Design Pattern and Security Influence on Application Development

Zia Ahmad
National Textile University
Faisalabad

Adeel Rauf
Virtual University of Pakistan

Mian Ali Asghar
Global Institute Lahore, Lahore

ABSTRACT
It's common among developers when they start coding for an application, a specific design pattern not found in the road map of development and it becomes a vulnerable point to exploit. Applications developed without design pattern difficult to change and understand. It is possible to reduce vulnerability at minimum level and it results in the reduction of maintenance cost. An application was developed for this paper using design patterns, two pages visitor information page and school member verification form were build using Factory design pattern and Interpreter design patter. SFDP and SIDP are the two secured design patterns proposed for making application secure and more reliable than before using encryption-decryption hashing algorithm encoding scheme. The points expressed in proposed model clearly explain the expected vulnerable points. A secure design will keep application more reliable and available as it was before.

Keywords
Software design patterns; vulnerability; security pattern; refactoring; secure software design; secure pattern.

1 INTRODUCTION
Internet users are growing day by day and due to this rapid increase sensitive information becomes insecure because some people use this service for negative purposes. There is consistently billions of dollars loss when an application compromised due to a vulnerable point in an application. Hackers build things, crackers break them[1]. Crackers, consistently trying to find weak points from the application where security is compromised. These points are actually vulnerable in the application. Attackers cause problems by exploiting vulnerabilities in code[1]. Vulnerability is also known as a fault in the security of an information system that some time may be known or unknown[2]. Vulnerability is simply a design flaw or an implementation bug that allows a potential attack on the software in some way[1]. A threat is a possible exploit of vulnerability where an attack is the actual use of such an exploit[1]. Some of the most serious and well-explored vulnerabilities and corresponding threats include the following characteristics: a) Lack of input validation, b) insecure configuration management, c) Lack of bounds checking on arrays and buffers and Unintentional disclosure[1]. This definitely becomes the reason of application to be compromised. Attackers are more inventive by making more complex attacks.
Malicious website steals important credentials from the victim or installs malware on the victim's machine to use it as a spring board for further exploits[3]. It is therefore, becomes vital to invent new technologies for counter and detect attacks on applications. There are too many limitations in the present available techniques for removing vulnerable points. It is not only the part of coding but design issue must be considered in mind while constructing sketch of application after gathering requirements. Antiviruses, and firewalls are used for blocking and removing attacks but this is not sufficient in this modern emerging world of World Wide Web. Today thousands of developers are writing code for multiple applications. While coding they do not think about vulnerable points in the application so they are compromised.
Web applications are used by many people via internet services. Wide varieties of web service are available on the internet with the expansion of computer networks. This variety of web application provides services like information searching, online commerce and social network services. The activities that support people become intertwined with the Internet, vulnerable applications are progressively attractive targets for malicious attacks and data theft. Several user authentication methods are used to avoid misuse or illegal use of highly sensitive data, therefore cyber security is a vital issue to address. The institutes and organizations increase utilization of security technology and regularly train security professionals. The criminals are working on research and development of latest cracking methods that may be helpful to steal money and valuable information. An error of software security is the exact cause of vulnerability.

It is essential to eliminate compromised point to prevent an attack when vulnerability is detected. Most of the developers while developing did not think about the pros & cons of the programming and they continuously write code so this is the real reason of vulnerability, even the developer did not know about the compromised point and it becomes point of attraction for a cracker. Software cannot be made completely secure because it is ongoing battle between developers and crackers. The development technology is progressing towards security as well as crackers are becoming powerful in finding weaknesses in development and security is yet in danger. The malicious software or malware which attempts to harm system or software's always trying to find weaknesses and compromised security measures from the system such as passwords and cryptographic data. Another reason of threat agent is human unintentional, ruptured security through carelessness or accidental, a backdoor in software caused by poor coding.

Right people and the right tools can develop software efficiently, rapidly with rework and reuse. A comprehensive development process must be able to acclimate change according to emerging technology and business needs for this system completion consistently and predictable.

1.1 Design Pattern History
- 1977/79 – Architect Christopher Alexander introduced the concept of design patterns with respect to the design of buildings and towns[4].
- 1987 – Beck and Cunningham experimented with applying patterns to programming and presented at OOPSLA[5].
- 1994/95 – The "Gang of Four" (Erich Gamma, Richard Helm, Ralph Johnson, and John M.

Vlissides) published a book containing a large number of design-level patterns aimed at object oriented programming languages[6].

- 1997 – Yoder and Baraclow published a paper outlining several security patterns[7].
- 2009 – Chad Dougherty et al. published technical report on Secure Design Pattern[8].
- 2011 – Ko, Andrew J., et al. "The state of the art in end-user software engineering." [9].
- 2014 – K. Lano work on Design Patterns: Applications and Open Issues[10].
- 2015 – Zia Ahmad et al. worked on Implementation of Secure Software Design and their impact on Application[2].

1.2 Important Terms

Software: A set of instructions provided to the computer to achieve a specific task. *SDLC*: It is a conceptual model used in project management and it defines steps elaborated in an information system development. *Use case*: A model to perform a specific task describes how end user interacts with system. *Model*: A model is a simplification of reality it provides blueprints of a system[11].

Secure Software: Software is secure when under malicious attack if it can provide certain operational features. Confidentiality, only authorized people can get access. Integrity, The data that is presented is unchanged. Availability, The system and its data is available even under hostile conditions. Authenticity, Users is who they claim to be.

Programming: English dictionaries states that the process of planning or writing a program. *Unified Modeling Language (UML)*: is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system[11, 12].

Design Processes: They constrain how requirements are translated into design specifications and then implementations[9]. *Pattern*: A pattern is a general reusable solution to a commonly occurring problem in design[8].

Design Pattern (DP): They provide solutions to common software design problems, in object-oriented programming, design patterns are generally aimed at solving the problems of object generation and interaction, rather than the larger scale problems of overall software architecture[6]. DP provides solutions of real-world problems.

Strategy Design Pattern: or policy pattern selects algorithm's behavior at runtime. It defines a family of algorithms, encapsulate algorithms and make them interchangeable within that family.

Builder Design Pattern: It is an object creation design pattern.

Factory Design Pattern: an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclass[6].

Hackers: A person who is proficient with computer, particularly in the field of network, and they were the foundation of what has led to the technology revolution in which we all find ourselves[1].

Crackers: People who wish to use technical proficiency to exploit weakness and break systems in almost every case[1].

Encryption: In it a single key is used to run an algorithm to transform plain-text to cipher-text[1]. *Decryption*: The same key is used to transform data back from cipher-text to plain-text[1]. *Cipher-text*: Information that is encrypted[1].

Zero-day vulnerability: is a hole in software exploited by the cracker before the developer aware of it. *Zero-day attack*: An attack that is previously unknown depending on the scale and target[1].

Zero-day threat: A known vulnerability that has yet to be exploited but which has not been mitigated[1]. *Zombie*: A machine that is compromised and works on the instructions of some other user instead of its appropriate user.

1.3 Related Work

The use of Design Patterns also encourages design exploration and experimentation[13]. Computer programming is becoming a practical skill of millions from first computer program in 1940's to today's speedily rising software business. Today programming is not only done by professionals but people with some knowledge about it. Software designing is too difficult, but secure software designing is more than difficult. For a design problem there may be many solutions but it is difficult to decide which one is the best and is able to reuse for future design. Each solution having pros and cons it is the reason to select best existing design solution. It is up to the non-functional requirements and goals of the application to choose best available solution. The requirements are gathered in first phases of SDLC so they specify the functionality of the application, qualities (non-functional requirements) and goals of stakeholders. The pattern methods have been adapted and tested in various disciplines including the field of object-oriented design (software development)[13].

A design pattern cannot be transformed directly into a source code or machine code because it is not a finished design. DP is a guide line, template or description to solve a problem that might adopted in various situations. During designing an application patterns are best practices used to solve common problems. Object-oriented DP represents relationships and interactions between classes / objects. Web pages in web applications are commonly described using computer languages such as HTML code and JavaScript[14]. In this work including both of these applications ASP.NET, C#, 'Microsoft® SQL Server is also used. In classical design patterns: (Behavioural) Template, Observer, Strategy, (Structural) Facade, Adapter, (Creational) Singleton, Builder[10]. The pattern has potentially negative implications for efficiency[10]. It's a common problem with many design patterns, improvements in the logical structure of a system may reduce efficiency[10].

The attack surface on a web application is typically much larger than thick client application and standalone system[1]. To protect a web application a practical application was developed that showed a path to protect an application to be compromised. To reduce accidental addition of vulnerability Secure Design Pattern (SDP) are used. Secure Strategy Design Pattern (SSDP) and Secure Builder Design Pattern (SBDP) were introduced and implemented in a web application[2]. SSDP is actually addition in working of Strategy design pattern and SBDP is extension in Builder design pattern. The working in these design patterns is to show a way for programmers to keep in view design structure when working on an application.

1.4 Critical Analysis

The nature of software has changed rapidly toward complex, often distributed and rapidly evolving systems[15]. The number of errors and vulnerabilities can be controlled if a secure software development process is adapted. Having broader scope in nature Architectural patterns (AP) are similar to DP. They address various issues in software engineering, like hardware performance limitations, reduction of a business risk, high availability etc. Some AP applied within software frameworks. The same pattern may be implemented by a number of different architectures that share related features. A concept that solves and represents necessary elements of software architecture is AP. An AP conveys an image of a system not architecture. Examples of Architectural-Level Patterns are a) Dis-trustful Decomposition b) PrivSep (Privilege Separation) c) Defer to Kernel etc.

Design pattern (DP) better explains object-oriented programming (OOP). By using design pattern object-oriented programming can easily be learned. DP made development easier than before because by following patterns coding is much easier. DP's are much easier to secure than AP's. The security class when attached with a set of classes called pattern, it becomes secure as it was free in past. The security class keeps some strong algorithm according to the demand and requirement of security. Secure Hashing Algorithm can be used for this purpose. There are SHA-1, SHA3, SHA-256, and SHA-512 etc. These encoding schemes encrypt data and compare with the code stored in the database if it matches then they allow to work or otherwise they close or sign-out or break session. Examples of DP are Strategy design pattern, Builder design pattern, Factory design pattern, Façade design pattern, Adapter design pattern, Decorator design pattern, Observer design pattern etc.

2 PROPOSED MODEL

2.1 Problem Statement

Progress in software development is a fact of life. The researchers have suggested hypotheses how software change over many years. After the first version deployment software's evolution starts. The developers progressively working on availability, reliability and reusability but ignoring design of application frequently. It is the main problem which became reason of a weak point that is being compromised and may provide imaginable attraction for attackers. One of the biggest issues that are compromised is software design issue and it attracts as a comfortable point for vulnerability. As a result organizations and developers have to pay more for removing the system vulnerability and its concomitant risk after the system installation. After understanding sources of security flaws it indicates to the importance of taking security in Software_Development_life_cycle (SDLC). Now a day's best available security techniques focus on implementation and deployment issues but not at all phases.

2.2 Model

No software is put to use without someone thinking about what it should do and how it should do it[15]. For developing good software the central part of activities is modeling. It is necessary that the software must be developed in a secure environment from its initial formation. To communicate behavior and structure of a system models are constructed. They control and monitor architecture of system and manage risks. A database developer focuses on ER-model and thinks about procedures and triggers. A structured analyst thinks about algorithmic technique and data flow of processes. A wild developer thinks about its goal and rapidly develops his system. An object-oriented developer works with classes and patterns and made the relationship of classes. Different approaches can be used for developing a system depending upon the system requirements but experience tells us that object-oriented view is best in coding for all types of systems.

The application SIT (Step-in-Security) is a web application which is developed for an organization which provides residential facility to its staff. All member data have to be collected because of high security risk. It keeps record of all staff members and visitors who visits staff members as guest. It is necessary to keep record of vehicle in application. The organization offers school facility to children of staff members and children of outsiders. So, due to security issue all children record form outside colony is also maintained in this application. All staff members, visitors, vehicle and students from outside also provide passes for entrance and exit form colony. This application is very useful in current situation in which terrorism activities occurs randomly. By using this application colony can be saved form expected terrorism.

2.3 Secure Factory Design Pattern

An application SIT (Step-in-Security) was developed by using design pattern. In this application a page used for keeping record of a visitor was made, with the help of factory design pattern. Factory design pattern was used to develop this page and it became secure factory design pattern (SFDP). It was made secure with used methodology and it is now known as secure factory design pattern (SFDP). In this page guest's proper record is entered who is coming to visit resident of colony. Guest's identity is entered into web page e.g. CNIC #, issue date, expiry date, date of birth, residence address etc. The guest is asked by the security officer about the night stay with the resident. If they say yes then they have to take special permission, after data entry a visitor pass is issued to the guest. The CNIC of the guest taken by the security officer will be returned to the guest when they depart from the colony. The use case diagram describes how to identify and mitigate vulnerability. The points represented by normal operation are internal working of the user with application. This is the valid way of working while the points shown in security operation are invalid ways adopted by the attacker. The points in use case represented by E/C and I/C are critical points that are always under consideration of attacker. These points are highly critical according to the vulnerability point of view. UML diagrams in this paper are used to show vulnerability; Figure 1 is use case diagram, Figure 2 is sequence diagram and Figure 3 is class diagram. The security officer enter required information on web page and the system require the security credentials at same spot attacker tries to gain access as shown in Figure 1. For an attacker these are vulnerable points. The working of the page starts from information (Host-Id, visitor-name, address, E-mail, mobile#, Gender and at the end type of visitor either night stay or day only) have to select and the security code from the security officer entered that is compared with the stored database encrypted code.

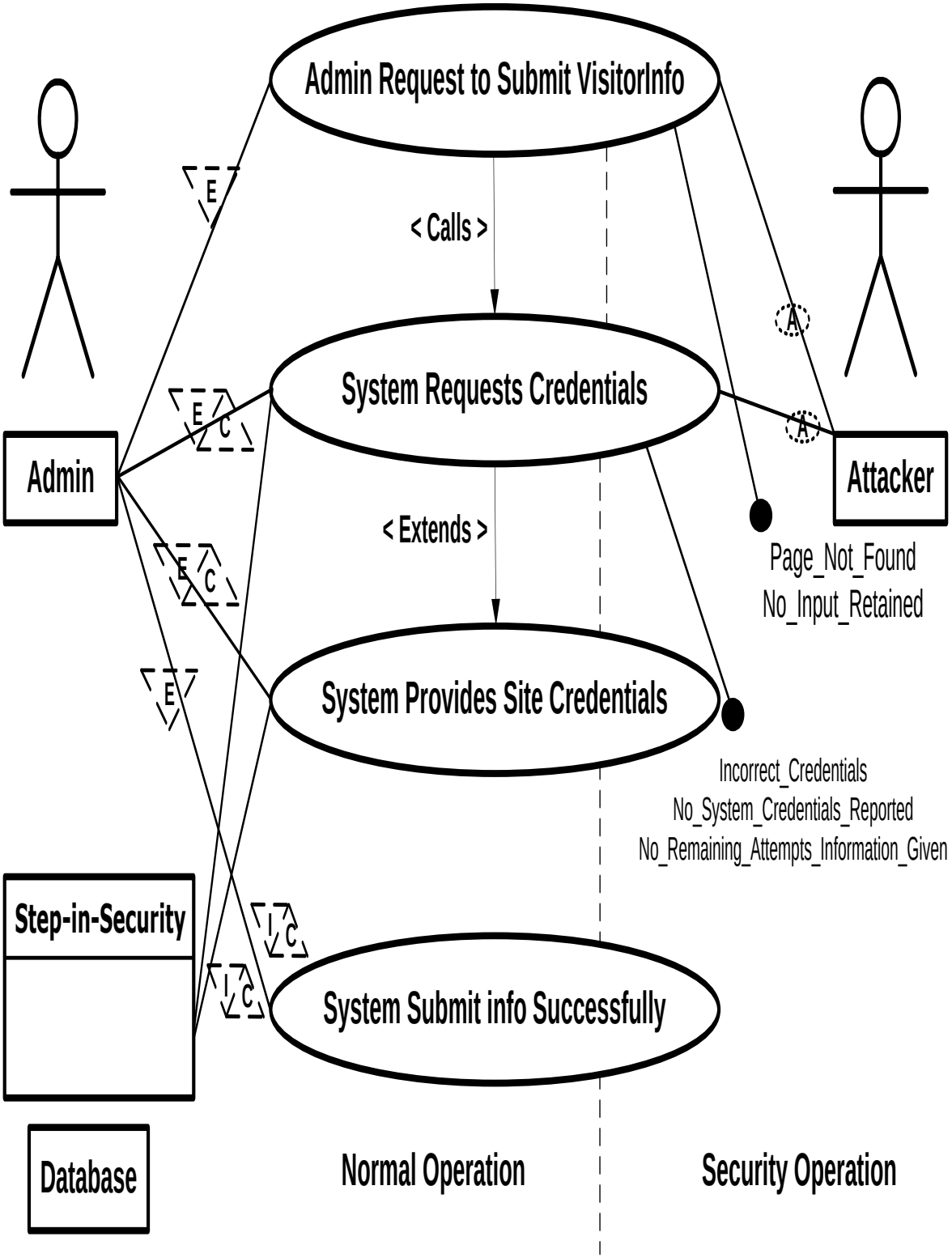

Figure 2-1:- Use Case Diagram with vulnerability (SFDS)

The matching security code with the database encrypted code will submit the information. If given code not matches with database encrypted code it requests the code 2 times more if correct code not provided the system will logout, extra attempts will not be provided to malicious user for using the website or again may have to log-in. 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].

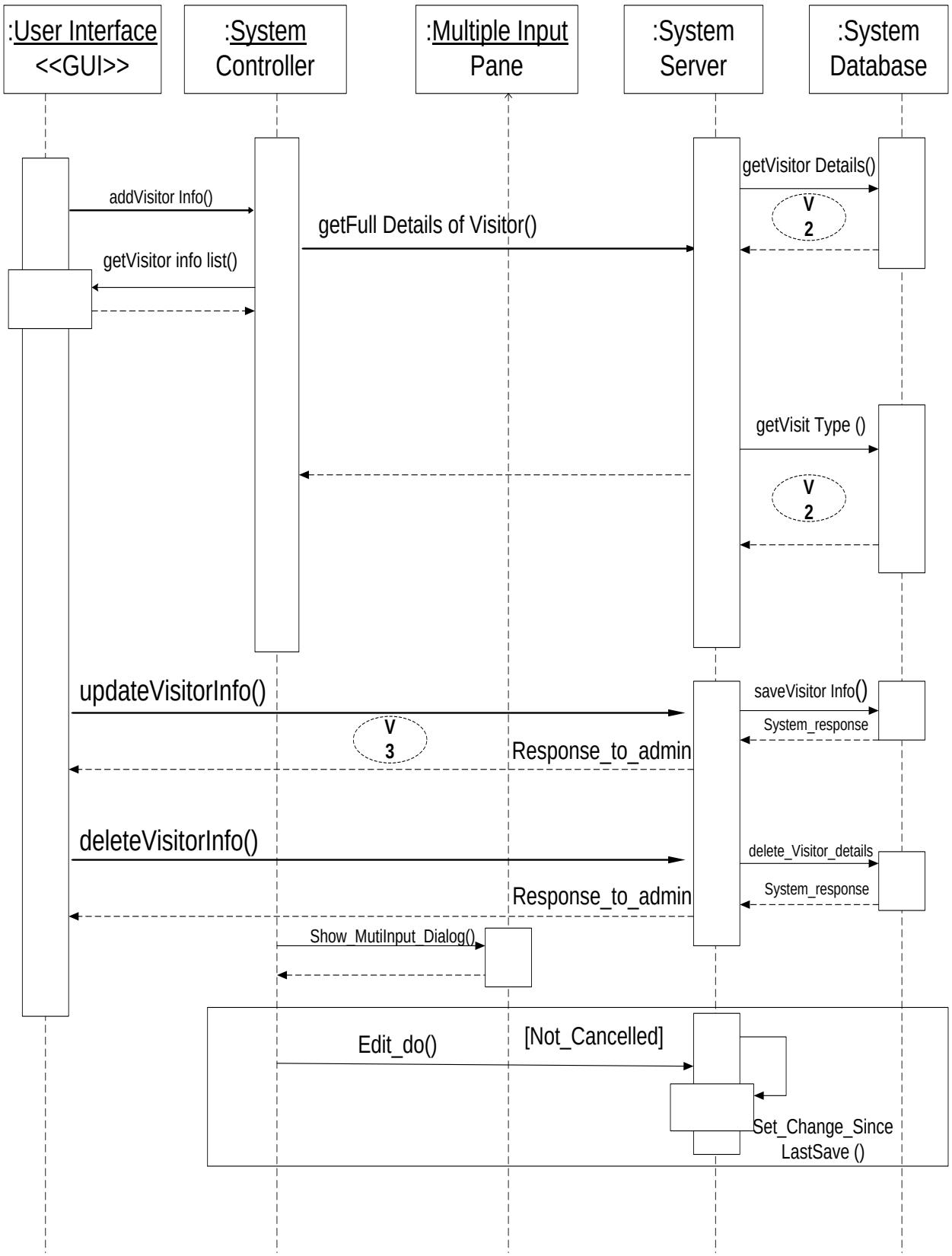

Figure 2-2: - Sequence Diagram with vulnerability

The sequence diagram of secure factory design pattern displays the vulnerability in Figure 2-2. In the diagram points v2 and v3 define the likely target point of moderate level and highest level of vulnerability respectively. These points must be covered while developing project. V3 must be finished but V2 can be compromised. There is another point V1 that can be neglected but with the change of requirements there is possibility of conversion V1 to V3 easily so this must be keep in mind.

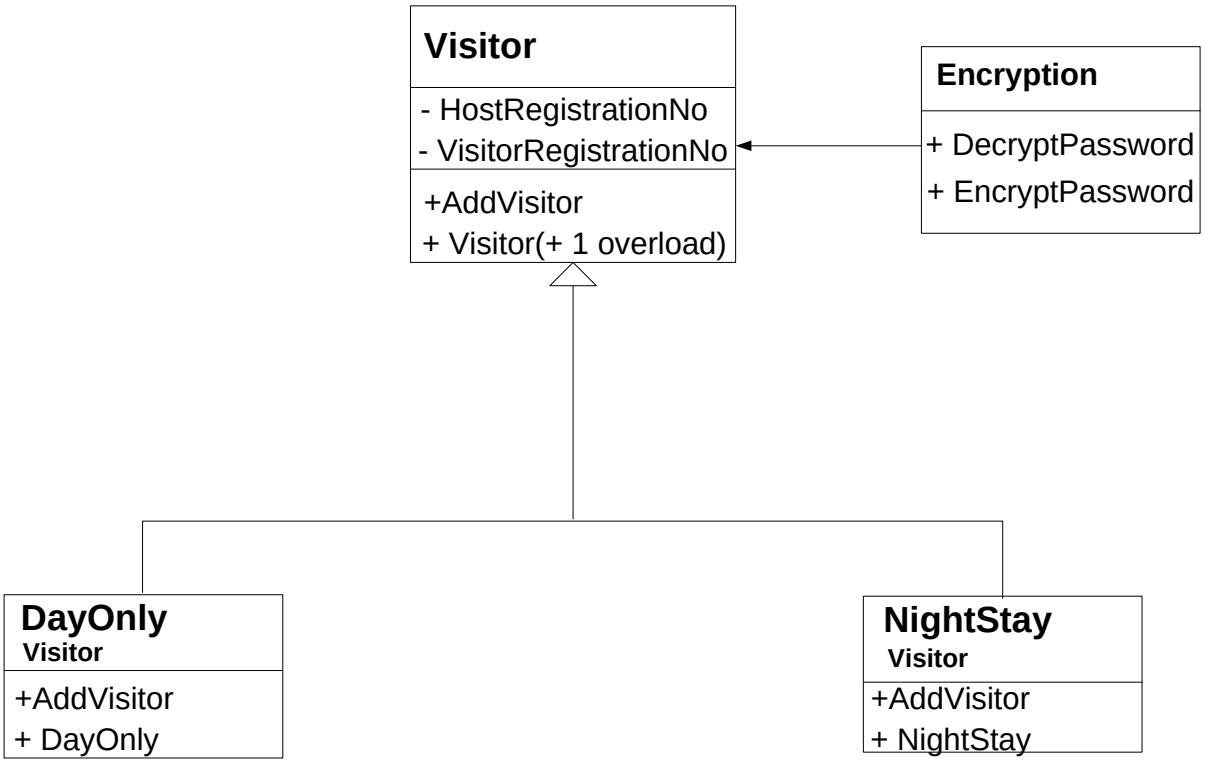

Figure 2-3: - Class Diagram of SFDP

The diagram 2-3 shows the working of application using to the Factory design pattern. Base class "Visitor" asks from its derived classes "DayOnly" and "NightStay" about its working after verification from "Encryption" class. The "DayOnly" and "NightStay" classes are inherited from the base class Visitor and the Encryption class is aggregated to Visitor class. The Encryption class keeps the codes encrypted and after decryption the compare each other to allow the usage of the application.

Figure 2-4: - Visitor Information Page

In fig 2-4 all of the information shown that must be entered to generate visitor pass by the authority. First of all host-Id # entered, after this name of visitor, address, contact #, gender and most value able thing visitor stay type selected, then visitor pass issued to the visitor. Here a security class is placed which verify the code belongs to the night and day stay. It firstly encrypts the code and then compares the code stored in database and after verification it allowed to save information otherwise it ask for correct code 2 times more, if correct code not provided by the security officer then session breaks and web application automatically log-off.

2.4 Secure Interpreter Design Pattern

The second page for this paper was taken from the same website SIT. This is named as school member verification page. First of all if any member wants to enter in colony school then he has to verify his identity from security officer at gate. The security officer will check their record by entering their member id. If it matches with the data base then he allowed go in side in else he is rejected. In the class diagram showed in figure 2-5 School-Member is the base class and Member-Type is abstract class Encryption class is aggregated with Member-Type while three classes are abstract classes from Member-Type named as Student, Teacher and Staff. All of these classed are assigned a code that is stored in data base after encryption. This code is in the knowledge on security officer. If authorized person using website try to use this page he have to provide correct code if it matches with the stored code in database then allowed to print permission pass otherwise 2 more times asked to enter correct password, it is provides correctly then task accomplished else session closed for specific period of time at that node.

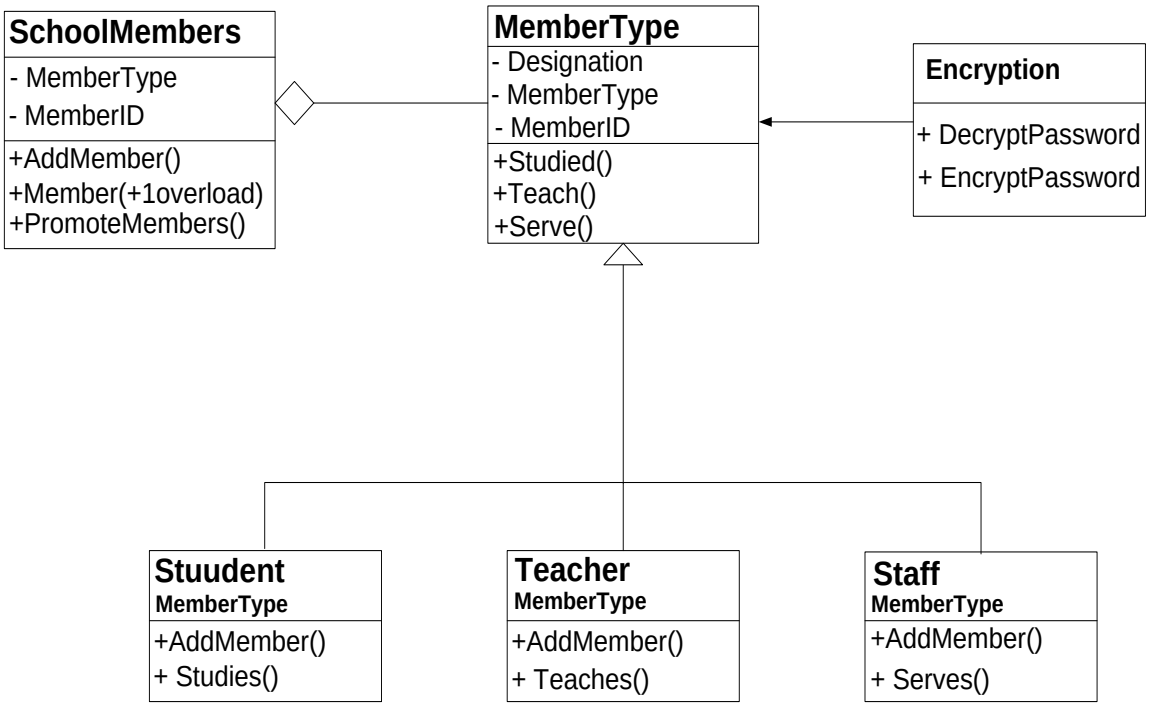

Figure 2-5 Class diagram of SIDP

The figure 2-6 is the working website page of the SIDP in this page first of all user have to select school member type from a dropdown list from student, teacher and staff. At second step he has to enter member type id and at the end its security code must be entered accordingly, if the given code found correct then permission letter will be provided to the member and allowed to go in.

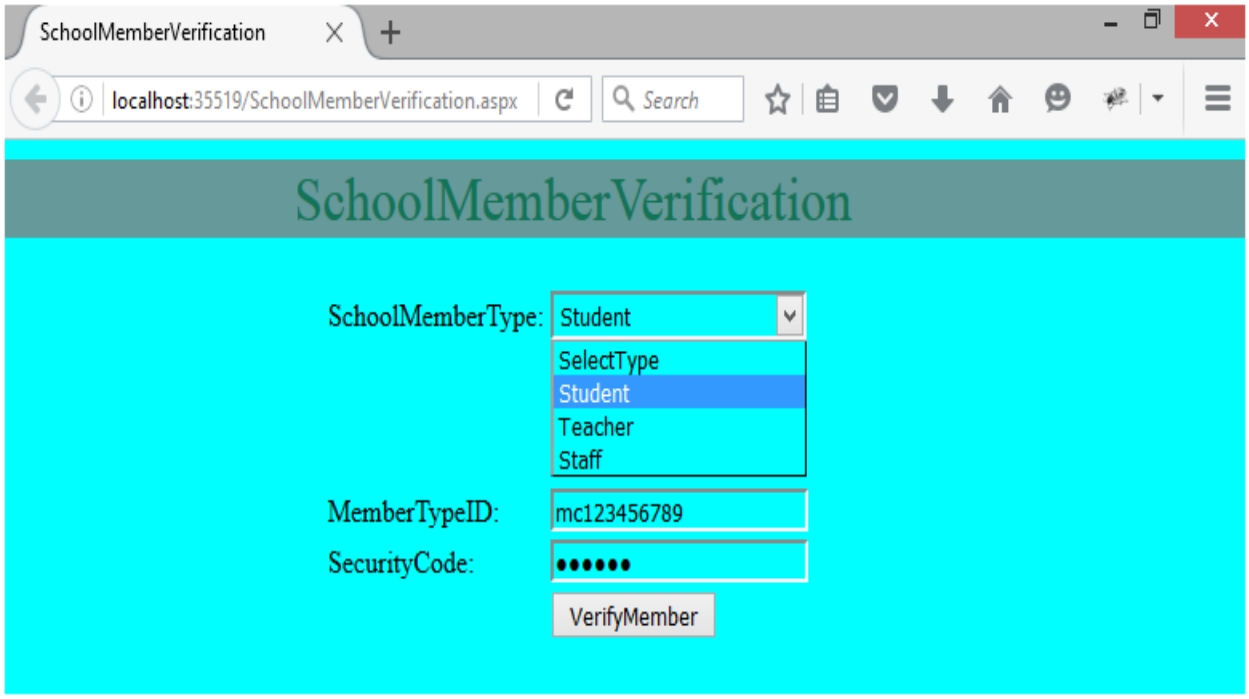

Figure 2-6 School member verification form

3 CONCLUSION & FUTURE WORK

3.1 Conclusion

The systematic inclusion of the Design Patterns approach to the learning strategy of programming in architecture and design proves to be highly beneficial[13]. The gains are more significant for software quality and therefore long-term reductions in software maintenance costs than the loss of optimal performance[10]. The identified security vulnerabilities are removed by revisiting and correcting requirements, design, and code[16]. A patch must have to be delivered if a copy of software is deployed after identifying the vulnerability. The security requirements should place separate, they are not interweaved with design and functional requirements. By using inheritance, DP improves the extensibility and adaptability of the system. DP increases designer's ability to reduce hurdles in programming and become helpful for architects to implement algorithmic design methods. Software design should also be considered along with extensibility, reliability and availability because it is valuable for reducing vulnerability. Usual development method provides vulnerable points as compare with design pattern and they became the cause of continuously increment in maintenance cost that must be taken at low.

3.2 Future Work

Two more design patterns 1) Factory Design Pattern & 2) Interpreter Design Pattern, used in this paper and proposed a way to secure them. Now they are Secure Factory Design Pattern (SFDP) &SIDP. In near future it is decided to practically launch both applications on the internet and make a comparison of applications by using secure design pattern and random coding. For this purpose a team of expert developers will be assigned a task of checking both types of application using secure design pattern/simply developed application thoroughly and try to find vulnerability. Moreover, there will be more work on SDP's. The coding techniques can be used are SHA-1, SHA-256 and SHA-512 etc. but an idea is there to introduce a new scheme or a new algorithm for security purpose that will be ease to use and more difficult to compromise. There is also an idea of working on architecture design patterns and make comparison on their security.

4 REFERENCES

[1] T. Richardson and C. Thies, *Secure Software Design*: Jones & Bartlett Publishers, 2012.

[2] Z. Ahmad, M. Asif, M. Shahid, and A. Rauf, "Implementation of Secure Software Design and their impact on Application," *International Journal of Computer Applications*, vol. 120, pp. 8-15, 2015.

[3] B. Eshete, A. Villafiorita, and K. Weldemariam, "Malicious website detection: Effectiveness and efficiency issues," in *SysSec Workshop (SysSec), 2011 First*, 2011, pp. 123-126.

[4] C. Alexander, S. Ishikawa, and M. Silverstein, *A pattern language: towns, buildings, construction* vol. 2: Oxford University Press, 1977.

[5] K. Beck and W. Cunningham, "Using pattern languages for object-oriented programs," 1987.

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

[7] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security," *Urbana*, vol. 51, p. 61801, 1998.

[8] Dougherty. Chad, Sayre. Kirk, Seacord. Robert, Svoboda. David, and Togashi. Kazuya, "Secure Design Patterns," *Software Engineering Institute, Carnegie Mellon University*, p. 118, 2009.

[9] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, *et al.*, "The state of the art in end-user software engineering," *ACM Computing Surveys (CSUR)*, vol. 43, p. 21, 2011.

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

[11] G. Booch, *The unified modeling language user guide*: Pearson Education India, 2005.

[12] L. S. Jimmy Wales, "Wikipedia, the free encyclopedia," *www.en.wikipedia.org*, January 15 2001.

[13] A. Globa, "Supporting the use of algorithmic design in architecture: An empirical study of reuse of design knowledge," 2015.

[14] Y.-T. Hou, Y. Chang, T. Chen, C.-S. Laih, and C.-M. Chen, "Malicious web content detection by machine learning," *Expert Systems with Applications*, vol. 37, pp. 55-60, 2010.

[15] A. Baker, A. van der Hoek, H. Ossher, and M. Petre, "Guest editors' introduction: studying professional software design," *Software, IEEE*, vol. 29, pp. 28-33, 2012.

[16] M. U. A. Khan and M. Zulkernine, "Activity and artifact views of a secure software development process," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, 2009, pp. 399-404.