla nieuprawnionych użytkowników ani dla żadnych nieuprawnionych obiektów żądających dostępu do danych.
- Integralność: dane nie mogą być aktualizowane ani modyfikowane w nieautoryzowany sposób.
- Dostępność: dane powinny być dostępne, gdy jest to potrzebne.
- Niezaprzeczalność: użytkownicy nie mogą odrzucać ani zaprzeczyć relacji przy użyciu danych ani żadnych innych procesów - brak możliwości wyparcia się swego uczestnictwa w całości lub w części wymiany danych przez jeden z podmiotów uczestniczących w tej wymianie

Zasady bezpieczeństwa

Aby aplikacja była bezpieczna musi zapewnić co najmniej 4 podstawowe zasady przedstawione na poprzednim slajdzie.

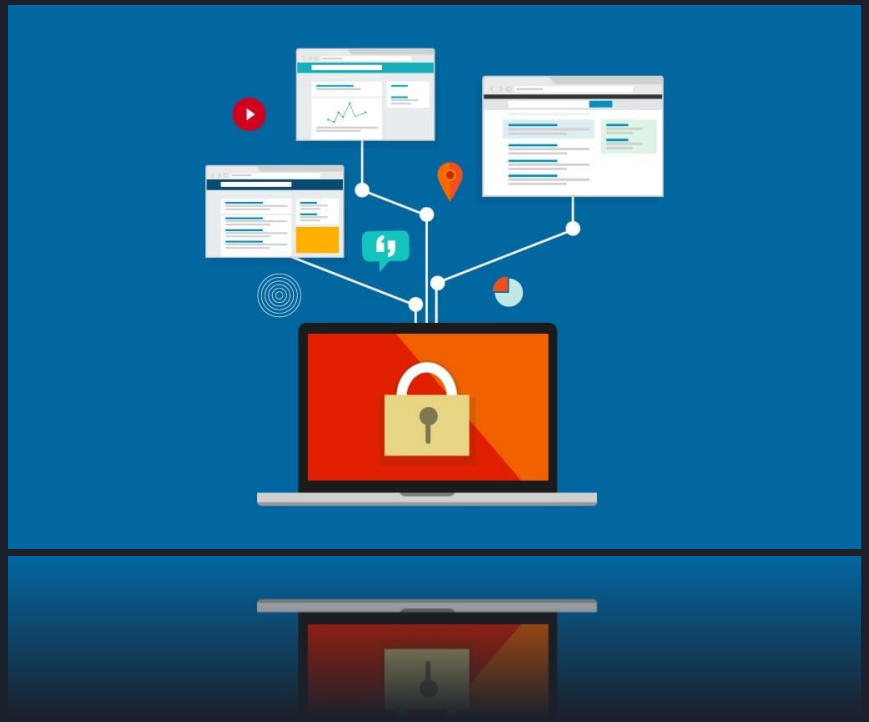
Wzorzec single-sign-on

W środowiskach biznesowych powszechne jest łączenie usług od danego dostawcy w taki sposób, że logując się w jednej z nich uzyskujemy dostęp do wszystkich usług. Przykładem jest logowanie do aplikacji firmy Google.



Wzorzec single-sign-on

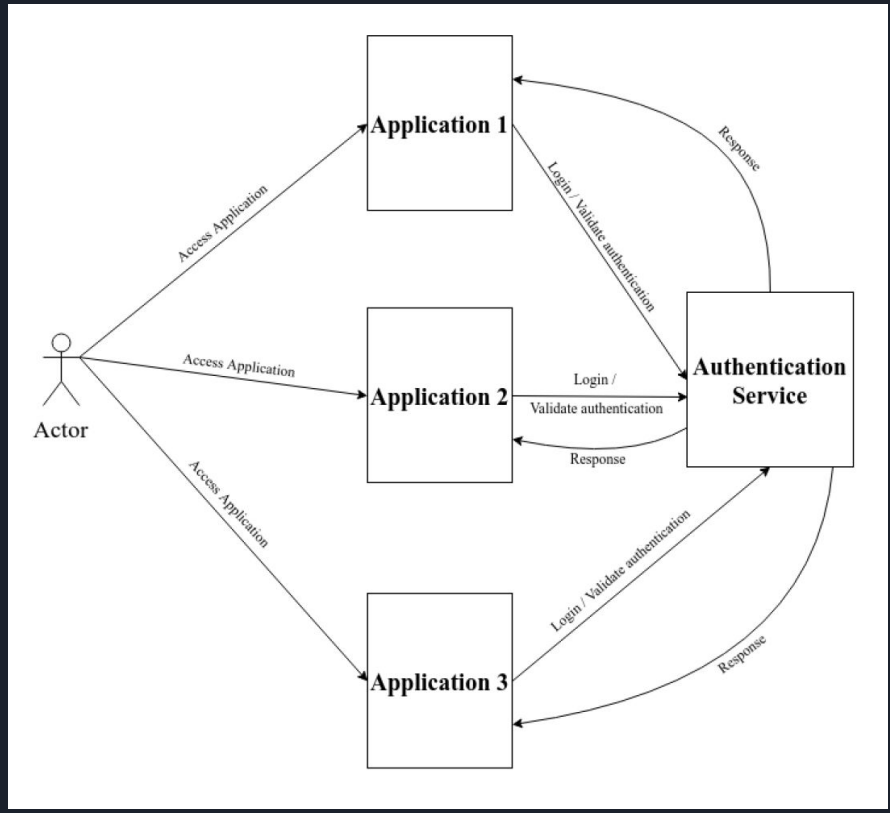
Wzorzec single-sign-on wykorzystuje centralną usługę uwierzytelniającą, która jest współdzielona pomiędzy kilka aplikacji. Użytkownik logując się raz uzyskuje zatem dostęp do wszystkich aplikacji korzystających z tej usługi.



Wzorzec single-sign-on

Przykład po prawej pokazuje proces logowania:

1. Użytkownik wysyła zapytanie do aplikacji.
2. Aplikacja wysyła zapytanie do usługi uwierzytelniającej.
3. Usługa w odpowiedzi zwraca informację, czy użytkownik jest już zalogowany.
    a. Jeśli nie jest, aplikacja przekierowuje go na stronę z logowaniem.
    b. Jeśli jest użytkownik otrzymuje dostęp do aplikacji.



Wzorzec single-sign-off

Wzorzec single-sign-off pozwala wylogować się jednocześnie z wielu aplikacji korzystających ze wspólnego serwera uwierzytelniającego. Jest to dobra praktyka, ponieważ pozwala oddzielić logikę odpowiadającą za autoryzację dostępu do aplikacji od logiki biznesowej.

Mechanizmy uwierzytelniania

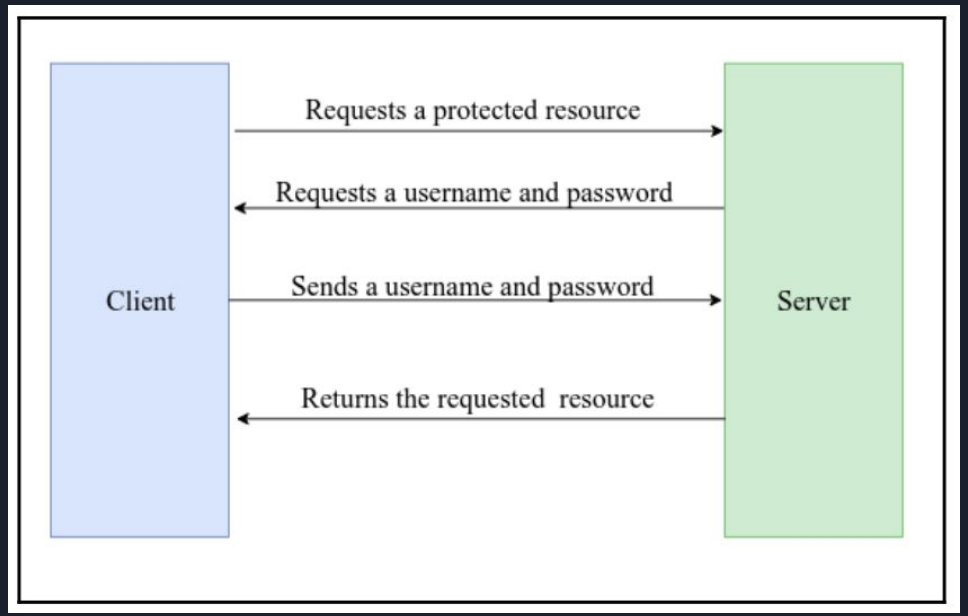
Proces uwierzytelniania polega na zweryfikowaniu zadeklarowanej przez użytkownika tożsamości. Przykładowo kiedy użytkownik żąda chronionych zasobów, musi być wpierw uwierzytelniony, aby jego prawa dostępu zostały zweryfikowane przez serwer.

Mechanizmy uwierzytelniania w Java EE 8:

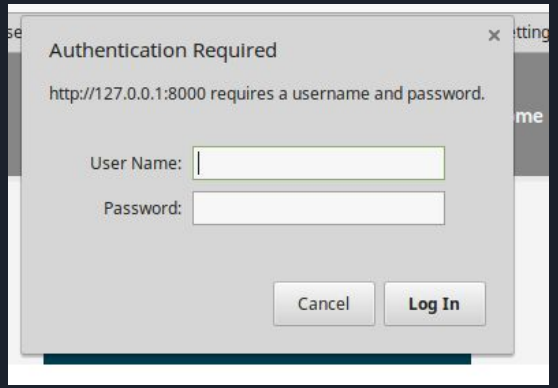
- podstawowe uwierzytelnianie (basic authentication)
- uwierzytelnianie oparte o formularze (form-based authentication)
- uwierzytelnianie oparte o funkcję skrótu (digest authentication)
- uwierzytelnianie klienta (client authentication)
- uwierzytelnianie wzajemne (mutual authentication)

https://securityintelligence.com/why-you-need-advanced-authentication-to-protect-identities-without-compromising-user-experience/

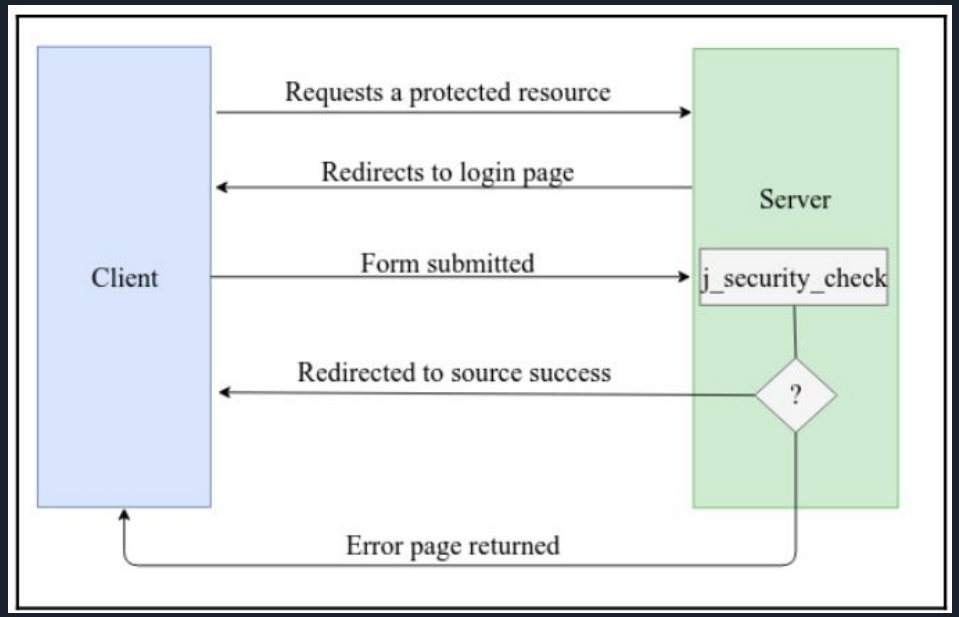
Podstawowe uwierzytelnianie



- domyślny mechanizm
- kiedy nieuwierzytelniony użytkownik prosi o dostęp do zasobów wyświetla mu się okno dialogowe żądające nazwę oraz hasło użytkownika
- ataki MITM (konieczność używania protokołów SSL, VPN)

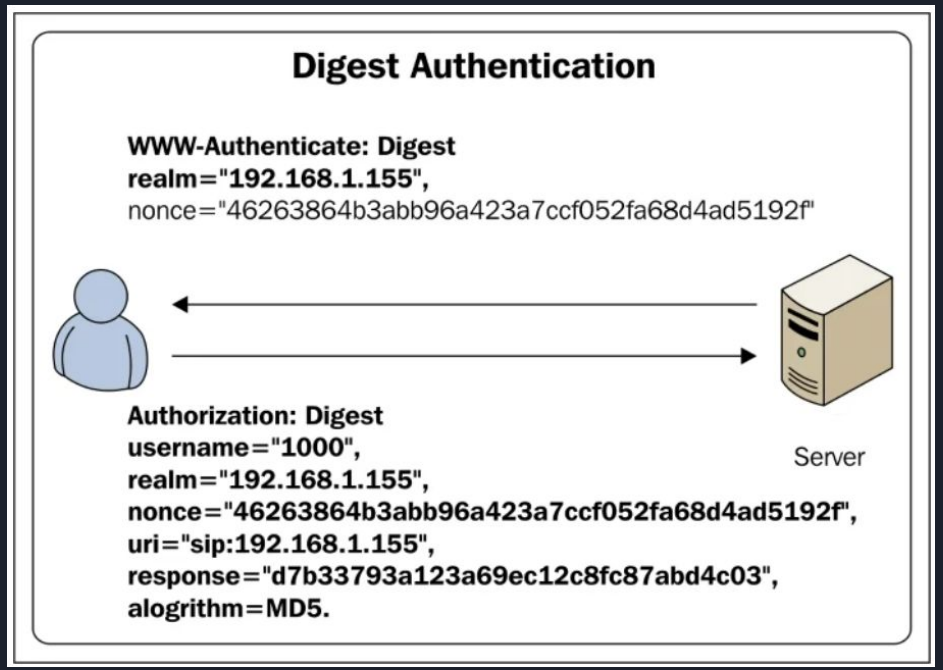


Uwierzytelnianie oparte o formularze



- korzysta z formularzy zawierających pola takie jak nazwa użytkownika czy hasło
- umożliwia deweloperowi dostosowanie odpowiedniej strony logowania oraz strony błędu (błędnej próby logowania)
- serwer sprawdza poprawność przesyłanych danych i odsyła odpowiednia stronę
- ataki MITM (konieczność używania protokołów SSL, VPN)

Uwierzytelnianie oparte o funkcję skrótu

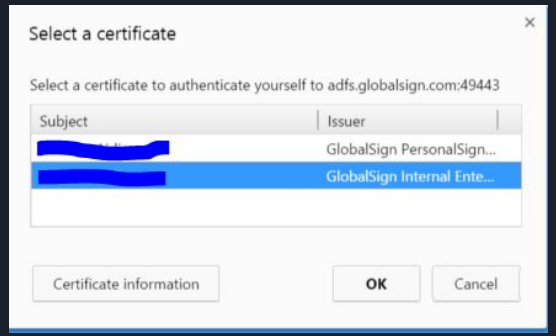


- przy przesyłaniu wrażliwych danych (np. hasła) korzysta z jednokierunkowej funkcji skrótu (haszującej)
- nie ma potrzeby korzystania z mechanizmów typu SSL czy VPN

Uwierzytelnianie klienta

https://comodosslstore.com/blog/wp-content/uploads/2018/06/ssl-tls-client-authentication.jpg

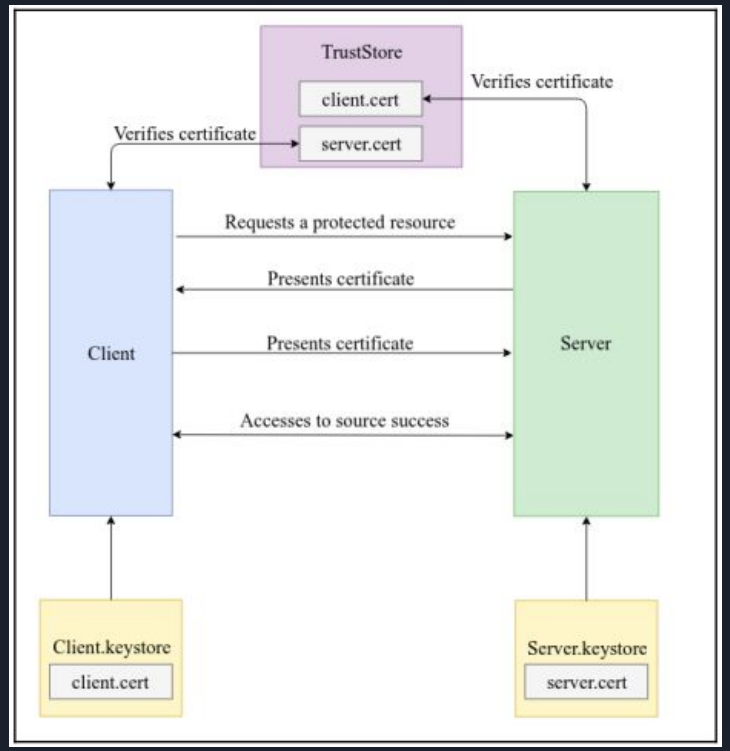
- Uwierzytelnienie klienta odbywa się przy pomocy jego certyfikatu.
- Wykorzystuje HTTP over SSL (HTTPS)
- Wymaga przechowywania certyfikatu klienta w przeglądarce/aplikacji



https://www.globalsign.com/application/files/4014/5407/5094/client_authentication_3.png

Wzajemne uwierzytelnianie

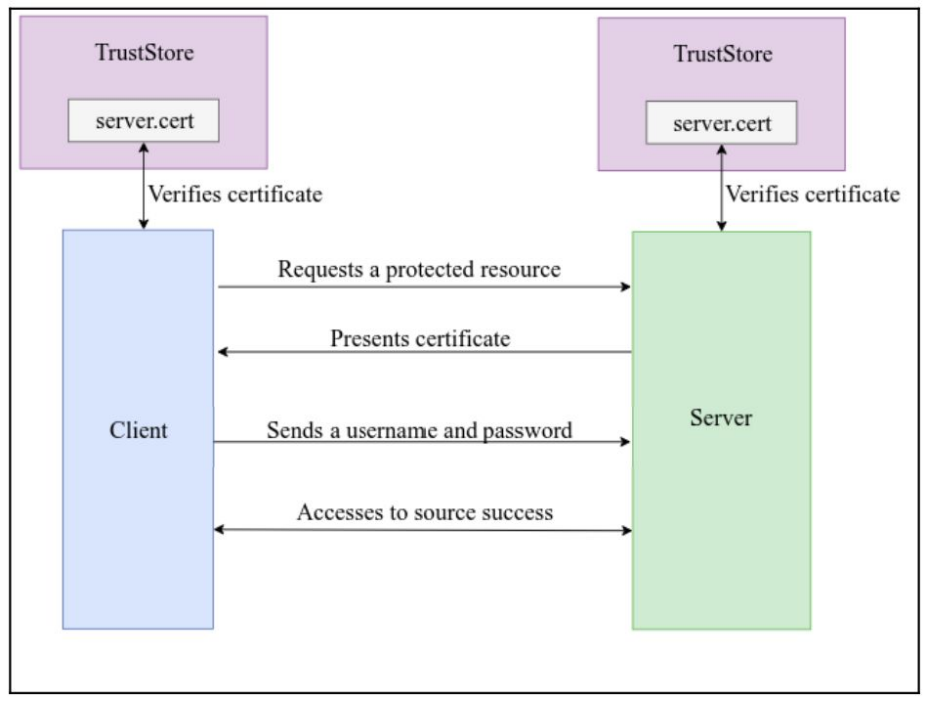
Certificate Based Mutual Authentication



- serwer uwierzytelnia klienta, a klient serwer
- połączenie następuje wyłącznie jeśli klient i serwer wymienią się, sprawdzą oraz będą ufać danym certyfikatom
- Zapobiega atakom np. MIM (Man in The Middle) czy spoofingu
- oparte na certyfikatach, albo loginie i haśle

Wzajemne uwierzytelnianie

Username/password Based Mutual Authentication



- serwer uwierzytelnia klienta, a klient serwer
- połączenie następuje wyłącznie jeśli klient i serwer wymienią się, sprawdzą oraz będą ufać danym certyfikatom
- Zapobiega atakom np. MIM (Man in The Middle) czy spoofingu
- oparte na certyfikatach, albo loginie i haśle

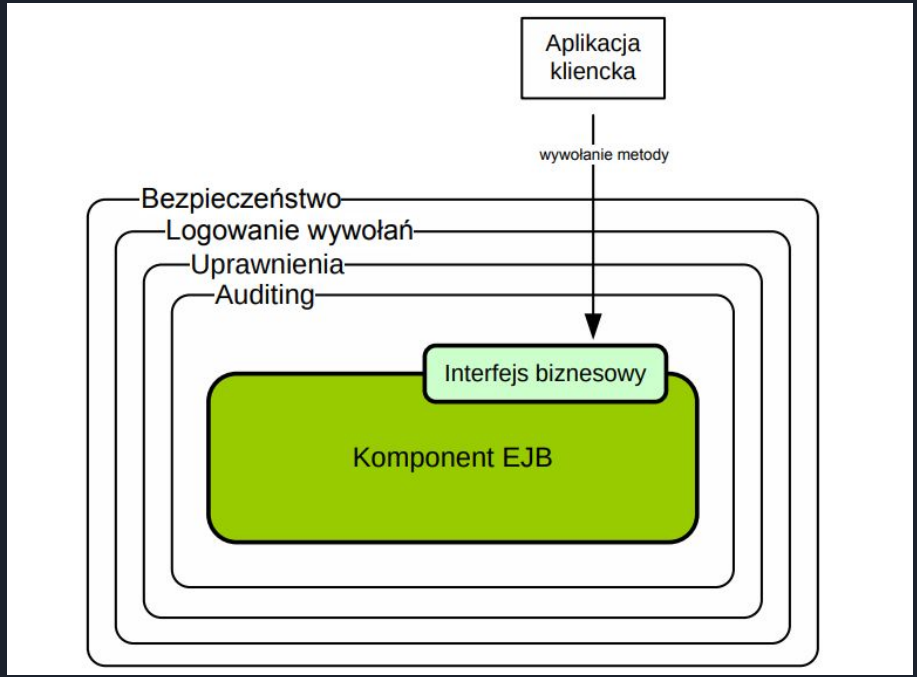
Interceptory

Wstrzykiwanie dodatkowych funkcjonalności do metody bez ingerencji w nią samą

Interceptory umożliwiają opakowanie wywołania metody dodatkową logiką

Interceptury działają opierając się na paradygmacie programowania AOP

Chcąc zaimplementować interceptor w Java EE8 możemy użyć interceptora EJB lub CDI



https://www.michalbartyzel.pl/wp-content/uploads/2020/04/04.-Interceptoryskrypt.pdf

Podsumowanie

Omówiono koncept Security Patterns

Przedstawiono ideę rozwiązania Single-Sign-On

Omówiono mechanizmy uwierzytelniania we współczesnych aplikacjach

Wyjaśniono pojęcie interceptorów

Jesteśmy w stanie stworzyć bezpieczniejszą aplikację!

Dziękujemy za uwagę