© May 2016 | IJIRT | Volume 2 Issue 12 | ISSN: 2349-6002

Identification, Implementation and Validation of Authentication and Authorization pattern in Distributed System using Spring Security Framework

Ketan Kumar[1], Prof. Kalpesh Patel[2]

[1]*P.G. Student, Computer Engineering, L.D. Engineering college, Ahmadabad, Gujarat, India*

[2]*Professor, L.D. Engineering college, Ahmadabad, Gujarat, India*

Abstract- **Software Security is an important part of Software development as the risk from attackers is constantly evolving through increased exposure. The main drawback in current approach is more concerned towards software requirement and not towards the security requirement of the software in SDLC. There are various techniques to identify the vulnerabilities which can be eliminated with the help of security pattern in software development life cycle. But the main concern is about which pattern is more efficient and can eliminate the vulnerabilities efficiently. The Security pattern must be applied in the Design phase and validated using USE TOOL that all the threats are mitigated. But Main aim is to identify the J2EEE pattern which can be implemented in spring Security Framework with the help of available Classes of Spring Security and this framework is more concern about Authentication and Authorization. So we are also focusing on both of this area. After identification of Security pattern it is tested in Design phase and checked Using USE TOOL whether it is mitigating all the threat and finally it is implemented in Spring Security Framework.**

***Index Terms*— Security pattern, secure software, software vulnerability, spring security framework.**

I. INTRODUCTION

Software security is an important quality aspect of any software system that manages valuable data and processes. If a system is not adequately secure, operational, reputational and business stakes rise significantly for both the users and owners of such a system

A Software Security Pattern [1] is defined as a generic well defined security solution proposed by software security experts to solve a recurrent problem in a given context. Using security patterns, developers can address security issues such as authentication, Authorization. Along with increasing popularity of security engineering with patterns, it is necessary to provide directions and guidelines helping system designers who are not security experts. Applying security patterns so far there is no clear, well documented and accepted process dealing with their full integration from earlier phases of software development until production of code [2].

But now again question arise that how developer integrate security pattern in secure software development process. And how apply security pattern so it give guarantee that threats and vulnerabilities are mitigated.Thus, our research aim to answer these research questions (RQ).

RQ1: Whether the Security pattern can be applied in Spring Security Framework.
RQ2: How software developer can apply security pattern in SDCL.
RQ3: How developer can validate that the applied Security pattern is Successful in mitigating the threat.

The organization of this document is as follows. In Section 2, the detail study of Spring Security framework and identifation of Security pattern is briefed. In Section 3, an approach to implement the security pattern in SDLC is proposed and in Section 4 conclusion is given.

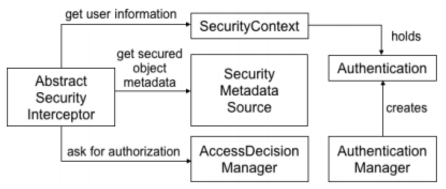
II. BACK GROUND

<u>What is Security Pattern?</u>

"A security pattern describes a particular recurring security problem that arises in specific contexts and presents a well-proven generic scheme for its solution."

<u>Details of Spring Security Framework:</u>

The core classes and their dependencies are shown in Figure. The Authentication class stores user information. It is part of a SecurityContext class for every authenticated user in an application. An AuthenticationManager loads this data and which verifies the authenticity of users using offered credentials and information from a user store.



<u>Identification of Security pattern and Classes of Spring Security Framework which can be applied to mitigate the threats</u>

The below table gives the general information about the threats, its countermeasure, security pattern as a countermeasure, and classes of spring Security Framework which can implemented as Security pattern.

Threats	Countermeasure	Security design pattern (as Countermeasure)	Identified classes for implementation in spring Security
1)Buffer overflow (leads to DOS attack) 2)Cross-site scripting 3)SQL Injection	Input Validation	1)Intercepting validater	1)Custom Validation 2)Javax validation 3)bean validation
1)Network eavesdropping (spoofing identity) 2)brute force attacks 3)dictionary attacks 4)cookie replay 5)credential theft	Identification and Authentication	1)Authentication enforcer 1)Account lockout	1) authentication manager, 2)authentication provider
1)Elevation of privilege 2)disclosure of confidential data 3)data tempering	Access control	1)Authorization Enforcer	1)Access decision manager
1)Session Hijacking 2)session replay 3)man in middle	Session management	1)Secure session manager 2)Secure pipe	1)Secure channel
1)user denies performing an operation(repudation) 2)attacker exploits an application without trace 3)attacker covers his or her tracks	Auditing and logging	1)Secure logger	1)logger

<u>Authentication and Authorization pattern identification</u>

A focus was put on authentication and authorization pattern as these are the main focus of the spring Security framework also.

<u>Authentication Patterns</u>

1) Authentication Information.

The main interface for implementing the Authentication Information pattern is the Spring Security Authentication interface, as its implementation offers information depending on the authentication mechanism. The Authentication interface is closely coupled to the Authentication Provider that loads the user information.

2) User Store pattern

Accessing storages with the Spring Security framework, as required by the User Store pattern, is achieved through different implementations of the Authentication Provider interface. Each implementation represents a different User Store and uses varying Authentication concretions.

3) Authentication Enforcer

Authentication Enforcer pattern is implemented using the filter chain mechanism introduced by the Java ServletSpecification.UsernamePasswordAuthenticationFilter tries to authenticate the client using the configured AuthenticationManager.

The filter chain and authentication provider offers flexibility in adding new authentication mechanisms and user stores needed to support the Authentication Enforcer pattern.

<u>Authorization Patterns</u>

1) Role-Based Access Control

RBAC raises the need for defining Roles of Users. In Spring Security Roles are called (Granted-) Authorities. Authorities can be assigned to Users via configuration or loaded from a User Store.

2) Attribute-Based Access Control

ABAC[9] is not directly supported by Spring Security, but can be easily implemented. Spring Security offers the Spring Expression Language (SpEL) to describe access control. Instead of annotating a right to methods or to a URL, expressions can be used. When evaluating to True, access is granted.

3) Identity –Based Access Control

IBAC [9]. In IBAC the set up of a database, holding the ACL and the configuration of Spring Security to use a database, is shown. For each Resource an Access Control Entry can be added to the database, giving specific Permissions to a Subject.

4)Authorization Enforcer

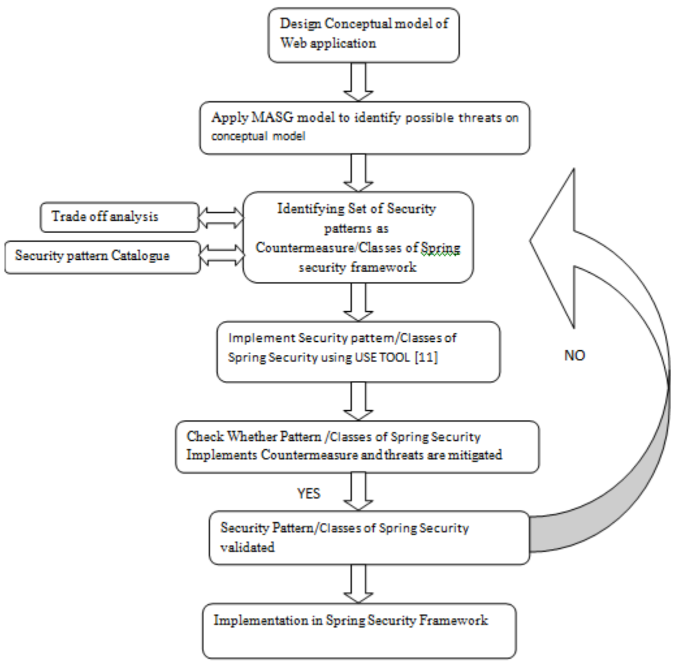
the Authorization Enforcer pattern can be implemented by using Spring Security access control.

III. PROPOSED APPROACH

In the proposed method as per [10] we decided to check the security concerns from the early phases of the life cycle like requirement phase and design phase.

In requirement Phase when the Identification of assets and stake holder and interaction between them is done with the help of Use case Diagram. We will Study the possible threats and vulnerability on the assets which are valuable in the system. And after analysing the possible attack we will identify the Security pattern as a countermeasure to the threat.

In Design phase, when the Counter measure is Identified we will implement it in USE tool [11] to check whether the Security Pattern as Countermeasure Can Successfully Eliminate the identified threat.


Design Conceptual model of Web application
Apply MASG model to identify possible threats on conceptual model
Trade off analysis
Security pattern Catalogue
Identifying Set of Security patterns as Countermeasure/Classes of Spring security framework
Implement Security pattern/Classes of Spring Security using USE TOOL [11]
NO
Check Whether Pattern /Classes of Spring Security Implements Countermeasure and threats are mitigated
YES
Security Pattern/Classes of Spring Security validated
Implementation in Spring Security Framework

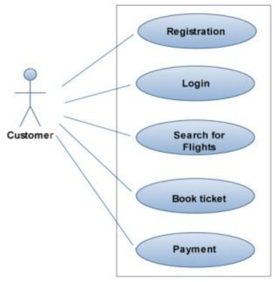
We will perform the following steps:

1. Design Conceptual model of the web application
2. Apply MASG model to identify the assets ,threat and countermeasure on conceptual model
3. Identify Security pattern/classes of Spring Security which can Countermeasure the Identified Threats.
4. Trade off Analysis- in Trade off analysis, the best possible security pattern is pick from the Collection of Security pattern of same kind.
5. Implement the Selected Security Pattern/classes of spring Security in the Design Phase with the help of USETool [11].
6. After Implementing, With the help of Security Requirement table, we will check whether the Security pattern implemented is successful in eliminating threats.
   If Security Requirement Satisfies then pattern is implemented successfully then Security pattern is validated but if it comes false. Then the identification of new security pattern is to be done which is the best possible suit.
7. Then follow Step: 3 to Step: 7.
8. If Security Pattern is validated then implement it in Spring Security framework.

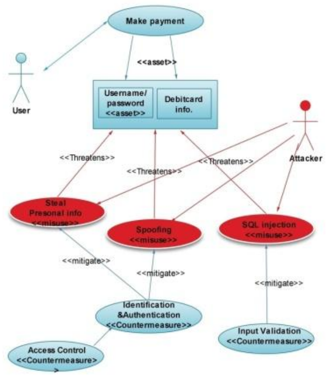
<u>Case Study</u>

Using our approach we try to design Secure Flight Reservation System. In which based on countermeasure we define security requirement and based on that we select set of j2ee security pattern to make secure shopping cart. Using definition of security pattern we try to create test case and using it we make sure that our method can contribute to mitigate all threats in early phases of SDLC.

Step1. Using conceptual model of Flight Reservation System create use case diagram to find non security asset and functional requirement. It describes functional requirement of flight reservation system or goal of user of cart application.



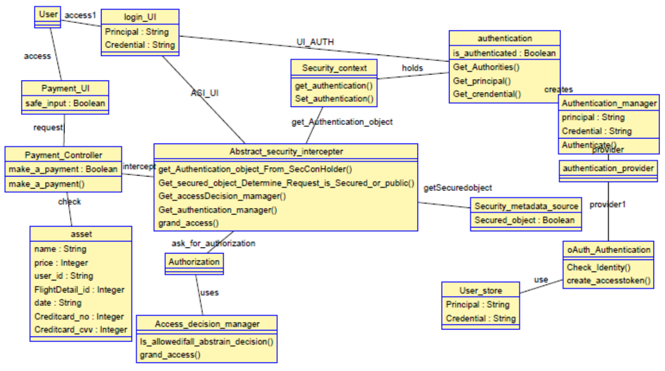
Step2. Apply MASG model on Use-case and find asset ,threat and counter measure.


Make payment
<<asset>>
User
Username/ password <<asset>>
Debitcard info.
Attacker
<<Threatens>>
<<Threatens>>
<<Threatens>>
Steal Presonal info <<misuse>>
Spoofing <<misuse>>
SQL injection <<misuse>>
<<mitigate>>
<<mitigate>>
<<mitigate>>
Identification &Authentication <<Countermeasure>>
Input Validation <<Countermeasure>>
Access Control <<Countermeasure>>

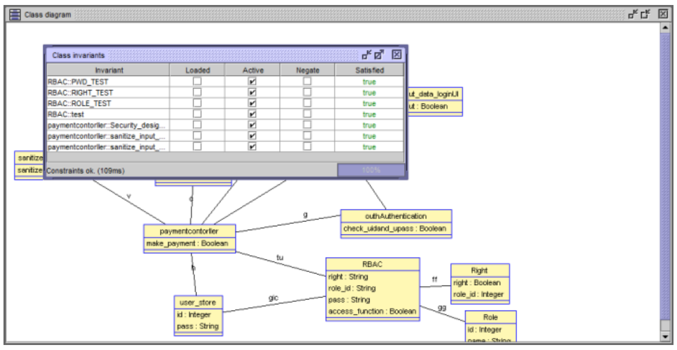
Step3. Now based on security requirement choose set of security pattern which can Fulfill security requirement and mitigate threats.

Countermeasure	**Security pattern**
Identification and Authentication	Authentication
Access control	Authorization
Input and Data validation	Intercepting validator
Auditing and logging	Log and audit
Access control	Single Sign on
Session management	Session management
Centralization of control	Secure base action
Secure communication	Secure pipe
Confidentiality	Cryptography and Authorization

Step4. Design class diagram with considering security means that class diagram with applying security pattern/ Classes of Spring Security framework.



Step5. Using definition of applied security pattern create test case in OCL [17]



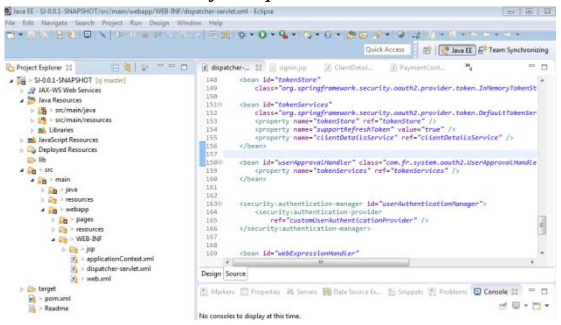
Invariant	Loaded	Active	Negate	Satisfied
RBAC::PWD_TEST	☐	✓	☐	true
RBAC::RIGHT_TEST	☐	✓	☐	true
RBAC::ROLE_TEST	☐	✓	☐	true
RBAC::test	☐	✓	☐	true
paymentcontorller::Security_desig...	☐	✓	☐	true
paymentcontorller::sanitize_input_...	☐	✓	☐	true
paymentcontorller::sanitize_input_...	☐	✓	☐	true

Constraints ok. (109ms)

Step6. now give input to USE tools [11] as class diagram (step4) and test case (step5).Execute tests to validate an input model satisfies security requirements and set of security pattern which we applied is creates secure and robust design of given application.

		1	2	3	4	5	6	7	8
Condition	The Same Principal and credential that are inputted into 'Login UI' exists in 'User store'	Yes	Yes	Yes	Yes	No	No	No	No
	Access permission is given in 'Role' to which User belong	Yes	Yes	No	No	Yes	Yes	No	No
	Sanitize Input Data in UI	Yes	No	Yes	No	Yes	No	Yes	No
Actions	Consider Regular user	X	X	X	X				
	Consider Non Regular user					X	X	X	X
	Consider that User have access Permission	X	X			X	X		
	Consider the user Does not have access permission			X	X			X	X
	Consider that valid input data is used	X		X		X		X	
	Consider that invalid input data is used		X		X		X		X
	Execute 'make a payment process'	X							
	Not Execute 'make a payment process'		X	X	X	X	X	X	X

Step8. Implement the Security pattern/spring classes which is validated in Use tool in Spring Security Framework and check whether the Security Requirement if fulfilled.



```
<bean id="tokenStore"
    class="org.springframework.security.oauth2.provider.token.InMemoryTokenSt

<bean id="tokenServices"
    class="org.springframework.security.oauth2.provider.token.DefaultTokenSer
    <property name="tokenStore" ref="tokenStore" />
    <property name="supportRefreshToken" value="true" />
    <property name="clientDetailsService" ref="clientDetailsService" />
</bean>

<bean id="userApprovalHandler" class="com.fr.system.oauth2.UserApprovalHandle
    <property name="tokenServices" ref="tokenServices" />
</bean>


<security:authentication-manager id="userAuthenticationManager">
    <security:authentication-provider
        ref="customUserAuthenticationProvider" />
</security:authentication-manager>


<bean id="webExpressionHandler"
```

No consoles to display at this time.

IV. CONCLUSION

Based on our studies, we found that the Spring Security Framework is a open source java framework, which provides authentication, authorization and access control Solution. We also found that the Security pattern is Quiet helpful in giving Best possible Solution to the Security problem. But we are quite surprised by reviewing the papers that there are no methodology to use the security pattern in Spring Framework. So we proposed a method by which Software developer can Implement and validate the security pattern in Design phase of SDLC Using USE Tool and implement it in spring framework in implementation phase

REFERENCES

[1] M.Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann and P. Sommerlad, Security Patterns:Integrating Security and System Engineering Wiley, 2006, Pg-600,

[2] T Devanbu and S.Stubblebine,"Software Engineering for Security: A Roadmap," in Proceeding of the conference of the Future of software engineering, 2000

[3] Kienzle, D., M. Elder, D. Tyree, and J. Edwards-Hewitt. "SecurityPatternsTemplateandTutorial".http://www.securitypatterns.com/documents.html, February 2002

[4] Aleksander Dikanski, Roland Steinegger, Sebastian Abeck" Identification and Implementation of Authentication and Authorization Patterns in theSpring Security Framework", Secureware 2012

[5] The Open Group. "Guide to Security Patten"http://www.opengroup.org/security/gsp.htm, 2002

[6] Munawar Hafiz, Paul Adamczyk, Ralph E. Johnson," Towards an Organization of Security Patterns"IEEE 2013

[7] "Misuse cases + Assets + Security Goals",International Conference on Computational Science and Engineering,2009

[8] C. Steel, R. Nagappan, and R. Lai, Core Security Patterns, 1st ed. Upper Saddle River, N. J.: Prentice Hall International, 2005, p. 1088

[9] E. Yuan, J. Tong, B. Inc, and V. McLean, "Attributed Based Access Control (ABAC) for Web Services," Proc. IEEE International Conference on Web Services (ICWS), Orlanda, Florida, July 2005.

[10] AleksanderDikanski, Sebastian Abeck"“Towards a Reuse-oriented Security Engineering for Web-based ApplicationsAnd ServicesSecureware2012

[11] USE: "Uml- based Specification Environment", sourceforge.net /projects/ useocl

[12] TakanoriKobashi, Nobukazu Yoshioka, Takao Okubo "Validating Security Design Pattern Applications Using Model Testing" IEEE 2012