Java Work Group

June 18th 2009

Spring Security

Application Security

- Security is a crucial *aspect* of most applications
- Security is a concern that transcends an application’s functionality
- An application should play no part in securing itself
- It is better to keep security concerns separate from application concerns

Acegi Security

- Started in 2003
- Became extremely popular
- Security Services for the Spring Framework
- From version 1.1.0, Acegi becomes a Spring Module

Spring Security

“Spring Security is a powerful, flexible security solution for enterprise software, with a particular emphasis on applications that use Spring.”

Spring Security

- provides declarative security for your Spring-based applications
- handles authentication and authorization
- takes full advantage of dependency injection (DI) and aspect-oriented techniques based on the Spring Framework

What it is not ?

- Firewall, proxy server, IDS (Intrusion Detection System)
- Operating system security
- JVM (sandbox) security

- ***Developers are trusted to use it properly***

Who uses it?

- Over 231,000 downloads on SourceForge
- At least 20,000 downloads per release
- Over 14,000 posts in the community forum
- Used in many demanding environments
  - Large banks and business
  - Defence and government
  - Universities
  - Independent software vendors
  - OSS like OpenNMS, OAJ, Roller, AtLeap,....

It plays nicely with others...

- Spring Portfolio
- AspectJ
- JA-SIG CAS
- JOSSO
- NTLM via JCIFS
- OpenID
- SiteMinder
- Atlassian Crowd
- jCaptcha
- JAAS
- Jasypt
- Grails
- Mule
- DWR
- Appfuse
- AndroMDA

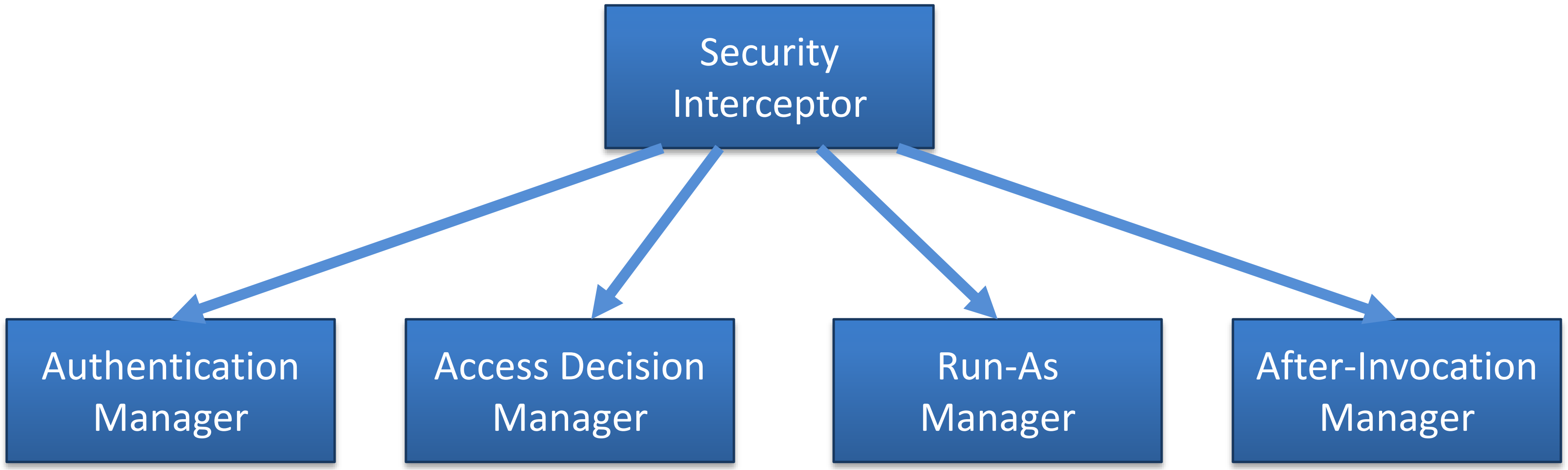
Major capability areas

- ***Authentication***
- ***Web URL authorization***
- ***Method invocation authorization***
- Domain instance based security (ACLs)
- WS-Security (via Spring Web Services)
- Flow Authorization (via Spring Web Flow)
- Human user detection (Captcha)

Key concepts

- Filters (Security Interceptor)
- Authentication
- Authorization
  - *Web authorization*
  - *Method authorization*

Fundamental elements of Spring Security

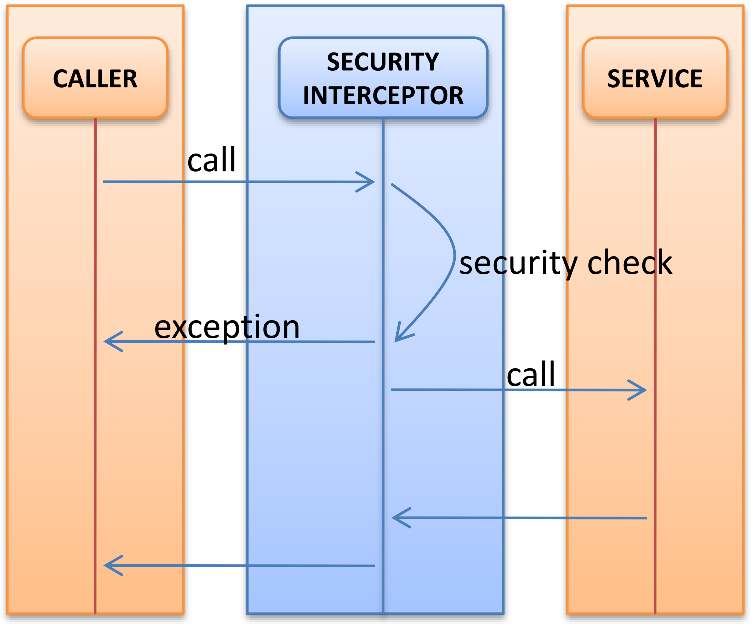


Spring Security

Filters

Security Interceptor

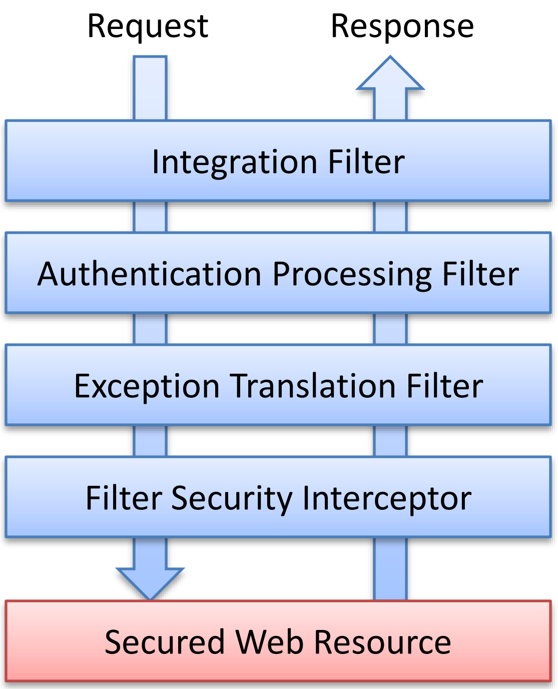
- a latch that protects secured resources, to get past users typically enter a username and password



- implementation depends on resource being secured
  - URLs - Servlet Filter
  - Methods - Aspects

- delegates the responsibilities to the various managers

Spring Security Filters

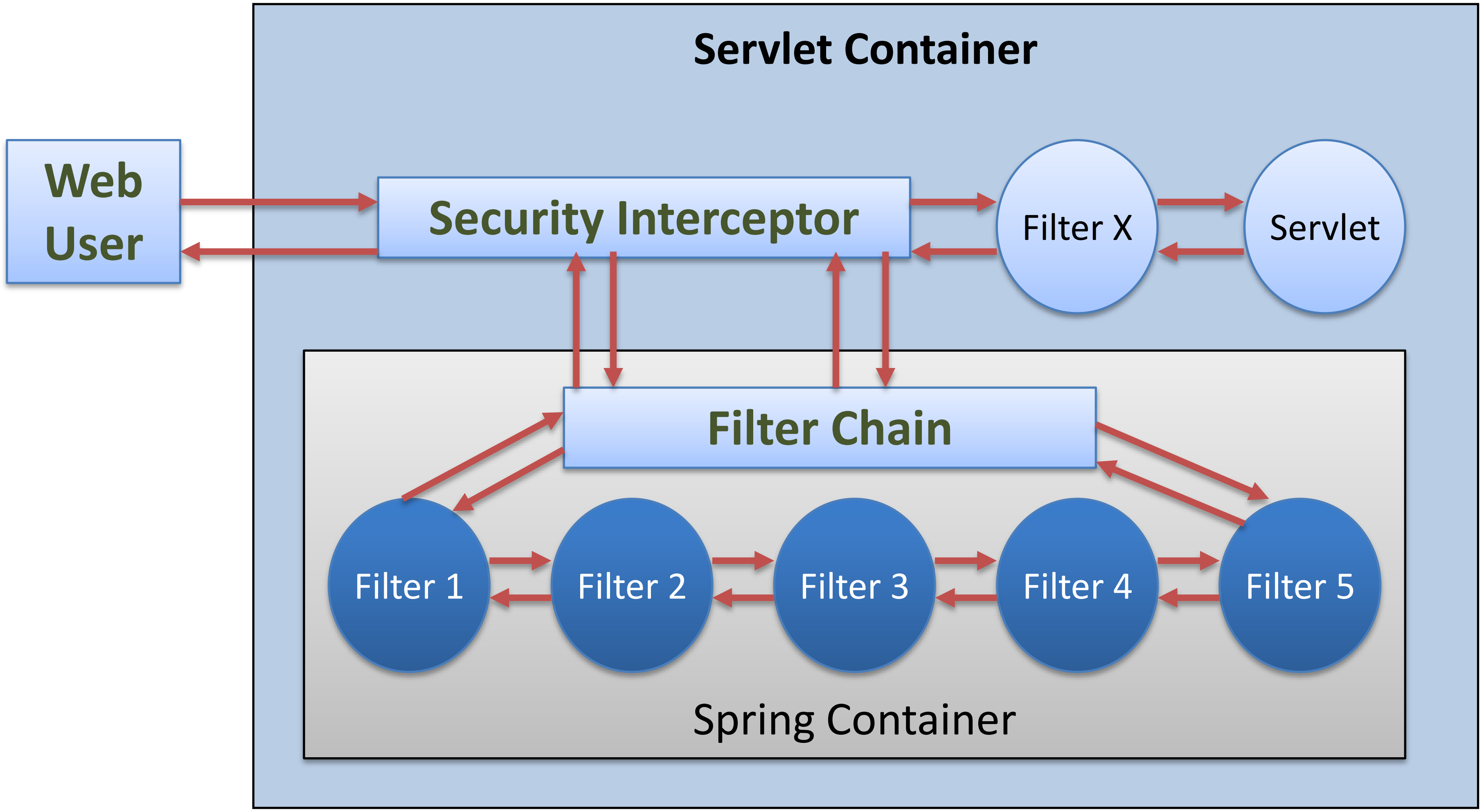

Request
Response
Integration Filter
Authentication Processing Filter
Exception Translation Filter
Filter Security Interceptor
Secured Web Resource

Filter	What it does
Integration Filter	responsible for retrieving a previously stored authentication (most likely stored in the HTTP session) so that it will be ready for Spring Security's other filters to Process
Authentication Processing Filter	determine if the request is an authentication request. If so, the user information (typically a username/ password pair) is retrieved from the request and passed on to the authentication manager
Exception Translation Filter	translates exceptions, for AuthenticationException request will be sent to a login screen, for AccessDeniedException returns HTTP 403 to the browser
Filter Security Interceptor	examine the request and determine whether the user has the necessary privileges to access the secured resource. It leans heavily on the authentication manager and the access decision manager

Spring Security Filters

Filter	Purpose
HttpRequestIntegrationFilter	Populates the security context using information from the user principal
CaptchaValidationProcessingFilter	Helps to identify a user as a human using Captcha techniques
ConcurrentSessionFilter	Ensures that a user is not simultaneously logged in more than a set number of times
HttpSessionContextIntegrationFilter	Populates the security context using information obtained from the http session
FilterSecurityInterceptor	Decides whether or not to allow access to a secured resource
AnonymousProcessingFilter	Used to identify an unauthenticated user as an anonymous user
ChannelProcessingFilter	Ensures that a request is being sent over HTTP or HTTPS
BasicProcessingFilter	Attempts to authenticate a user by processing an HTTP Basic authentication
CasProcessingFilter	Authenticates a user by processing a CAS (Central Authentication Service) ticket
DigestProcessingFilter	Attempts to authenticate a user by processing an HTTP Digest authentication
ExceptionTranslationFilter	Handles any AccessDeniedException or AuthenticationException
LogoutFilter	Used to log a user out of the application
RememberMeProcessingFilter	Automatically authenticates a user who has asked to be “remembered” by the application
SwitchUserProcessingFilter	Used to switch out a user. Provides functionality similar to Unix’s su
AuthenticationProcessingFilter	Accepts the user’s principal and credentials and attempts to authenticate the user
SiteminderAuthenticationProcessingFilter	Authenticates a users by processing CA/Netegrity SiteMinder headers.
X509ProcessingFilter	Authenticates a user by processing an X.509 certificate submitted by a client web Browser
SecurityContextHolderAwareRequestFilter	Populates the servlet request with a request wrapper.

Flow of a request through Spring Security's core filters



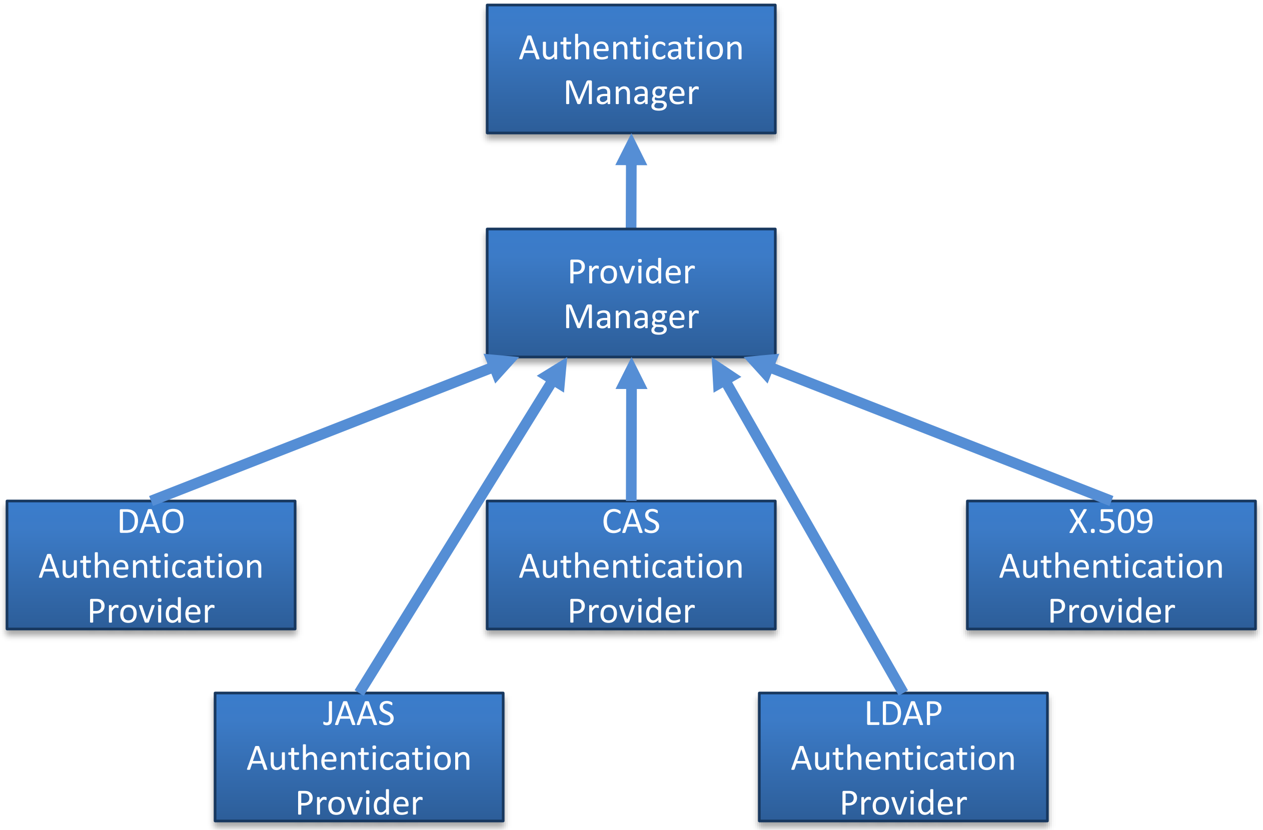
Spring Security

Authentication

Authentication Manager

- verifies *principal (typically a username) and credentials* (typically a password)

- Spring Security comes with a handful of flexible authentication managers that cover the most common authentication strategies


Authentication Manager
Provider Manager
DAO Authentication Provider
CAS Authentication Provider
X.509 Authentication Provider
JAAS Authentication Provider
LDAP Authentication Provider

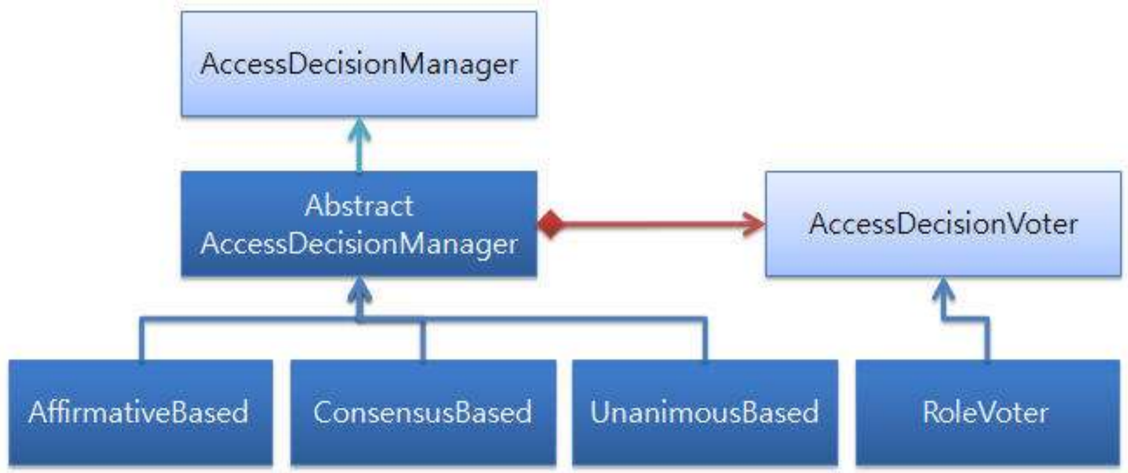
Spring Security

Authorization

Access Decision Manager

- responsible for deciding whether the user has the proper access to secured resources by invoking Voters and tallying votes

- Spring Security comes with three implementations of Access Decision Manager



Access decision manager	How it decides to grant/deny access
Affirmative Based	Allows access if at least one voter votes to grant access
Consensus Based	Allows access if a consensus of voters vote to grant access
Unanimous Based	Allows access if all voters vote to grant access

Web Authorization

```
<authz:authorize ifAllGranted="ROLE_MOTORIST,ROLE_VIP">
    Welcome VIP Motorist!<br/> <a href="j_acegi_logout">Logoff</a>
</authz:authorize>

<authz:authorize ifAnyGranted="ROLE_MOTORIST,ROLE_VIP">
    Welcome Motorist!<br/> <a href="j_acegi_logout">Logoff</a>
</authz:authorize>

<authz:authorize ifNotGranted="ROLE_ANONYMOUS">
    <p>This is super-secret content that anonymous users aren't allowed to see.</p>
</authz:authorize>

<authz:authorize ifAllGranted="ROLE_MOTORIST“
                 ifAnyGranted="ROLE_VIP,ROLE_FAST_LANE“
                 ifNotGranted="ROLE_ADMIN">
    <p>Only special users see this content.</p>
</authz:authorize>
```

Method Authorization

```
@Secured(“ROLE_ADMIN”)
@Secured(“ROLE_REGISTRAR”)
public void enrollStudentInCourse(Course course, Student student)
throws CourseException {
    ……
}
```

Sources

Spring in ACTION by Craig Walls
Chapter 7 – Securing Spring
http://www.manning.com/walls3/

http://static.springsource.org/spring-security/site/articles.html

Introducing Spring Security . Video of Ben Alex's Øredev 2008 Presentation. Includes walkthroughs on incrementally securing a demo web application (a variation of the "tutorial" sample).

Using Spring Security. Video of Mike Wiesner's presentation at Spring One 2008.

Diagrams http://blog.mindtheflex.com/wp-content/uploads/2008/07/uml.png