FROM THREATS TO SECURITY PATTERNS: TOWARDS SECURE SOFTWARE DEVELOPMENT BASED ON AUTHENTICATION, AUTHORIZATION AND SECURE SESSION PATTERN.

Foram Bhavsar[1], Dhiren Prajapati[2]

[1]*M.E Student, Computer Engineering, Merchant Engineering Collage, Basna, Mehsana, Gujarat*
[2]Asst. Professor, Computer Engineering, Merchant Engineering Collage, Basna, Mehsana, Gujarat

---***---

Abstract - *Software Security is an important part of Software development as the risk from attackers is constantly evolving through increased exposure. Strategies for software development often slight security related considerations due to difficulty of developing realizable requirements, identifying and applying appropriate techniques to develop software. 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. There methods to implement security pattern in early stages of software development lifecycle and to validate them that all the available Threats and vulnerabilities in a system is eliminated with the use of appropriate security pattern in Spring framework.*

***Key Words*: Security pattern, Secure software, Spring Security Framework**

1.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 [2] 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].

Security modelling is a process of identifying threats, which may give the required understanding to design or identify vulnerability countermeasures. The identification of the potential threats and vulnerabilities must be done in earlier stages of the SDLC to mitigate the threats early as possible but there are many threats which can be identified in the rest of the phases so it is a must to identify the possible threats in every phase of the SDLC.

1.1 Research Objectives:

- Identify threats and vulnerabilities in a Web application during SDLC.
- Analyze J2EE security pattern.
- Analyze the classes of Spring Security framework.
- Using those Security pattern give the reusable approach to the software developer.
- Bridging the gap between developers and security experts.
- Adapt identified security pattern to phases of software development.

The Organization of this Document is as follow: In Section 2, The Details study of Spring Security Framework and indentation of security pattern in briefed. In Section 3, An approach to implement the security pattern in SDLC is proposed and in Section 4 Conclusion is given.

2. BACK GROUND

2.1 Security pattern

A software pattern describes a recurring problem that arises in specific contexts, and presents a well-proven generic solution. The solution can form multiple concrete instantiations, and has become a pattern through many successful implementations, proving itself as a best practice. The use of patterns has been adopted in the security field. A substantial collection of security patterns has been developed. The definition of a security pattern has the same key characteristics as general software patterns [2]:

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

2.2 Spring Security Framework

Spring Security[8] provides versatile and Protractible resolution for authentication, authorization and access management. The main aim of spring security framework is to secure net aapplication supported J2EE platform. Many technologies and standards are unified within the framework like light-weight Directory Access Protocol (LDAP), Central Authentication System (CAS), OpenID and

OAuth. other forms of authentication and authorization can be easily adapted because of the flexible architecture.

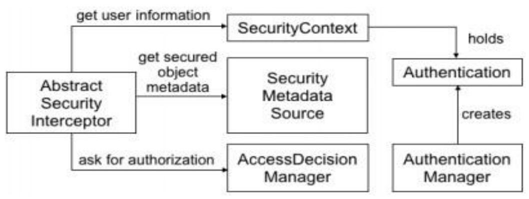
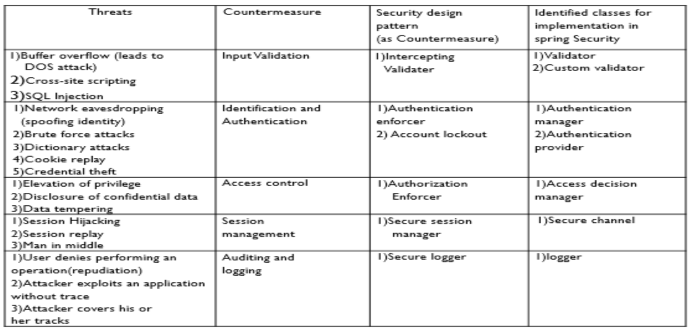


Fig -1: Primary classes of spring Security Framework [8]

The Spring Security framework contains loosely coupled elements, who are connected Victimization Dependency injection. the most categories and their dependencies are in figure one The Authentication category contains user data and is a component of a Security Context class for every attested user in application. Authentication Manager hundreds this data and verifies the credibility of users by offered user name and positive identification from user store. The secured resources are intercepted by AbstractSecurityInterceptorclass, Categories extend this category class in term of authorization. Here the Security Context and SecurityMetadataSource categories provide data on the user and also the secured object severally. selections on assess are performed by the AccessDesionManager, that is termed by the AbstractSecurityInterceptor. AccessDesionManager decides with the assistance of elector whether or not access ought to be granted or not and it will be accessorial dynamically to the applying.

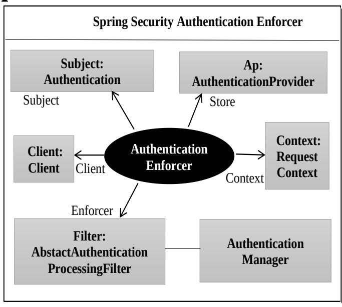
2.3 Identification of security pattern in spring security framework

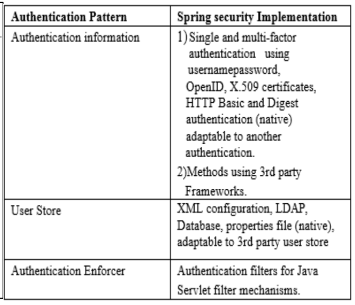


Threats	Countermeasure	Security design pattern (as Countermeasure)	Identified classes for implementation in spring Security
1)Buffer overflow (leads to DOS attack) 2)Cross-site scripting 3)SQL Injection	Input Validation	1)Intercepting Validater	1)Validator 2)Custom validator
1)Network eavesdropping (spoofing identity) 2)Brute force attacks 3)Dictionary attacks 4)Cookie replay 5)Credential theft	Identification and Authentication	1)Authentication enforcer 2) 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	1)Secure channel
1)User denies performing an operation(repudiation) 2)Attacker exploits an application without trace 3)Attacker covers his or her tracks	Auditing and logging	1)Secure logger	1)logger

Table -1: Threats and countermeasure as security pattern and spring security classes

Spring Security implementation of Authentication pattern



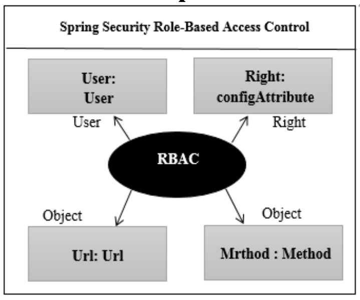


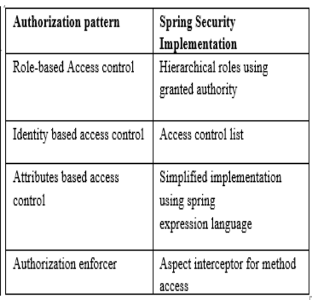
Authentication Pattern	Spring security Implementation
Authentication information	1) Single and multi-factor authentication using usernamepassword, OpenID, X.509 certificates, HTTP Basic and Digest authentication (native) adaptable to another authentication. 2)Methods using 3rd party Frameworks.
User Store	XML configuration, LDAP, Database, properties file (native), adaptable to 3rd party user store
Authentication Enforcer	Authentication filters for Java Servlet filter mechanisms.

Fig -2: Spring security Implementation of Authentication Enforcer [8]

Table -2: Supported Authentication Patterns [8]

Spring security framework implementation of Authorization pattern





Authorization pattern	Spring Security Implementation
Role-based Access control	Hierarchical roles using granted authority
Identity based access control	Access control list
Attributes based access control	Simplified implementation using spring expression language
Authorization enforcer	Aspect interceptor for method access

Fig 3: Implementing RBAC with spring security [8]

Table -3: Supported Authorization patterns [8]

3. PRPOSED METHODOLOGY

In the proposed method as per [22] 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 analyzing 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 [7] to check whether the Security Pattern as Countermeasure Can Successfully Eliminate the identified threat.

3.1 Flow of Proposed Method

We will perform the following steps:

Step:1 Design Conceptual model of the web application.

Step:2 Apply MASG model to identify the assets, threat and countermeasure on conceptual model

Step:3 Identify Security pattern/classes of Spring Security which can Countermeasure the Identified Threats.

Step:4 Trade off Analysis- in Trade off analysis, the best possible security pattern is pick from the Collection of Security pattern of same kind.

Step:5 Implement the Selected Security Pattern/classes of spring Security in the Design Phase with the help of USE Tool.

Step:6 After Implementing, With the help of Security Requirement table, we will check whether the Security pattern implemented is successful in eliminating threat. 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.

Step:7 Then follow Step: 3 to Step: 7.

Step:8 If Security Pattern is validated then Implement it in Spring Security framework.

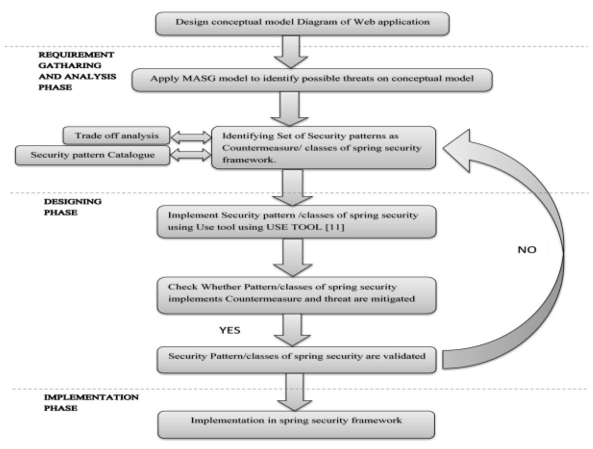

Design conceptual model Diagram of Web application
REQUIREMENT GATHARING AND ANALYSIS PHASE
Apply MASG model to identify possible threats on conceptual model
Trade off analysis
Security pattern Catalogue
Identifying Set of Security patterns as Countermeasure/ classes of spring security framework.
DESIGNING PHASE
Implement Security pattern /classes of spring security using Use tool using USE TOOL [11]
NO
Check Whether Pattern/classes of spring security implements Countermeasure and threat are mitigated
YES
Security Pattern/classes of spring security are validated
IMPLEMENTATION PHASE
Implementation in spring security framework

Fig -5: Flow of Proposed methodology

4. Implementation: Case Study

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 design secure software using j2ee security pattern.

Step 1. Create conceptual model of Secure Flight Reservation System

First start with create conceptual model of application. It is very simple architecture which gives us brief idea about application functionality and service provided by it. In flight Reservation web application customer use flight reservation for creating account, search flights, booking flight, make a payment process etc. where Search flight contain flight detail, etc.

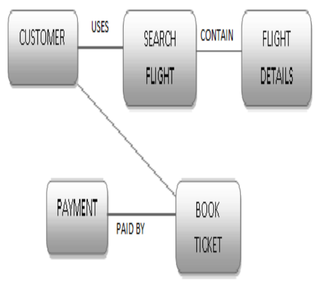


Fig -6: Conceptual Model of Application

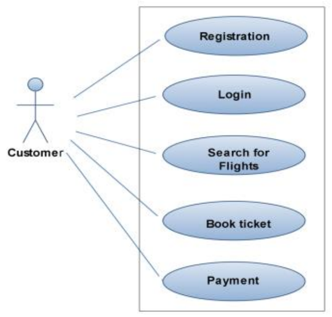


Fig -7: Customer Use case Diagram

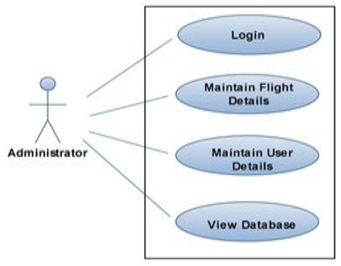


Fig -8: Administrator use case

Step 2. Apply MASG model on Use-case and find asset, threat and counter measure.

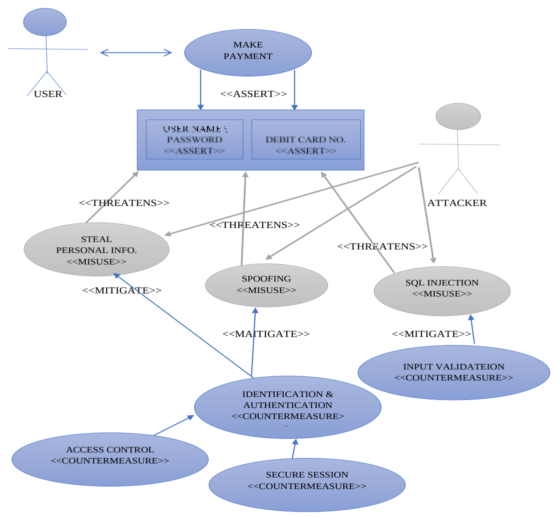

USER
MAKE PAYMENT
<<ASSERT>>
USER NAME \ PASSWORD <<ASSERT>>
DEBIT CARD NO. <<ASSERT>>
ATTACKER
<<THREATENS>>
<<THREATENS>>
<<THREATENS>>
STEAL PERSONAL INFO. <<MISUSE>>
SPOOFING <<MISUSE>>
SQL INJECTION <<MISUSE>>
<<MITIGATE>>
<<MAITIGATE>>
<<MITIGATE>>
INPUT VALIDATEION <<COUNTERMEASURE>>
IDENTIFICATION & AUTHENTICATION <<COUNTERMEASURE>
ACCESS CONTROL <<COUNTERMEASURE>>
SECURE SESSION <<COUNTERMEASURE>>

Fig -9: Mis use case Diagram (Make Payment)

Step 3. 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

Table -5: Countermeasure and related security pattern

Step 4. Design class diagram with considering security means that class diagram with applying security pattern/ Classes of Spring Security framework.

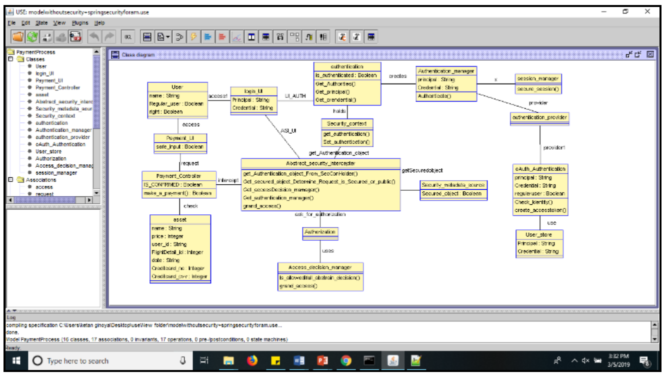
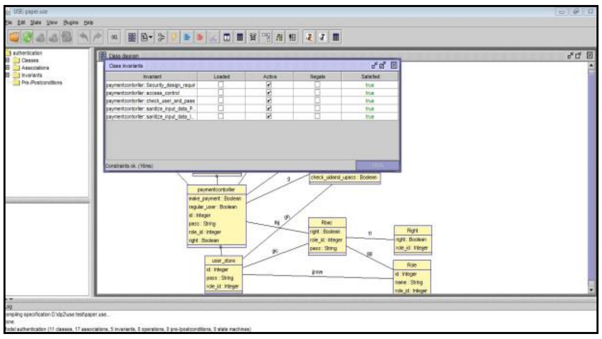


Fig -10 : Class diagram using Use tool

Step 5. Using definition of applied security pattern create test case in OCL (Fig 11) [17]



Invariant	Loaded	Active	Negate	Satisfied
paymentcontorller::Security_design_requir	☐	✓	☐	true
paymentcontorller::access_control	☐	✓	☐	true
paymentcontorller::check_user_and_pass	☐	✓	☐	true
paymentcontorller::sanitize_input_data_P...	☐	✓	☐	true
paymentcontorller::sanitize_input_data_I...	☐	✓	☐	true

Constraints ok. (16ms)

Fig -11 : Constraints on classes using OCL

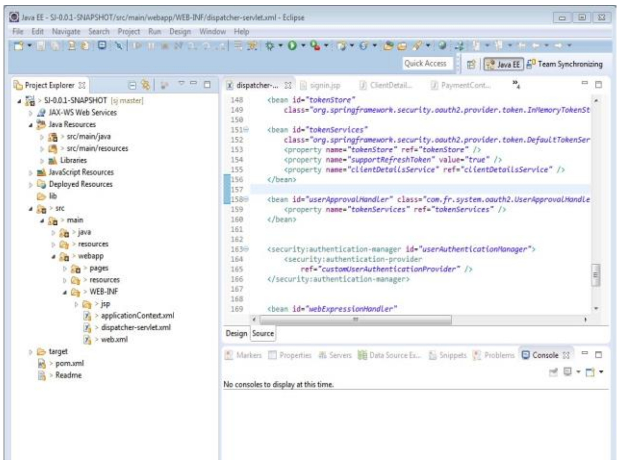
Step 6. Now give input to USE tools [18] as class diagram (step4) and test case (step5). Run tests to check that the model fulfills the security requirements.

		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 entered	X		X		X		X	
	Consider that invalid input data is entered		X		X		X		X
	Execute „make a payment process"	X							
	Not Execute „make a payment process"		X	X	X	X	X	X	X

Table -6: Security Requirement which must be satisfied after applying classes of Spring Security Framework

Step:7 Then follow Step: 3 to Step: 7.

Step 8. 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.

Fig -12: Implementing in spring

5. CONCLUSIONS

Based on my studies I found out that the Spring Security Framework is a open source java framework, which provides authentication, authorization and access control Solution. I also found that the Security pattern is Quite helpful in giving Best possible Solution to the Security problem. But we are quite surprised by reviewing the papers that there is no methodology to use the security pattern in Spring Framework. So, we proposed a method by which Software developer can use security pattern in Design phase of SDLC.

REFERENCES

[1] Allan holmes" Top vulnerabilities and what to do about them" Computer world UK, Jan 2010

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

[3] Kienzle, D., M. Elder, D. Tyree, and J. Edwards-Hewitt. "Security Patterns Template and Tutorial". HTTP://www.securitypatterns.com/documents.html, February 2002 AleksanderDikanski, Sebastian Abeck" "Towards a Reuse-oriented Security Engineering for Web-based Applications and Services Secureware 2012

[4] AleksanderDikanski, Sebastian Abeck "Towards a Reuse-oriented Security Engineering for Web-based Applications and Services" Secureware2012

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

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

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

[8] AleksanderDikanski, Roland Steinegger, Sebastian Abeck" Identification and Implementation of Authentication and Authorization Patterns in the Spring Security Framework", Secureware 2012

[9] B. Schneier. "Attack trees: Modelling security threats". Dr. Dobb's Journal, 1999

[10] D. Byers, S. Ardi, N. Shahmehri, and C. Duma. Modelling software vulnerabilities with vulnerability cause graphs. In Proceedings of the International Conference on Software Maintenance, Philadelphia, PA, USA, 2006

[11] Seamonster-Security-sourceforge.net/projects/seamonster/.

[12] M. Wiesner, "Introduction to Spring Security 3 /3.1," Spring One 2GX. Chicago, Oct.-2010.

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

[14] Implementation Support of Security Design Patterns Using Test Templates www.mdpi.com/journal/information 2016

[15] "Security Patterns: Research Direction, Metamodel, Application and Verification" 978-1-5386-2038-0/17/$31.00 c 2017 IEEE

[16] "A classification methodology for security patterns to help fix software weaknesses" 978-1-5090-4320-0/16/$31.00 ©2016 IEEE 2016

[17] A Misuse Pattern for Transport Layer Security (TLS): Triple Handshake Authentication Attack ALI ALKAZIMI, FLORIDA ATLANTIC UNIVERSITY EDUARDO B. FERNANDEZ, FLORIDA ATLANTIC UNIVERSITY 2016
[18] “A Methodology for Modelling and Analysis of Secure Systems Using Security Patterns and Mitigation Use Cases” 2018 7th International Conference on Computer and Communication Engineering (ICCCE) 978-1-5386-6992-1/18/$31.00 ©2018 IEEE
[19] 2018 ACM/IEEE 1st International Workshop on “Security Awareness from Design to Deployment Security Patterns 2.0”
[20] 2017 International Conference on “Software Security and Assurance A Comprehensive Pattern-Driven Security Methodology for Distributed Systems”
[21] “Assessing and improving the quality of security methodologies for distributed systems”© 2018 John Wiley & Sons, Ltd.
[22] The Open Group. “Guide to Security Patterns”. http://www.opengroup.org/security/gsp.htm, 2002.