Implementation of Secure Software Design and their impact on Application

Zia Ahmad
Department Of Computer Science, National Textile University Faisalabad Pakistan

Muhammad Asif, PhD
Assistant Professor, Department of Computer Science, National Textile University Faisalabad Pakistan

Muhammad Shahid
Lecturer, Department of Computer Science, National Textile University Faisalabad Pakistan

Adeel Rauf
Department Of Computer Science, Virtual University of Pakistan.

ABSTRACT

Vulnerability is associated with the system and it is a big risk for system and result in surplus maintenance cost. It is due to many reasons those are not considered during the stages of System Development Life Cycle (SDLC). During SDLC it may be reduced to minimum level. Millions of dollars waste due to vulnerable application and rescind working. Most of the software are not secure and cause Physical and Financial mutilation. It may not be possible to eliminate vulnerability completely but it might be reduced to the minimum level because it is the ongoing process. A web application using secure design patterns (SDPs) is presented in this paper. Two secure design patterns and their implementation are given. Secure Strategy Design Pattern (SSDP) and Secure Builder Design Pattern (SBDP) are purposed for two different forms SSDP is used for Driver information page and SBDP is used for Route information page. Special codes are used for inquiring whether valid user is using site or not. A class of encryption/decryption technique is added to add security. An encryption/decryption technique named SHA-1 is used. The result shows that SDPs are beneficial to all application developers especially for the developers of critical and sensitive systems. The system suits secure and design pattern makes it simple to understand its functionality. However, any other encryption/decryption techniques may also be applied on it in place of SHA-1. In future we plan to attach this class with other design patterns to make them secure from attackers and eliminate vulnerable points. Many features can be included in web application with the help of different design patterns and can be secured by attaching encryption/decryption class.

Keywords: design patterns, pattern selection, security, software engineering, security pattern, refactoring, design vulnerability, secure software design, secure pattern.

1. INTRODUCTION

Vulnerability is a feebleness which allows an attacker to change and use system's information and use system to bring down attackers vital functionality. It is the insertion of three elements 1) a system susceptibility or flaw 2) attacker access to the flaw 3) attacker capability to exploit the flaw. It is a threat to the software. Vulnerability is also known as a fault in the security of an information system that some time may be known or unknown. There may be many reasons of vulnerability, one of them in which security is compromised, reduced, a reason may be code mistake, there may be untrained users, insecure configuration setting may also one big reason of it. Vulnerability is a state in which security measures are compromised, reduced or lacked. Secure design patterns are intended for two main reasons:

- They may reduce the accidental insertion of vulnerabilities into code
- It may lessen the penalties of these vulnerabilities.

The creation of secure design pattern is to block entrance of accidental vulnerability in code and reduce the hazard of vulnerability[1]. Vulnerability is simply a design flaw or an implementation bug that allows a potential attack on the software in some way[1]. A high risk of attack is the information that is immediately communicated when software system are busy in communication.

1.1 History of Design Pattern

Alexander introduced the concept of pattern which is based on town and building design in 1977/79[2]. Beck and Cunningham experimented with applying patterns to programming and presented at (OOPSLA '87 workshop on Specification and Design for Object-Oriented Programming) OOPSLA[3]. Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides (Gang of Four) presented large number of design patterns in 1994/95[4]. The 23 patterns were introduced in the classification of First Behavioral, Second Structural, Third Creational and they contains patterns like State, Strategy, Visitor, observer for the first, Adapter, Bridge, Decorator, Façade for the second and Singleton, Builder, prototype & Factory etc. Design level patterns implemented with real life problems. The change in logical structure results in efficiency reduction. Achievements for software quality are more significant and therefore software maintenance cost is reduced and results optimal performance harm[5]. Design Pattern (DP) Reliable solution for frequently occurring problem in software design is known as design pattern. Such solution cannot be converted into machine or source code directly. This solution can be used in many different situations. DP is recurring solution of real world problem. DP not only helps to reuse code but also improve extensibility and maintainability. It is proved that DP build flexible software architecture. For the new developer selection of appropriate design pattern is a hard task. DP, on the other hand having bad effect at efficiency it is due to interaction among views and data through method call, many DP has this problem. DP captures the best practices for solving recurring software design problems. For software security is most important topic. The attackers in today's environment can easily harm software working in industry, commercial application because they are poor in quality and design with various weaknesses. The news heard may time over all around the world about attack on database or website resulting in the loss of millions of dollars either directly or indirectly. Therefor most software is not secure in recent times and is the big reason for financial and physical damages. This is because most of the efforts placed onto maintenance and development processes. These vulnerabilities of system and their associated risk after software completion become very high for both user and customer if needed to fix it. Due to the importance of security flaws it must be considered in all phases of software development lifecycle (SDLC). There is a change of attack model day by day on software systems. Attackers are getting more inventive, and more complex attackers are being constructed and formulated[1]. The basic root of all cybercrime may be considered is Networking. The data when transferred from any source of networking is the invitation to corrupt the application. An in depth research has already been performed in the field of security patterns. Secure Design Pattern (SDP) is used to eliminate the unintentional addition of vulnerability into code and to reduce vulnerability.

Strategy Design Pattern describes a family of algorithms, encapsulate each one, and make them exchangeable. Strategy allows the algorithm differ freely from customers that use it. The classes and objects participating in this pattern are: Strategy (Sort Strategy): declares an interface common to all supported algorithms. Context uses this interface to call the algorithm defined by a Concrete Strategy. Concrete Strategy (Quick Sort, Shell Sort, and Merge Sort): implements the algorithm using the Strategy interface. Context (Sorted List): is configured with a Concrete Strategy object, maintains a reference to a Strategy object, and may define an interface that lets Strategy access its data.

Builder Design Pattern disseminated the creation of an intricate object from its illustration so that the same creation process can create different representations. A Builder class builds the absolute entity step by step. This builder is sovereign of other objects.

Encryption is the renovation of electronic data into another form, called ciphertext, which cannot be certainly understood by anyone except legal revelries.

Decryption is the process of taking encoded or encrypted text or other data and mutable it back into text that you or the computer are able to read and understand. This term could be used to describe a method of un-encrypting the data manually or with un-encrypting the data using the proper codes or keys. Data may be encrypted to make it difficult for someone to snip the information. Some companies also encrypt data for general protection of company data and trade riddles. If this data needs to be viewable, it may require decryption. If a decryption passcode or key is not presented, distinctive software may be required to decrypt the data using algorithms to fissure the decryption and make the data readable. Many encryption/decryption techniques are used like SHA-1, SHA-3, SHA-3, SHA-256 and SHA-512.

A pattern is a reusable structure in different applications represents the knowledge and experience of software application developer. Any specific problem can be solved by a single and/or a number of patterns in given context and can be prepared to work in different situations. Analysis patterns can be used to build conceptual models and security patterns can be used to build secure systems[6, 7]. In requirements stage use cases define the required interactions with the system. From the use cases the needed rights for each actor can be determined and thus apply a need-to-know policy[8]. Design patterns are a familiar tool used by the software development community to help solve recurring problems encountered during software development[9]. These patterns try to address head on the thorny problems of secure, stable, and effective software architecture and design. Since the introduction of design patterns, many other types of patterns relevant to software have been conceived, including a relatively new construct known as attack patterns[10].To identify the vulnerability from the application being to be developed according to its logic and design is the vital step when gathering requirements, application design and developing system from the security perspective. By mapping and ranking the system vulnerabilities it is possible to determine the preliminary attack surface of the system, the likely points that an attacker would chose to explore for potential compromise [1]. It should be completed before the completion of code because as the development progress the attack surface always increases. An attacker sends malicious input and tries to compromise the system and one case is also important that system exiting the traffic called "information leakage" [1]. It is the processing of application revealing large amount of information during its original processing. The sensitive information sending with too much error by which errors can be manipulated can be used with remaining number of attempts to guess a password is because of the information leakage[1]. The techniques are learned, understand and enhanced by attackers and defenders as attackers attack, and defenders defend, so security cannot taken as static and it is an arms race. The attackers attempts for anticipate and stay ahead and defenders acquire from vulnerabilities and make better ways for being compromised. Algorithm solves computational problems not the design problem so they are not design pattern. The creation of secure design pattern is to block entrance of accidental vulnerability in code and reduce the hazard of vulnerability[1]. In contrast to the design-level patterns popularized in the issue of security is discussed in secure design pattern with variety of range specifying architectural level patterns ,high level design and implementation level patterns road map for implementing methods and functions in application[4]. Software development is not an insignificant task; treks (methods of understanding manual working) are used to gather requirements from stakeholders. Small software may be hard to develop and large software may be easy and may not provide resistance. A pattern is a form of knowledge for capturing a recurring successful practice[11]. They explicitly capture knowledge that experienced developers understand implicitly and facilitate training and knowledge transfer to new developers[12].

Unified Modeling Language (UML) diagrams allow a direct incorporation of the security element[1]. The preliminary attack surface of the system can be determined by mapping and ranking of system vulnerabilities. These may be points that an attacker can chose for potential compromise. The surface of attack may be increased as the development proceeds. The expected points of attack must be identified before the implementation of mitigate techniques at design level[1]. The techniques are learned, understand and enhanced by attackers and defenders as attackers attack, and defenders defend from previous study so security cannot taken as static and it is an arms race. The attackers attempts for anticipate and stay ahead and defenders acquire from vulnerabilities and make better ways for being compromised. In design when problem occurs commonly a pattern is reusable solution.

1.2 Related Work

A technical report on Secure Design Patterns was published by Software Engineering Institute on March, 2009[13]. There are three categories of Secure Design Patterns (SDPs) Architectural-Level Pattern, Design-Level Pattern and Implementation-Level Pattern[13]. In the work proposed at Design-Level Patterns include Factory, Strategy, Builder, Chain of Responsibility, State and Visitor. The work at them represents the secure working by the patterns. A book on security of software design was given by Theodor Richardson and Charles Thies named "Secure Software Design" in 2013[1]. In this book introduction of vulnerability is given and discussed about current and emerging threats are described. Moreover, in this book a method is provided by author that by using UML diagrams. The emerging and current threats are discussed under the circumstances of human factor, over network, in operating system environment, data management and data centric threats. These all explains the importance of removing threats in described circumstances also. In 2014 more work done at the DP as Application and Open Issues: Design Patterns. This paper said that the Observer Pattern of DP is studied and its adaptability and extensibility are increased. This work reduces the efficiency but it is the problem of most DPs. This is due to the improvement in the logical structure[5]. Threat is a possible exploit of a vulnerability where an attacker is the actual use of such an exploit[1].

2. PROPOSED MODEL

2.1 Problem Statement

The application developers consistently working on the reliability, availability and reusability but they continuously ignore design of application. It is the major drawback which becomes real cause of point that is being compromised and may provide potential attraction to attackers. Software design becomes the biggest issue and leaked or compromised point that may become welcome point for the vulnerability. For this reason developers and end users have to pay more for fixing the system vulnerability and its associated risk after the system deployed. There are many types of practices in use to report of software security vulnerability but these are difficult to use. Many current best security practices focus on implementation and deployment issue and so do not address security flaws introduced in earlier phase of the development process[13].

2.2 Model

Existing DPs converted in to secure design pattern after taking security issue into account. The proposed secure DPs of this paper and their practical developed application provide benefit for developers of secure software product. The use of these patterns will help to reduce the cost and associated risk of vulnerability. After the arrangement and grouping of use cases system functionality can be mapped and try to find if there are any gaps; if gaps found new use case diagram construct to bridge the gap. Failure to do this will force the developer to make this connection and it could open new security holes[1]. Developers are excellent problem solvers and creative workers. However, a good system will have well defined specifications from the outset, meaning those creative problem solving skills can be applied to better coding instead of determining some new and unspecified method for point A to connect to point G because no one define B through F in the system planning[1].

An application developed by using the proposed solution is a web based application, which uses Google Maps APIs "Map My Way". It is developed to facilitate the users of any Organization to define their routs on Google Map, so that others colleagues of same organization can see/view routes and get the information about their weekly journey plan and addresses of their home to office from this application. Any user can define route. But if user wants to see all other similar routes defined by other users then user will have to login. If new user comes to this website then registration is required. After login user can view/see all required per-defined routes of same organization. User can send friend request to other users having same/similar routes. This application also gives the functionality for users to contact with their colleague via SMS or Email. By using address information the colleague can ask people to drop them at their home. So, to obtain this goal we get such a system which can maintain colleague personal information and their vehicle information (if colleague have their own vehicle). Personal information about any user of this site will be hidden to any other. Admin will be responsible to manage site. However, user will manage their route and other information themselves.

2.3 Secure Strategy Design Pattern

This page is established by using the Secure Strategy Design Pattern and vulnerabilities exposed by the UML diagram are removed, moreover it is now secure strategy design pattern. This page is used to add new driver with its vehicle and personal information. After identifying pointes to be open for attack and presence of vulnerability, the vulnerability can be reduced to minimum by secure strategy design pattern. UML diagrams are used here to show vulnerability; in this page Figure 1 is about use case diagram and Figure 2 for this page is sequence diagram and Figure 3 is class diagram. The driver submit required information asked on the page and the system ask him the security credentials at the same time the attacker may be available to do this as shown in Figure 1. These are expected vulnerable points for the attacker. The expected solution is, the driver when select the vehicle type to select bike, car, van or APV the system ask a security code from the driver which is already stored in the database in encrypted form which is then compared from the code that is provided by the driver. If the security code matches with the database encrypted code the submit process will be done otherwise it asks the correct code 2 times again if not provided the correct code the system will logout, no more time for the malicious user to use the website for specific time period or may have to login again.

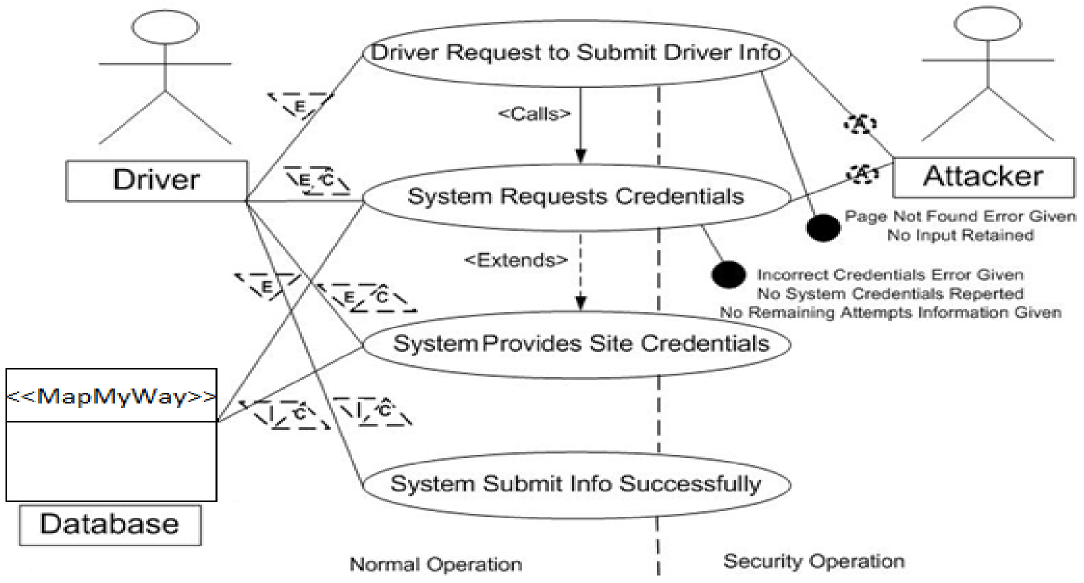

Driver Request to Submit Driver Info
<Calls>
System Requests Credentials
<Extends>
System Provides Site Credentials
System Submit Info Successfully
Driver
Attacker
<<MapMyWay>>
Database
Page Not Found Error Given
No Input Retained
Incorrect Credentials Error Given
No System Credentials Reperted
No Remaining Attempts Information Given
E
E C
I C
A
Normal Operation
Security Operation

Figure 4: use case diagram representing Vulnerability

The use case diagram for secure strategy pattern indicates that [E] and [C] points. Any place where both [E] and [C] are located on the same association, there must be one of the most likely target for attack[1]. At the same time, when [I] and [C] are located together, a level of protection must be added to the data in transit; otherwise, there is no protection from internal threats sitting on the same network or attackers recording the transmission from the another node in the network such as a wireless hub[1].

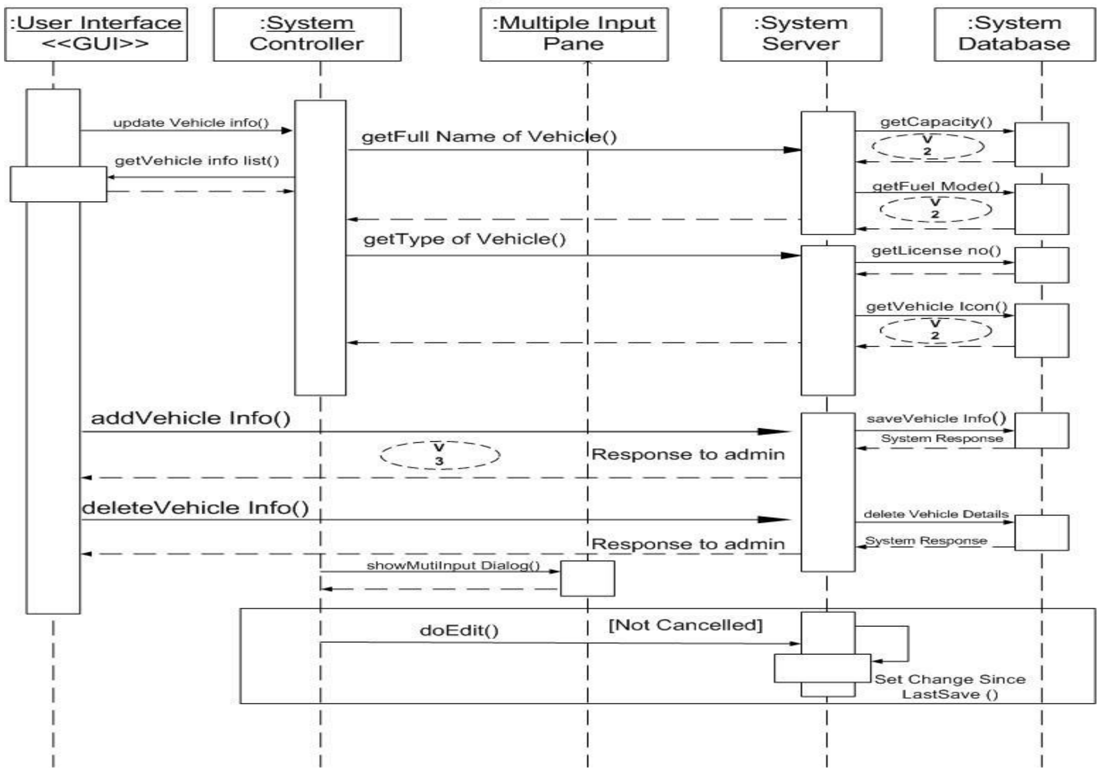


Figure 5: Sequence diagram representing Vulnerability

The sequence diagram of secure strategy design pattern shows the vulnerability in Figure 2. In the diagram points v3 and v2 describe the likely target point of highest level and moderate level of vulnerability respectively.

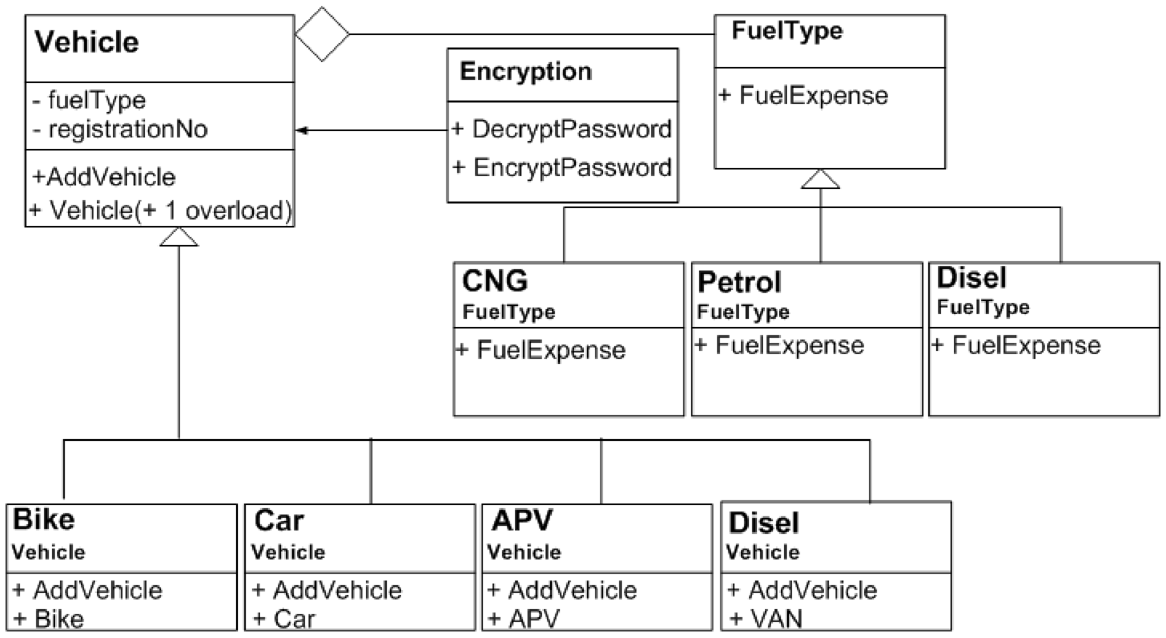
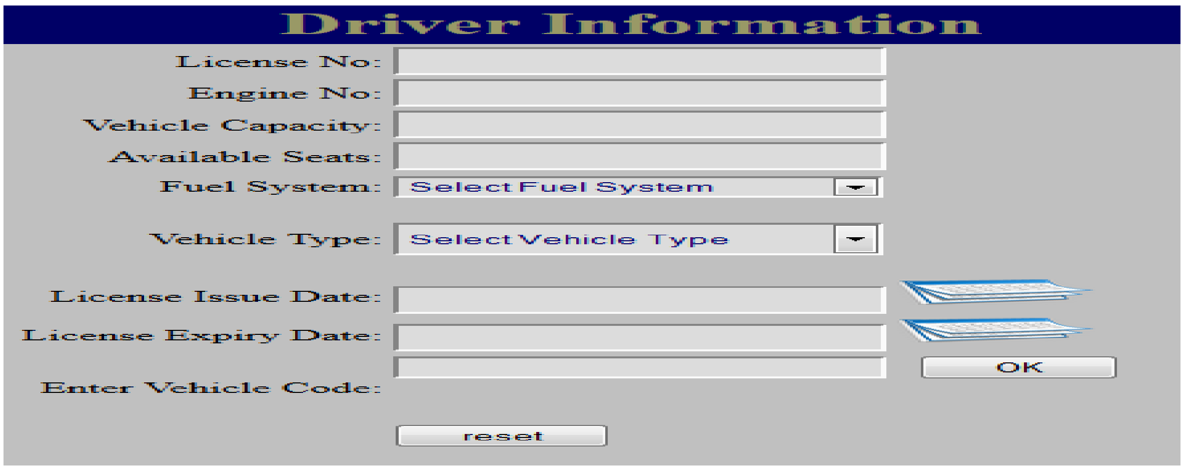


Figure 6: Class diagram of SSDP with proposed solution

The class diagram of driver information page for secure strategy design pattern shown in Figure 3. It illustrates the functionality of the web page; the subclasses of class vehicle are attached for the web functionality. The abstract classes of vehicle info are bike, care, van and APV, while Encryption/Decryption and Fuel Type class is aggregated with vehicle information. The Encryption/Decryption is an encryption and decryption scheme that is used for the authentication for selection of bike, car, van, APV etc.

First user registers itself on the website. After that it has to login to the website. If the person has its own vehicle then he has to enter driver information. The driver information page require the information as License No, Engine No, Vehicle capacity, Available sets, Fuel system, Vehicle type, License issue date, License expiry date and at the end website ask for the required code that is for security purpose. If the user enter correct code, which admin of this website has allotted to user ,then the option submit appear and user can submit information otherwise he would be ask 2 times more to enter valid code otherwise user will be logout from the website and all sessions will be destroyed. In fact, system will consider such person as hacker who will try to enter information by entering wrong passwords. Invalid user cannot use this site any more due to session destruction. At this point if such hacker tries to snip password then such hacker will again remain unsuccessful because, security passwords will be secured due to usage of encryption/decryption technique. After valid data entry it first shows the price that owner of the vehicle bear the expected cost for his vehicle. The strategy design pattern is used to develop this page shows that when we change the type of vehicle the result in cost that have to bear by vehicle owner changes. The view of driver information page before data entry is given in Figure 7.


Driver Information

License No:	
Engine No:	
Vehicle Capacity:	
Available Seats:	
Fuel System:	Select Fuel System
Vehicle Type:	Select Vehicle Type
License Issue Date:	
License Expiry Date:	
Enter Vehicle Code:	

Figure 8 Driver Information page without data entry

2.4 Secure Builder Design Pattern

First of all, the user of this system, request the system to submit the Route Information of his route which is used for office, weekend or official trip. System asks valid password of required route. If a user enters enter valid password, which admin of website allotted to him, system allows user to submit route information. Otherwise system gives two more chances to user for entering valid code. If user failed to enter valid code, then system considers such user as attacker and logout such user. System destroys all session information of that user. Therefore, attacker remains unable to harms such system. If attacker attacks on system's database, to steal password for route information. Then such attacker again failed to enter required password because password stored in the database is in encrypted form and user will not be able to decrypt such password because he does not know which password technique of encryption and decryption is used in this system. This encryption or decryption technique will save system. Use case diagram for Secure Design Pattern is shown in Figure 9.

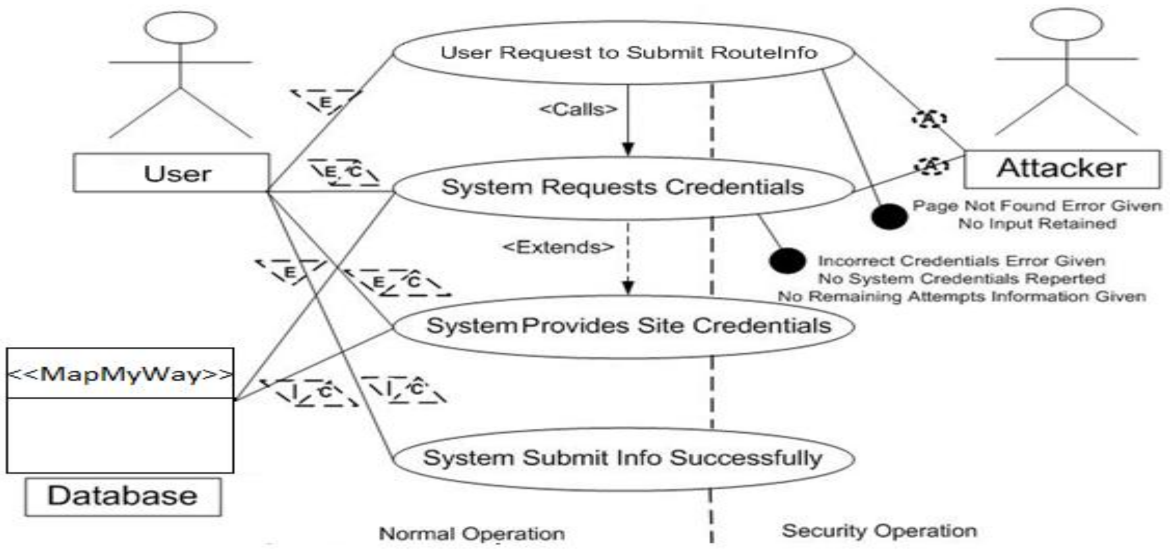

User
Attacker
User Request to Submit RouteInfo
<Calls>
System Requests Credentials
<Extends>
System Provides Site Credentials
System Submit Info Successfully
<<MapMyWay>>
Database
Page Not Found Error Given
No Input Retained
Incorrect Credentials Error Given
No System Credentials Reperted
No Remaining Attempts Information Given
Normal Operation
Security Operation

Figure 10: Use case diagram representing vulnerability

The sequence diagram of SBDP shows the vulnerability in Figure 6. In the diagram points v3 and v2 describe the likely target point of highest level and moderate level of vulnerability respectively. The class diagram of route information page for SBDP shown in Figure 7.

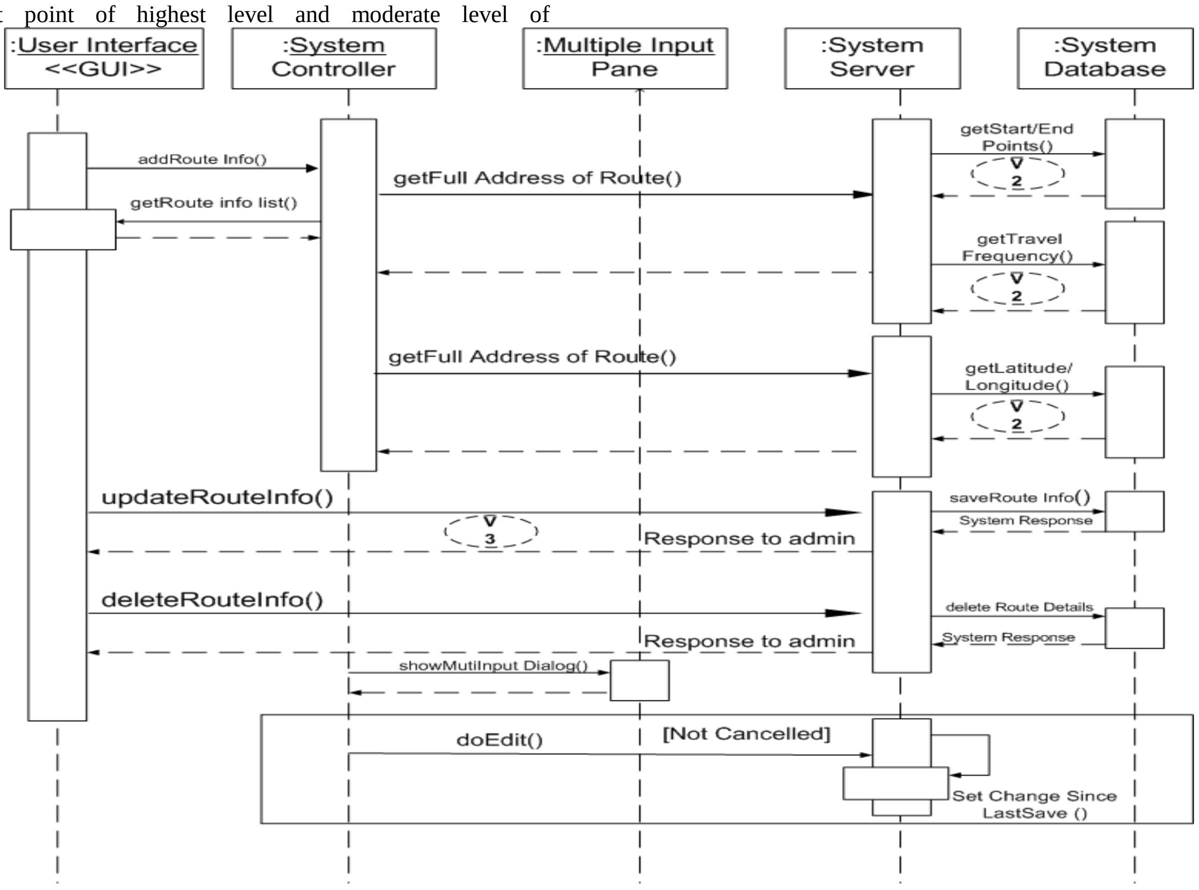


Figure 11: Sequence diagram representing Vulnerability

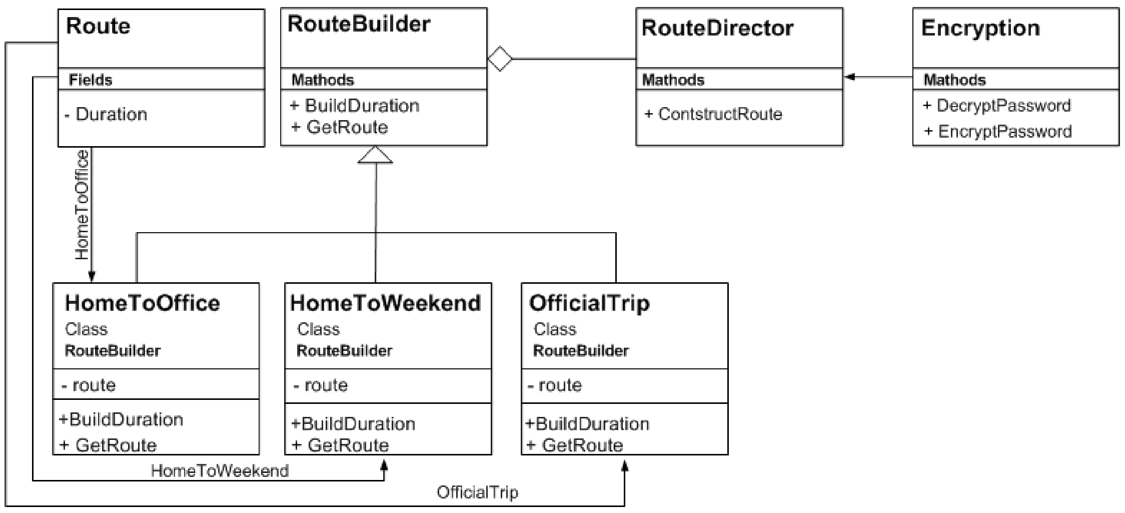
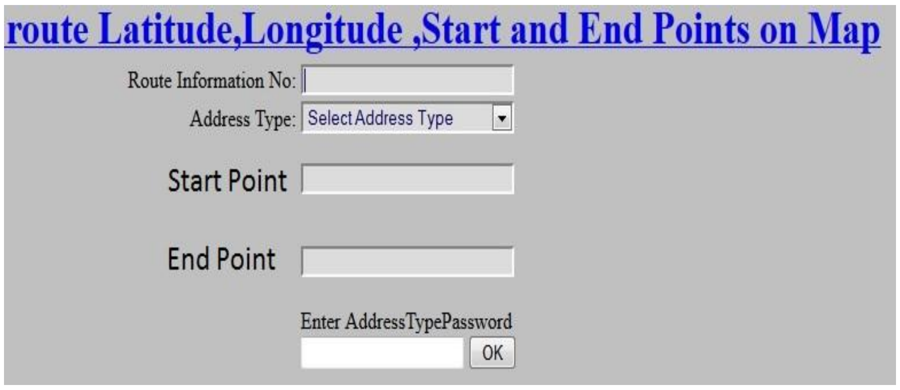


Figure 12: Class diagram of SBDP with proposed solution

It illustrates the functionality of the web page; the subclasses of class RouteBuilder are attached for the web functionality. The derived classes of RouteBuilder are HomeToOffice, HomeToWeekend, and OfficialTrip, while Encryption/Decryption is associated with RouteDirector. RouteDirector class is aggregated with RouteBuilder class. The Encryption/Decryption is an encryption and decryption scheme that is used for the authentication for selection of HomeToOffice, HomeToWeekend, or OfficialTrip etc. The display of route information page before data entry is presented in Figure 13.


route Latitude,Longitude ,Start and End Points on Map

Route Information No:
Address Type: Select Address Type
Start Point
End Point
Enter AddressTypePassword
OK

Figure 14 : Route Information without data entry

3 CONCLUSION & FUTURE WORK

3.1 Conclusion

As the developers during application development just focus on extendibility and neglect software design result in the application failure due to compromise on vulnerability. The applications developed randomly by developers contain more vulnerable points than the application developed by using classes and security class. The method proposed by us makes application more secure against vulnerabilities. Moreover, usage of design pattern reduces vulnerable points than ordinary development method. Therefore, we developed this application by using secure software design used encryption/decryption technique SHA-1 for the security purpose. DP's are used to develop the application and encryption/decryption class is associated with DP to make it secure design pattern. The application now shows highly secure and proves that if developer foster websites by using such design patterns in association with encryption/decryption technique then security risk may be diminished. This encryption/decryption technique used in this website by applying security class may also implement on other design patterns. Hence, we conclude that security class used in this website for security reasons proved that our website is highly secure from attackers and this method is useful for other patterns to make secure them.

3.2 Future Work

The SSDP and SBDP are two techniques proposed in this paper are the result on the working of Strategy Design Pattern and Builder Design Pattern[4]. In future, we plan to expand this security feature to other design patterns to make them secure form attackers and vulnerable points at the application. As for as this web site concern, we plan to add more pages and features developed by using other securities patterns and apply our security technique to make them secure form attackers and reduce vulnerable points. There are many encryption or decryption techniques like SHA-1, SHA-256 and SHA-512 etc. We also have planned to develop other applications by using these or new encryption/decryption technique with newly developed algorithm for more security. There are also many security techniques like security questions and matching images. We planned to use such techniques to make our website more secure.

4 REFERENCES

[1] T. Richardson and C. N. Thies, *Secure software design*: Jones & Bartlett Publishers, (**2012**).

[2] C. Alexander, S. Ishikawa, and M. Silverstein, "*A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series*)," (**1977**).

[3] Beck, Kent, Cunningham, and Ward, "*Using Pattern Languages for Object-Oriented Programs, Design Methodology for Object-Oriented Programming, Panel Session*, OOPSLA," *ACM, (***1987**).

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: elements of reusable object-oriented software: Pearson Education, (**1994**).

[5] K. Lano, "*Design patterns: applications and open issues*," in *Cyberpatterns*, ed: Springer, (**2014**), pp. 37-45.

[6] E. Fernandez, M. Larrondo-Petrie, T. Sorgente, and M. VanHilst, "*Layers and non-functional patterns*," *Procs of ChiliPLoP, Phoenix,* vol. 1, pp. 10-15, (**2003**).

[7] Fernandez, E. B, and X. Yuan, "*Semantic analysis patterns*," in *Conceptual Modeling—ER 2000*, ed: Springer, (**2000**), pp. 183-195.

[8] Fernandez, E. B, and J. Hawkins, "*Determining role rights from use cases*," in *Proceedings of the second ACM workshop on Role-based access control*, (**1997**), pp. 121-125.

[9] C. Alexander, *Notes of the Synthesis of Form* vol. 5: Harvard University Press, (**1964**).

[10] Hoglund, Greg, McGraw, and Gary, "Exploiting Software: How to Break Code," *Addison Wesley, (***2004**).

[11] L. Rising, "Understanding the power of abstraction in patterns," *Software, IEEE,* vol. 24, pp. 46-51, (**2007**).

[12] D. C. Schmidt, "*Using design patterns to develop reusable object-oriented communication software*," *Communications of the ACM,* vol. 38, pp. 65-74, (**1995**).

[13] C. R. Dougherty, K. Sayre, R. Seacord, D. Svoboda, and K. Togashi, "*Secure design patterns*," *Software Engineering Institute, (***2009**).