EQUILIBRIUM

JDD 2006 Java Developers Day

Bezpieczeństwo aplikacji webowych na przykładzie Acegi Security for Spring

Grzegorz Rdzany

Kraków-PROIDEA, 21.10.2006 r.

EQUILIBRIUM

Plan prelekcji

- Bezpieczeństwo aplikacji webowych – zagrożenia i rozwiązania
- Acegi Security jako framework realizujący warstwę bezpieczeństwa aplikacji
- Budowa Acegi Security
- Podstawowa konfiguracja Acegi Security
- Wady i zalety Acegi Security for Spring

Podział zagrożeń

- **Warstwa łącza sieciowego**
  - Możliwość podsłuchu
- **Serwer**
  - Luki w bezpieczeństwie oprogramowania serwera
  - Błędna konfiguracja oprogramowania serwera
  - Niewłaściwe rozwiązania w kodzie stron WWW
  - Niewłaściwe rozwiązania w logice biznesowej aplikacji webowej
- **Klient**
  - Szerokie spektrum zagrożeń, zależnych od wykorzystywanej technologii (szkodliwe skrypty, ActiveX, applety, cookies itd.)
- **Czynnik ludzki**

Zagrożenia dla aplikacji webowych

- Nieuprawniony dostęp do chronionych danych
- Przejęcie profilu użytkownika
- Ujawnianie szczegółów implementacyjnych
- Manipulowanie parametrami tak, aby uzyskać niedozwolone w normalnych okolicznościach efekty
- Zaburzanie pracy aplikacji

Zabezpieczenia – od czego zacząć?

- 100% bezpieczeństwa = brak dostępu do sieci
- **Dokładne przemyślenie potencjalnych zagrożeń**
- Podział systemu bezpieczeństwa na warstwy
- Zapewnienie każdej warstwie najściślejszej ochrony w ramach wyznaczonej dla niej odpowiedzialności

Warstwy bezpieczeństwa

- Transmisja danych – szyfrowanie, sprawdzanie integralności pakietów
- Kontrola połączeń – firewall, ograniczanie dostępu z nieautoryzowanych klientów
- System operacyjny serwera – właściwa konfiguracja, aktualizacje
- Zabezpieczenie przed klasycznymi atakami – DoS (DDoS), brute force itd.
- Właściwa konfiguracja JRE
- Zabezpieczenia aplikacji webowej – np. Acegi Security

Koncepcja bezpieczeństwa aplikacji webowych

- Wyodrębnienie zasobów chronionych i publicznych
- Wyodrębnienie grupy użytkowników mających prawa dostępu do zasobów chronionych
- Realizacja mechanizmu uwierzytelniania (authentication)
- Realizacja mechanizmu autoryzacji (authorization)

Uwierzytelnianie

- Opiera się na weryfikacji informacji uwierzytelniających (credentials)
  - Login, hasło
  - „Bilet” (ticket)
  - Hasła jednorazowe
  - Wzór głosu
  - Wzór tęczówki oka
  - Wszystko inne co podpowie nam wyobraźnia...

Autoryzacja

- Ocena, czy użytkownik (zdalny system) ma prawo do korzystania z żądanych zasobów
- Kryteria oceny
  - Role
  - Ramy czasowe
  - Zakresy adresów IP
  - Identyfikatory użytkowników
  - Wszystko inne co podpowie nam wyobraźnia...

EQUILIBRIUM

Acegi Security for Spring - dlaczego?

- Realizuje powyżej przedstawioną koncepcję bezpieczeństwa aplikacji webowych
- Open Source
- Elastyczność
- Uniwersalność
- Przezroczystość
- Doskonała współpraca z innymi technologiami
- Z powodzeniem wdrożony w wielu aplikacjach

Acegi Security for Spring - cechy

- Oferuje szeroki zakres usług związanych z bezpieczeństwem
- Nie wymaga ingerencji w logikę biznesową
- Stworzony z myślą o projektach opartych o Spring Framework
- Może działać bez Springa (nie zalecane)
- Wysoce konfigurowalny
- Dostarcza bazę dla rozwoju własnych koncepcji bezpieczeństwa
- Współpracuje z popularnymi rozwiązaniami stosowanymi przez developerów – Webwork, Struts, Velocity, SpringMVC, JSF i wiele innych

Acegi Security for Spring - budowa

- Oparty na kilku centralnych interfejsach
  - Acegi Security oferuje kilka implementacji tych interfejsów, realizujących podstawowe wymagania
  - W celu realizacji bardziej złożonych wymogów można napisać własne implementacje
- Kluczowym obiektem jest *SecurityContextHolder* – przechowuje dane związane z bezpieczeństwem, istotne z punktu widzenia działania Acegi Security
- *SecurityContextHolder* zawiera w sobie *SecurityContext*, który z kolei przechowuje obiekt typu *Authentication* – reprezentację danych zalogowanego użytkownika, związanych z systemem bezpieczeństwa

EQUILIBRIUM

Acegi Security for Spring - budowa

- *HttpSessionContextIntegrationFilter* – dba o to, by *SecurityContextHolder* był przechowywany między kolejnymi żądaniami HTTP
- *AuthenticationEntryPoint* - „brama” do aplikacji webowej, punkt od którego rozpoczyna się proces uwierzytelniania
- *AuthenticationProvider* – jego zadaniem jest utworzenie właściwego obiektu *Authentication*, w przypadku podania przez użytkownika prawidłowych danych uwierzytelniających
- *ExceptionTranslationFilter* – wyłapuje i właściwie obsługuje wszelkie wyjątki rzucane przez obiekty Acegi Security

Acegi Security for Spring - budowa

- *AccessDecisionManager – zarządza podejmowaniem decyzji, czy zalogowany użytkownik ma prawa do rządanego zasobu*
- *AccessDecisionVoter* – obiekt wykorzystywany przez AccessDecisionManager do podejmowania decyzji autoryzacyjnych

EQUILIBRIUM

Acegi Security for Spring - konfiguracja

- Acegi Security opiera swe działanie na pewnej grupie filtrów, które dostarczają odpowiednich usług związanych z bezpieczeństwem
- Filtry należy zdefiniować w pliku *web.xml*
- Filtry muszą być zdefiniowane w określonej kolejności
- Istnieją dwa sposoby definiowania filtrów: za pomocą obiektu *FilterToBeanProxy* oraz za pomocą *FilterChainProxy*

Acegi Security for Spring - konfiguracja

- *FilterToBeanProxy*

```
<filter>
	<filter-name>Acegi HTTP Request Security Filter</filter-name>
	<filter-class>org.acegisecurity.util.FilterToBeanProxy</filter-class>
	<init-param>
		<param-name>targetClass</param-name>
		<param-value>
			org.acegisecurity.ClassThatImplementsFilter
		</param-value>

		<!-- zamiast targetClass można podać parametr targetBean, czyli
	nazwę beana zdefiniowanego w applicationContext.xml:
		<param-name>targetBean</param-name>
		<param-value>
			classThatImplementsFilter
		</param-value>
		-->

	</init-param>
</filter>
```

Acegi Security for Spring - konfiguracja

- *FilterChainProxy*

```
<bean id="filterChainProxy" class="org.acegisecurity.util.FilterChainProxy">
      <property name="filterInvocationDefinitionSource">
            <value>
                  CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
                  PATTERN_TYPE_APACHE_ANT
                  /webServices/**=httpSessionContextIntegrationFilter,
                  logoutFilter,basicAuthenticationProcessingFilter,
                  exceptionTranslationFilter,filterSecurityInterceptor
                  /**=httpSessionContextIntegrationFilter,logoutFilter,
                  anonymousProcessingFilter,formAuthenticationProcessingFilter,
                  exceptionTranslationFilter,filterSecurityInterceptor
            </value>
      </property>
</bean>
```

Acegi Security for Spring - konfiguracja

- *FilterChainProxy* c.d.

```
<filter>
	<filter-name>Acegi Filter Chain Proxy</filter-name>
	<filter-class>org.acegisecurity.util.FilterToBeanProxy</filter-class>
	<init-param>
		<param-name>targetBean</param-name>
		<param-value>filterChainProxy</param-value>
	</init-param>
</filter>
```

Acegi Security for Spring - konfiguracja

- Kolejność definiowania filtrów
  1. *ChanelProcessingFilter*
  2. *ConcurrentSessionFilter*
  3. *HttpContextIntegrationFilter*
  4. filtry odpowiedzialne za proces uwierzytelniania – *AuthenticationProcessingFilter, BasicProcessingFilter, HttpRequestIntegrationFilter, JbossIntegrationFilter itd.*
  5. *SecurityContextHolderAwareRequestFilter*
  6. *RememberMeProcessingFilter*
  7. *AnonymousProcessingFilter*
  8. *ExceptionTranslationFilter*
  9. *FilterSecurityInterceptor*

Acegi Security for Spring - konfiguracja

- Można tworzyć własne filtry
- Nie ma konieczności dodawania wszystkich filtrów – dodajemy tylko te, które są nam potrzebne
- Jedynym filtrem, który musi koniecznie się pojawić jest *HttpSessionContextIntegrationFilter*

```
<bean id="httpSessionContextIntegrationFilter"
      class="org.acegisecurity.context.HttpSessionContextIntegrationFilter">
      <property name="context">
            <value>org.acegisecurity.context.SecurityContextImpl</value>
      </property>
</bean>
```

EQUILIBRIUM

Acegi Security for Spring - uwierzytelnianie

- Acegi Security posiada własne mechanizmy uwierzytelniające
- Można napisać mechanizm uwierzytelniania dopasowany do naszych potrzeb
- Można połączyć z Acegi Security mechanizmy już istniejące, wymagane przez politykę firmy itd.

EQUILIBRIUM

Acegi Security for Spring - uwierzytelnianie

- Acegi Security wspiera uwierzytelnianie oparte na poniższych technologiach
  - HTTP BASIC authentication headers
  - HTTP Digest authentication headers
  - HTTP X.509 client certificate exchange
  - LDAP
  - Autentykacja oparta na formularzach
  - CAS
  - Automatyczne uwierzytelnianie „remember me”
  - Anonimowe uwierzytelnianie
  - JAAS
  - Wiele, wiele innych...

Acegi Security for Spring - uwierzytelnianie

- *AuthenticationManager* – zarządza procesem uwierzytelniania

```
<bean id="authenticationManager"
        class="org.acegisecurity.providers.ProviderManager">
        <property name="providers">
                <list>
                        <ref local="daoAuthenticationProvider"/>
                        <ref local="anonymousAuthenticationProvider"/>
                        <ref local="rememberMeAuthenticationProvider"/>
                </list>
        </property>
</bean>
```

Acegi Security for Spring - uwierzytelnianie

- *AuthenticationProvider* – realizuje uwierzytelnianie według określonych reguł, z wykorzystaniem odpowiednich danych

```
<bean id="daoAuthenticationProvider"
        class="org.acegisecurity.providers.dao.DaoAuthenticationProvider">
        <property name="userDetailsService">
                <ref bean="jdbcAuthenticationDao"/>
        </property>
        <property name="saltSource">
                <ref bean="saltSource"/>
        </property>
        <property name="passwordEncoder">
                <ref bean="passwordEncoder"/>
        </property>
</bean>
```

Acegi Security for Spring - uwierzytelnianie

- Kodowanie haseł

```
<bean id="passwordEncoder"
        class="org.acegisecurity.providers.encoding.ShaPasswordEncoder"/>

<bean id="saltSource"
        class="org.acegisecurity.providers.dao.salt.ReflectionSaltSource">
        <property name="userPropertyToUse" value="getUserSalt"></property>
</bean>
```

Acegi Security for Spring - uwierzytelnianie

- *UserDetailsService* – interfejs realizujący dostęp do danych uwierzytelniających

```
<bean id="jdbcAuthenticationDao"
        class="org.acegisecurity.userdetails.jdbc.JdbcDaoImpl">
        <property name="dataSource" ref="dataSource"/>
        <property name="usersByUsernameQuery">
                <value>
                        SELECT login,password,enabled FROM users
                        WHERE login = ?
                </value>
        </property>
        <property name="authoritiesByUsernameQuery">
                <value>
                        SELECT login,rolename FROM users usr
                        INNER JOIN user_roles roles
                        ON usr.id = roles.ref_user WHERE login = ?
                </value>
        </property>
</bean>
```

Acegi Security for Spring - uwierzytelnianie

- Uruchomienie uwierzytelniania w aplikacji wymaga dołączenia odpowiedniego filtru. W przypadku uwierzytelniania za pomocą formularza logowania jest to *AuthenticationProcessingFilter*

```
<bean id="formAuthenticationProcessingFilter"
        class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilter">
        <property name="authenticationManager">
                <ref bean="authenticationManager" />
        </property>
        <property name="authenticationFailureUrl">
                <value>/security/showLoginForm.do</value>
        </property>
        <property name="defaultTargetUrl">
                <value>/startPage.do</value>
        </property>
        <property name="filterProcessesUrl">
                <value>/security/j_acegi_security_check</value>
        </property>
        <property name="alwaysUseDefaultTargetUrl">
                <value>true</value>
        </property>
</bean>
```

Acegi Security for Spring - uwierzytelnianie

- Formularz logowania

```
<html>
        <body>
                <form action="j_acegi_security_check" method="post">
                        Login: <input type="text" name="j_username">
                        Password: <input type="text" name="j_password">
                        <input type="submit" value="Login">
                </form>
        </body>
</html>
```

Acegi Security for Spring - autoryzacja

- Podobnie jak w przypadku uwierzytelniania, aby uruchomić autoryzację w aplikacji trzeba zdefiniować odpowiedni filtr – *FilterSecurityInterceptor*

Acegi Security for Spring - autoryzacja

```
<bean id="filterSecurityInterceptor"
        class="org.acegisecurity.intercept.web.FilterSecurityInterceptor">
        <property name="authenticationManager">
                <ref bean="authenticationManager"/>
        </property>
        <property name="accessDecisionManager">
                <ref bean="accessDecisionManager"/>
        </property>
        <property name="objectDefinitionSource">
                <value>
                        CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
                        PATTERN_TYPE_APACHE_ANT

                        /security/logout.do=ROLE_ANONYMOUS
                        /security/showLoginForm.do=ROLE_ANONYMOUS,ROLE_USER
                        /security/accessDenied.do=ROLE_ANONYMOUS,ROLE_USER
                        /startPage.do=ROLE_USER
                        /somePage.do=ROLE_USER
                        /usersArea/createAccount.do=ROLE_ANONYMOUS
                        /usersArea/**=ROLE_USER
                        <!-- cała reszta tylko dla administratora -->
                        /**=ROLE_ADMIN
                </value>
        </property>
</bean>
```

Acegi Security for Spring - uwierzytelnianie

- *AccessDecisionManager* – obiekt zarządzający dostępem do poszczególnych zasobów

```
<bean id="accessDecisionManager"
        class="org.acegisecu rity.vote.AffirmativeBased">
        <property name="decisionVoters">
                <list>
                        <ref bean="roleVoter" />
                </list>
        </property>
</bean>
```

Acegi Security for Spring - uwierzytelnianie

- *AccessDecisionVoter* – obiekt „głosujący” czy należy udzielić dostępu do danego zasobu
- *AccessDecisionManager* może zawierać kilka takich obiektów (odpytywane są one po kolei)
- Można stworzyć własne „votery” podejmujące decyzje na podstawie dowolnych reguł
- „Voter” może wstrzymać się od głosu np. gdy podjęcie decyzji nie leży w jego kompetencjach

```
<bean id="roleVoter" class="org.acegisecurity.vote.RoleVoter"/>
```

EQUILIBRIUM

Acegi Security for Spring - uwierzytelnianie

- Implementacje *AccessDecisionManagerów*
  - *AffirmativeBased* – udziela dostępu jeśli choć jeden z *AccessDecisionVoter*ów przyzna dostęp
  - *ConsensusBased* – działa według demokratycznych zasad
  - *UnanimousBased* – udziela dostępu jeśli żaden z „voterów” nie odmówi dostępu

Acegi Security for Spring – obsługa wyjątków

- *ExceptionTranslationFilter* – obiekt przechwytujący wyjątki Acegi Security i obsługujący je

```
<bean id="exceptionTranslationFilter" class="org.acegisecurity.ui.ExceptionTranslationFilter">
        <property name="authenticatio nEntryPo int">
                <ref bean="authenticationEntryPo int"/>
        </property>
        <property name="accessDeniedHandler">
                <bean class="org.acegisecurity.ui.AccessDeniedHandlerImpl">
                        <property name="errorPage" value="/security/accessDenied.do"  />
                </bean>
        </property>
</bean>
```

Acegi Security for Spring – obsługa wyjątków

- *AuthenticationEntryPoint* – punkt od którego rozpoczyna się proces uwierzytelniania

```
<bean id="authenticationEntryPoint"
        class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilterEntryPoint">
        <property name="loginFormUrl">
                <value>/security/showLoginEntryForm.do</value>
        </property>
        <property name="forceHttps">
                <value>false</value>
        </property>
</bean>
```

EQUILIBRIUM

Acegi Security for Spring - podsumowanie

- **Zalety**
  - Elastyczność
  - Przezroczystość
  - Przenośność
  - Wysoka konfigurowalność
  - Nieograniczone możliwości rozbudowy
  - Umiejętność współpracy z wiodącymi rozwiązaniami w zakresie aplikacji webowych
  - Open Source
  - Bardzo dobra dokumentacja
  - Wypróbowany w wielu aplikacjach

EQUILIBRIUM

Acegi Security for Spring - podsumowanie

- **Wady**
  - Niezbyt łatwa początkowa konfiguracja
  - Open Source

Dziękujemy za uwagę!