CHAPTER VI

CASE STUDY: DEVELOPMENT OF ONLINE E-BUSINESS WEBSITE (PRICE) USING SAIFQT

6.1 Introduction

In this study, an online business field has been selected as the case study in question, due to the fact that this field is informative and it is full of tasks; that it has sensitive information and money, which must be implemented and has too much of security requirements to be achieved in the future software (website). There are four phases of SAIFQT to complete the eliciting software security requirements process and quantify the security requirements process; the steps are explained in the following sections through an actual case study.

6.2 Discovery Phase

In this phase, the developer discovers the current situation of the organization. The stakeholders will provide their experiences, skills and how they have been running the organization up to the present. They will disclose their success stories in the organization and how they have accomplished it. The basic questions devised by Cooperrider (2008) will be used to perform these assessments:

a) What were your hopes and dreams, when you chose this project?

Answer: To deliver an effective and secure system to the clients.

b) Based on your past experiences, what was the greatest experience you have had with a project, or when were you most successful and satisfied?

Answer: The trust by the customer and Stakeholders' participations is the greatest achievement. It gives us immense satisfaction, when we serve the customers with minimal effort and time.

c) Did you get any help from your friends/colleagues? Were you able to help them in return?

Answer: Yes, we did receive assistance from our friends and colleagues, and of course we reciprocated the help, whereby based on our experience, we were able to solve the problems of our colleagues.

d) Did you experience any unexpected incidents or face a difficult challenge? What did you learn from those?

Answer: Yes, every difficult challenge we face is our learning curve, above all it boosts our encouragement to face any other bigger challenge. For example (Real Case), when our system became malfunctioned by external hackers, paying for products (services) using stolen credit cards and there have been a few times when some hackers had succeeded in viewing our databases and exploiting our data. The repair system and getting back to normal method needs a lot of time, cost, effort and high margin of error, other than the fact that checking in this case is difficult.

e) What conditions have contributed to that extraordinary level of success and satisfaction?

Answer: The most important aspect is protecting sensitive data, which is not available in the old system. This feature must be carried out with high precision, which is very challenging in web applications. However, we are now very much relieved, as the new website can protect the data after each movement and update records and data, so we hope that the website will be secure and available 24 hours.

f) What do you value most about yourself and your capabilities, as a member of the team, or as a contributor of the project?

Answer: We can get involved and play a significant role in building the foundation of this website, by using our expertise and skills, to transform the weak functions towards the strong and protected functions by telling you (security experts and developers) what our problems are and what seems to be the problem in the old website. This will help us to get over the complications faced by us, especially subtracting the process of paying to get services, and ensuring the sustainability of the work, and being able to store user information safely again in tables; the new website will also help us to solve all the previous problems that we have suffered in the old website.

g) What do you value most about yourself as a member of the organization and/or member of the team?

Answer: "I value both, as a member in the organization I perform my duties to solve problems, and I provide the best services to our customers; and at the team-level, we focus on the software development, which is essential in putting things that are essential for the system".

h) What are the most important attributes that support your highest levels of success and satisfaction?

Answer: Perform all functions related to our website in a successful way, satisfy the customers with all the operations, save and protect all customer's and user's information and sustainability of the work in the website.

The above questions motivate the developers and security experts and give a big picture to the whole system; in addition they also give them the magnitude of the suffering, through the works accomplished by users every single day.

Output of Phase 1:

The outputs of this phase are: what are the security situations of the old system; what are the old software/business requirements that are needed in the future system, what

are the definitions agreed, what are the resources and the trust boundaries, what are the global security policy and what are the existing situation of the old system (Software Requirements and security requirements). Then, the answers of these questions as follow.

- Identify the existing situation of the old system (Software Requirements); System description (Stakeholders, Users and Developers). What are the weaknesses of the current system, what are the things that are not achieved by the current system and specify the operational environment:

    a) The system does not send a receipt to the customer.
    b) Margin of error is high.
    c) Checking the problem is difficult.

Agree on definition's step, enables the connections between security requirements' experts and stakeholders, by structured interviews or focus group, to agree on definitions using NIST and SANS standards (NIST, 8 September 2013; KLOCWORK, 12 November 2013), also this facilitates identifying security goals and the assets that must be protected, thus, it contributes to facilitate better elicitation of the security requirements in the future:

    a) Protecting sensitive data.
    b) Protecting the data after each movement and update records and data.
    c) Make the website available 24 hours (for sustainability of the work).
    d) Store user information safely in databases.
    e) Protect the website against:

        i. Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection').
        ii. Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection').
        iii. Buffer Copy without Checking the Size of Input ('Classic Buffer Overflow').

iv. Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting').
v. Missing Authentication for Critical Function.
vi. Missing Authorization.
vii. Use of Hard-coded Credentials.
viii. Missing Encryption of Sensitive Data.
ix. Unrestricted Upload of File of Dangerous Type.
x. Reliance on Untrusted Inputs in a Security Decision.
xi. Execution with Unnecessary Privileges.
xii. Cross-Site Request Forgery (CSRF).
xiii. Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal').
xiv. Download the Code Without Integrity Check.
xv. Incorrect Authorization.
xvi. Inclusion of Functionality from Untrusted Control Sphere.
xvii. Incorrect Permission Assignment for Critical Resource.
xviii. Use of a Potentially Dangerous Function.
xix. Use of a Broken or Risky Cryptographic Algorithm.
xx. Incorrect Calculation of Buffer Size.
xxi. Improper Restriction of Excessive Authentication Attempts.
xxii. URL Redirection to Untrusted Site ('Open Redirect').
xxiii. Uncontrolled Format String.
xxiv. Integer Overflow or Wraparound.
xxv. Use of a One-Way Hash without a Salt.

- Current system feedback and identifying security goals, which are achieved by the old system.

a) Sometimes databases are protected against hackers.
b) Payment process is encrypted.
c) The password contains at least 6 characters for users' login.

- It is guaranteed that, security requirements have the identical degree of ownership, as are all other prerequisites:

    a) The system was down by external hackers.
    b) Paying for many products (services) using stolen credit cards.
    c) Hackers are able to view databases and exploit the data successfully.
    d) Margin of error is high.
    e) Checking the problem is difficult.
    f) Process of paying to get services.

- Identify global security policy. The level of protection is suitable to be addressed, restrictions to which it is subjected, and the prospective effect on its reputation should an application be exploited:

    a) The password must contain at least 8 characters for normal users.
    b) The administrator must use the admin's ID as the username and the password must contain at least 10 characters.
    c) The customers must login the system by their usernames and passwords, which mean they must register to gain access to the system.
    d) The customer must have an account in the PayPal system and must have a valid credit card.
    e) Encrypt all passwords.
    f) Encrypt all databases' contents.

6.3 Dream Phase

This phase looks into what the user sees as the future of the system, specifically and across the organization as a whole. Here, the imagination and creativity of the stakeholder will be encouraged. The stakeholder will be able to contribute more effectively, since they have already completed the discovery phase. There should be no constraints in terms of the dreams of the stakeholder.

- Stakeholders, developers and security experts (team) dream (future system, i.e. what do they dream about the software requirements to be in the new system (what are the new business requirements they dream to achieve)).

a) What results do you dream/expect from a team or project?

Team answer: We dream/expect all the following in the new website: that service seller and customer can register into the system, administrator be able to view all the service seller and customer information and services, delete seller and customer accounts, administrator can be able to make suggestion messages to service seller and customer, allow service seller to be able to post services and edit/delete his services, system administrator can accept/reject the services, customer can view any posted service, customer can purchase any service and get a receipt after making a payment.

b) What do you envision as an ideal project in the future after many years?

Team answer: To be the most powerful and largest Online E-Business in the Market and to be trusted by all customers.

Output of Phase 2:

The outputs of this phase are: 'what might be' the system boundaries, assets, resources, besides the agreed security definitions, which are related to business or software requirements.

- Identify security needs/goals; what are the dreams to protect the system assets (future system). Definitions, candidate goals, business drivers, policies and procedures, where all these should be considered to make a concrete draft about 'what might be', to finish daily tasks using the proposed system, in secured and safe means by protecting all organization's assets and resources.

a) Security needs/goals: (Protecting sensitive data. Protect the data after each movement and update records and data. Store the user information safely in databases. Sub-step number five agrees with the definition's step in the discovery phase (25 SANS Vulnerabilities)).

b) Definitions:

   i. Availability of website 24 hours (sustainability of the work).
   ii. Encryption.
   iii. Addressing Denial Of Service Attack (DOS).
   iv. Addressing SQL Injection Attack.

c) Policies : (The password must contain at least 8 characters for normal users. The administrator must use the admin's ID as username and the password must contain at least 10 characters. The customers must login the system by their usernames and passwords which mean they must register to get access to the system. The customer must have an account in the PayPal system and must have a valid credit card. The system administrator accepts and allows services to be posted). The next phases will use the output of this phase to know "how can be and what will be".

6.4 Design Phase

This phase looks at how the system could be, where the user and the developer looks at the possible area of the size of the system. Using the identified strength that the stakeholder has highlighted, the size and the strength of the system can be now formulated by security experts. The role of the organization, relationship, policies and processes within the organization is observed from an outsider's viewpoint.

- Stakeholders and the developers examine the feasible size of the system, based on the recognized strengths highlighted by the stakeholders.

- Develop artifacts to support security requirements definition; the following relics are to be gathered: system diagram, use case scenarios/diagrams, misuse case scenarios/diagrams, sequence diagram, class diagram and standardized templates and forms. Consequently, the dimension and the strength of the system can be now derived. Consequently, the dimension and the strength of the system can be now derived (Draw Use Case Diagrams (Identify user roles and resource capabilities) and Misuse Case Diagrams; apply security principles to design (detail misuse cases, identify attack surface, and annotate class designs with security properties)); (Perform the security analysis of system requirements and design).

- Eliciting software and security requirements. as An essential factor in this phase is to make sure that the software/business requirements are proven, and that, they do not have, implementation or design restrictions, rather than requirements; thus elicitation is needed for all security requirements that are related to software requirements.

a) The role of the organizations, their companies and procedures are viewed in the external point of view.

b) As a matter of fact, it is not possible to analyze security into an application, therefore application testing and evaluations should remain a core element of an entire security technique. In particular, automated assessment tests can discover the security problems that are not recognized during code or implementation reviews, discover security threats unveiled by the operational environment, and act as a protection mechanism by finding downfalls in design, requirements, or execution. Typically, test and assessment functions are held by a test analyst, or by the quality assurance organization; however, it can be extended to the complete life cycle. The factors and elements that should be considered are:

i. Verify security attributes of resources.

ii. Perform source-level security review.

iii. Identify, implement, and perform security tests.

iv. Document security-relevant requirements.

v. Integrate the security analysis into the source management process: A vital objective of software security is to generate and sustain multiple-use source code, which fortifies the fundamental security services in software and over an organization's applications. This objective is most effectively accomplished, by applying secure development practices into an organization's general development process as soon as possible in the SDLC.

Output of Phase 3:

The outputs of this phase are: 'what should be' needed artifacts: scenarios, misuse cases diagrams, models, templates, threat modeling diagrams, reported security issues, forms, initial cut at security and software requirements are as follows:

Analysis and design are very important phases in the development stage of the research methodology and by using the requirement analysis techniques, requirements gathering techniques, security requirements techniques, requirement modeling and system requirements an attempt will be made to reduce the percentage of the failing rate of the system.

The Rational Rose software is based on the theory and the structure of UML, however UML is simply a tool to document system design. Rational advances step more and allow the developer to design software with additional business designs and components, as well as facilitates communication and organization of the project. These additional functions create an effective tool for enterprise level application. Following the aforementioned above, the requirements by identifying the actors of the system will be formulated.

6.4.1. Primary Actors

1. System Administrator: will be able to login into the system, view uploaded services, reject services, and accept services. Moreover, he can view accounts, delete accounts, suggest changes.
2. Service seller: will be able to make registration, login, post services, edit and delete services, purchase services, and make payment.
3. Customer: will be able to make registration, login, purchase services, and make payment.
4. PayPal: third party (external system) facilitates and secures the payment process.
5. Attacker: will be able to make brute force login, login system, make registration, disclose user information, and craft malicious code, purchase services, and make payment.

6.4.2. Business/User Requirement Determination

Determining the analysis of the requirements for any system is by identifying the existing systems, identifying the improvement and defining the requirements for the new systems. So, the requirement analysis technique has been adopted to explore how the old system performs the business process and then all the main functions needed are exploited and applied to the system. Therefore, phases one and two (discovery and dream phase) will be used to show a very high level of the business requirements researchers have managed to produce the main use cases that will be implemented to cover the requirements as mentioned in Table 18.

1. M: Mandatory 2. O: Optional & 3. D: Desirable Requirements

TABLE 18: The functional requirements of online e-business website

No	Requirement ID	Requirement Description	Priority
1	Login System	This function would enable the registered users (system administrator, service seller, and customer) to login into the system	M
2	Make Registration	This function would enable unregistered users (service seller and customer) to register into the system.	M
3	View Accounts	This Function will allow the system administrator to be able to view all the service sellers and customers' information and services	M
4	Delete Accounts	This function will allow the system administrator to be able to delete the service sellers and customers' accounts	M
5	Suggest Changes	This function will allow the system administrator to be able to make suggestion messages to service sellers and customers	M
6	Post Services	This function will allow service seller to be able to post services	M
7	Edit Services	This function will allow service seller to be able to edit his services	M
8	Delete Services	This function will allow service seller to be able to delete his services	D
9	View Uploaded Services	The function would enable the system administrator to view all uploaded services	M
10	Reject Services	The function would enable the system administrator to reject services	M
11	Accept Services	The function would enable the system administrator to accept services	M
12	View Admin Feedback	This function would allow both of the service seller and customer to be able to view the system administrator's suggestion message	D
13	View Posted Services	This function would allow the customer to be able to view any posted service	M
14	Purchase Services	This function would enable the customer to be able to purchase any service	M
15	Make Payment	This function would enable the customer who purchase any services to be able to make payment by PayPal to the service seller account	M

6.4.3. Use Case Specification

The use case specification is created to describe the interaction between the users and the system. The purpose of the use case specification is to explain the procedures, conditions and flow of events for each use case. The system has the least functions in the use case specification. (See appendix C) (Sequence and Collaborative diagrams).

FIGURE 19: Use case for Online E-Business Website.

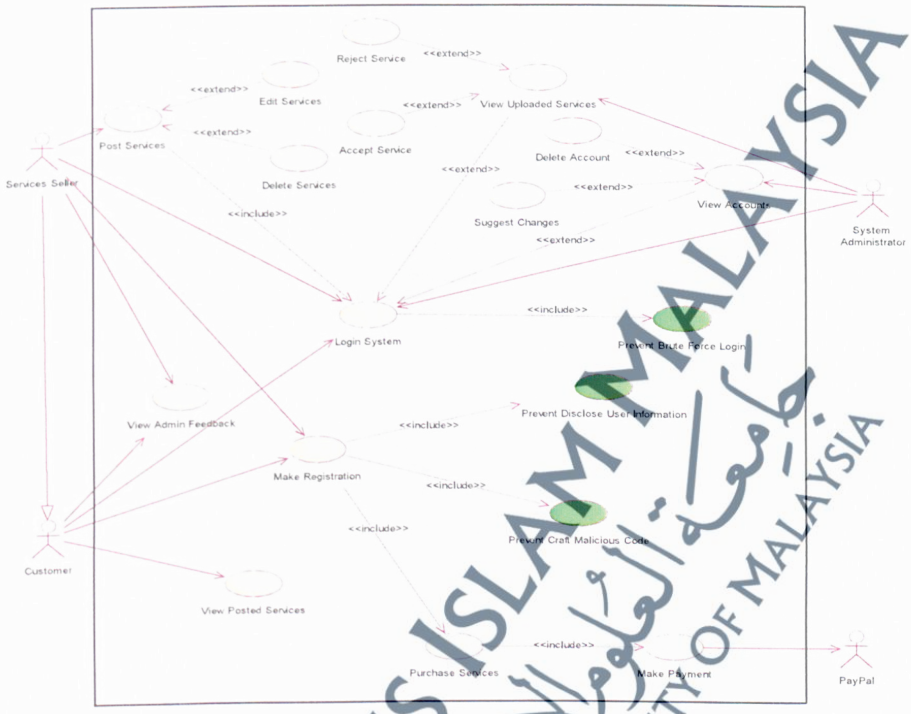


FIGURE 20: Misuse Case for Online E-Business Website.

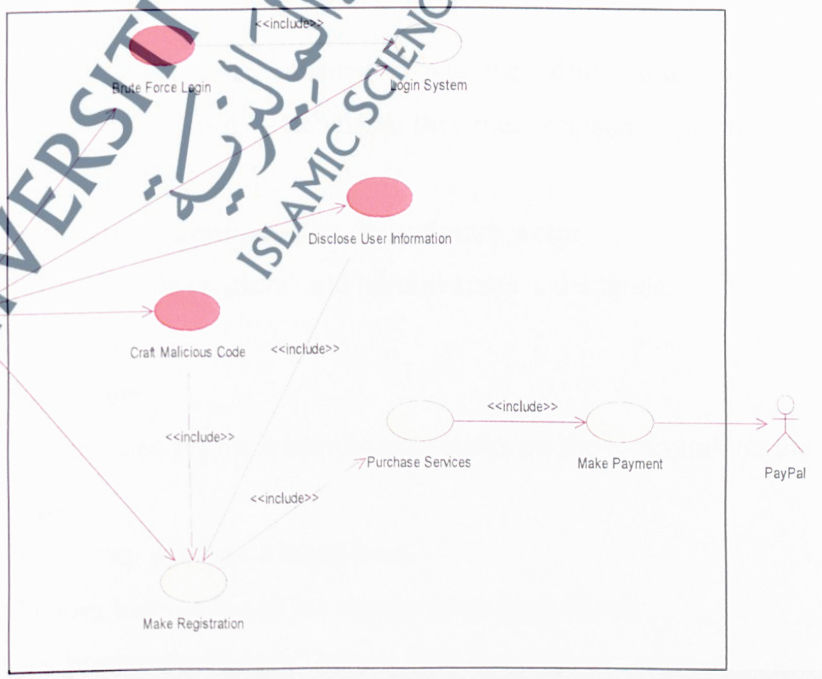
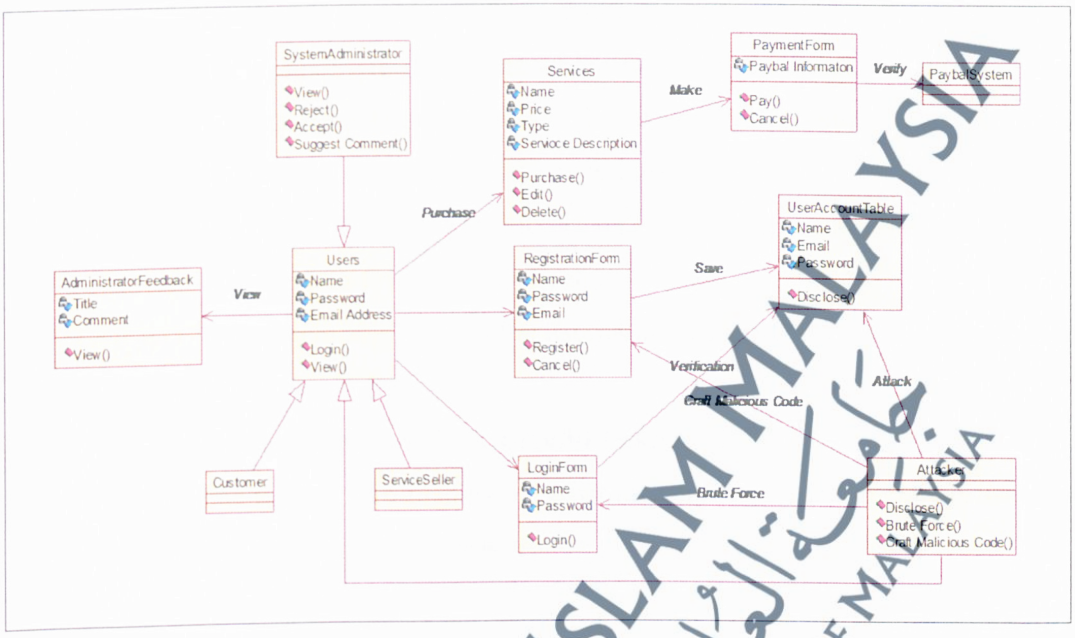


FIGURE 21: Class diagram for customer and attacker.



a. **Use case: login**

i. **Brief description**

This use case will be used to allow users (service seller, customer) and administrator to use the system.

ii. **Pre-conditions**

The users (service seller, customer) and the administrator must have a username and password, which mean they must register to get access to the system.

iii. **Characteristic of activation or the primary actor**

Execution depends on users' and administrator's demands.

iv. **Flow of events**

- **Basic flow**

This use case begins when the user clicks on the login link on the main page.

The system displays a login form.

The user keys-in his or her username and password.

User presses the "Login" button [A1: Cancel].

The system shall verify the information entered (username and password) [E-1: Invalid user name or password].

The system will determine which functions are available for users and administrators and display the specific workspace or the main homepage, depends on the entire information.

The user will be able to interact with their authorized functions.

- **Alternative flow(A1): Cancel**

The system shall cancel the username and password, and allow change in the user's data.

- **Exceptional flow(E1): Invalid Password**

The system will display an error message and the user has to re-enter the username and password.

v. **Post- conditions**

The users and administrator can enter the system.

Users and administrator will have different levels of privileges.

vi. **Limitation or Constraints**

The password must contain at least 8 characters for normal users.

The administrator must use the admin's ID as username and the password must contain at least 10 characters.

b. Use case: make registration

i. **Brief description**

This use case will be used to allow users (service seller, customer) to register into the system in order to use the system.

ii. **Pre-conditions**

The users (service seller, customer) should be able to access the website.

iii. **Characteristic of activation or the primary actor**

Users, all the users of the system will be able to make registration

iv. **Flow of events**

- **Basic flow**

This use case begins when the user clicks on the registration button on the main page.

The system displays the registration form.

The user keys-in username, password, and email address.

User presses "Register" button [A1: Cancel].

The system shall save the information entered (username, password, email address).

The system will show the "successfully registered" message.

The system will display the main homepage, that depends on the entire information.

The user will be able to interact with their authorized functions.

- **Alternative flow(A1): Cancel**

  The system shall cancel the username, password, and email address. Then it goes back to the main page.

v. **Post- conditions**

The users can enter the system.

Users have their own levels of privileges.

vi. **Limitation or Constraints**

The password must contain at least 8 characters.

c. Use case: purchase services

i. **Brief description**

This use case will be used to allow customers to be able to purchase any service.

ii. **Pre-conditions**

The customers must login the system using their usernames and passwords which mean that they must register to get access to the system.

iii. **Characteristic of activation or the primary actor**

Users

iv. **Flow of events**

- **Basic flow**

  This use case begins when the user clicks on the "order now" link in the service page.

The system displays "ordered details" page.

The user clicks on the "purchase now" button [A1: Select Additional Order], [A2: Contact Seller].

The system shall verify the information entered (username and password) [E-1: Register].

The system will display "payment details" page.

The user will be able to interact with their authorized functions.

- **Alternative flow(A1): Select Additional Order**

  The system shall allow the user to order more services.

- **Alternative flow(A2): Contact Seller**

  The system shall allow the user to contact seller by sending an email message.

- **Exceptional flow(E1): Register**

  The system displays a registration form and the user has to register.

v. **Post- conditions**

The users can enter the system.

The user has his own username, password, and email address (registered).

Users have their levels of privileges.

vi. **Limitation or Constraints**

The user has a username, password, and email address (registered).

d. Use case: make payment

i. **Brief description**

This use case will be used to allow customer to make payment to the corresponding service seller.

ii. **Pre-conditions**

The customer must have a username and password which means they must register to get access to the system.

The customer must have an account in the PayPal system.

iii. **Characteristic of activation or the primary actor**

Users

iv. **Flow of events**

- **Basic flow**

  This use case begins when the customer fills in the PayPal payment information in the "payment details" page.

  The customer clicks on the "Pay" button.

  The system interacts with the PayPal system to verify customer's PayPal account [E1: Invalid Account Information], [E2: Insufficient Fund].

  The system shall display the "Successfully Paid for Service" message.

  The customer will be able to gain his service which he has already paid for.

- **Exceptional flow(E1): Invalid Account Information**

  The system will display error messages and the user has to re-enter the right account information.

- **Exceptional flow(E2): Insufficient Fund**

  The system will display error message and the user has to refill his PayPal account with sufficient money to pay for this service.

v. **Post- conditions**

The users will be able to use or receive the requested service(s).

vi. **Limitation or Constraints**

User must have a PayPal account and must have a valid credit card.

6.4.4. Misuse Case Specification

a. **Misuse case: Brute Force Login**

i. **Brief description**

This misuse case will be used to show attacker the brute force login to exploit the system.

ii. **Pre-conditions**

The attacker should gain access to the website.

iii. **Characteristic of activation or the primary actor**

Execution depends on the attacker's demands.

iv. **Flow of events**

- **Basic flow**

  This misuse case begins when the attacker clicks on the login link in the main page.

  The system displays the login form.

  The attacker enters (Expected User Name, Expected Password) manually or using the dictionary software.

  Attacker presses the "Login" button [A1: Direct login by the dictionary software].

  The system shall verify the information entered (username and password) [E-1: Invalid user name or password].

  The system will determine which functions are available for users and administrator and will display the specific workspace or the main homepage depending on the entire information.

  The attacker will be able to interact with their authorized functions.

- **Alternative flow(A1): Direct login by the dictionary software**

  The attacker uses dictionary software to enter a username and password many times, and allow change happening to the user's data until they get the right username and password.

- **Exceptional flow(E1): Invalid user name or password**

  The system will display error message and the attacker has to re-enter the username and password manually or by using the dictionary software.

v. **Post- conditions**

The attacker can enter the system.

The attacker as a normal user will have his level of privileges.

vi. **Limitation or Constraints**

Not applicable, the attacker could try several user names and passwords until she/he manages to login using other users' account.

b. **Misuse case: Malicious Code Injection**

i. **Brief description**

This misuse case will be used to show the attacker's Intention to post a service and inject a hidden malicious code inside the posted service.

ii. **Pre-conditions**

The attacker must have a username and a password which means they must register or make a brute force attack to get access to the system.

iii. **Characteristic of activation or the primary actor**

Execution depends on the attacker's demands.

iv. **Flow of events**

- **Basic flow**

This misuse case begins when the attacker injects/posts service with hidden malicious codes inside the service.

The system activates the posted service.

The system displays the "Thanks for posting, service will be posted within 24 hours" message.

v. **Post- conditions**

The attacker can enter the system.

The attacker as a normal user will have his level of privileges.

vi. **Limitation or Constraints**

Attackers must login to the E-business main page. Hence, in order to craft malicious codes they must login into the system.

c. **Misuse case: Disclose User Information**

i. **Brief description**

This misuse case will be used to show that the attacker has obtained, either directly or indirectly, sensitive information in a database.

ii. **Pre-conditions**

The attacker should be able to access the website with administrative privileges.

iii. **Characteristic of activation or the primary actor**

Execution depends on attacker's demands.

iv. **Flow of events**

- **Basic flow**

  This misuse case begins when the attacker injects mass exploits with data and SQL Injection in the main page at the login page.

  The system verifies the data entered.

  The system gives him the authorization to access the database.

  The attacker has full privileges where he is able to access all rows in the associated tables (read, write and modify data).

v. **Post- conditions**

The attacker can view everything in the databases.

vi. **Limitation or Constraints**

Attackers must login to the E-business main page. Hence, in order to inject mass exploits with data and SQL injection, they must login into the system as administrator.

6.4.5. Elicit Security Requirements

Security requirements: are the requirements which cover all software/business requirements (Assets). So, all security requirements that depend on the discovery and dream phase output should be discovered. Table 19 below demonstrates both of software requirements and related security requirements.

TABLE 19: Software requirements and related security requirements

Software Requirements	Security Requirements
Login System	Data and Input Validation Cross Site Scripting (XSS) Command Injection Flaws Authentication and Authorization Cryptography
Make Registration	Data and Input Validation Authentication and Authorization Cryptography
View Accounts	Command Injection Flaws
Delete Accounts	Command Injection Flaws
Suggest Changes	Cross Site Scripting (XSS) Command Injection Flaws
Post Services	Command Injection Flaws
Edit Services	Command Injection Flaws
Delete Services	Command Injection Flaws
View Uploaded Services	Buffer Overflows Denial Of Service
Reject Services	Command Injection Flaws
Accept Services	Command Injection Flaws
View Admin Feedback	Denial Of Service
View Posted Services	Denial Of Service
Purchase Services	Buffer Overflows Denial Of Service
Make Payment	Denial Of Service

6.4.6. Quantify Vulnerability Index

Every software has its own vulnerabilities, which are considered as the weak points that allow attackers to either reduce a system's information assurance or to deny the service. In this section, the VI that may happen in the system will be calculated.

Now an algorithm (3) is used to solve a problem in the following case. The security experts wish to find all vulnerabilities' effects (for every single vulnerability related to the system) in the system. A universe U is described as follows:

$$U=\{V_1, V_2, V_3, V_4, V_5, \ldots\ldots V_{25}\}.$$

Where V denote "vulnerabilities".

Let $E_u = \{E_u\}$ be a set of decision parameters related to the above universe U "vulnerabilities" according to the system type and security experts. According to the last two phases (discovery and dream) and after investigating all vulnerabilities related to this type of systems depending on the SANS list and supported by NIST, *Vulnerabilities* are:

Note: abbreviations used:

1. SQL Injection (*SQLI*).
2. OS Command Injection (*OSCI*).
3. Classic Buffer Overflow (*CLBO*).
4. Cross-site Scripting (*CR-sS*).
5. Download of Code Without Integrity Check (*DOCWIC*).
6. Missing Authentication for Critical Function (*MACF*).
7. Missing Authorization (*MA*).
8. Use of Hard-coded Credentials (*UH-cC*).
9. Missing Encryption of Sensitive Data (*MESD*).
10. Unrestricted Upload of File of Dangerous Type (*UUFDT*).
11. Reliance on Untrusted Inputs in a Security Decision (*RUISD*).
12. Execution with Unnecessary Privileges (*EUP*).
13. Cross-Site Request Forgery (*CSRF*).
14. Improper Limitation of a Pathname to a Restricted Directory (*ILPRD*).
15. Incorrect Authorization (*IA*).
16. Inclusion of Functionality from Untrusted Control Sphere (*IFUCS*).
17. Incorrect Permission Assignment for Critical Resource (*IPACR*).
18. Use of Potentially Dangerous Function (*UPDF*).
19. Use of a Broken or Risky Cryptographic Algorithm (*UBRCA*).
20. Incorrect Calculation of Buffer Size (*ICBS*).
21. Improper Restriction of Excessive Authentication Attempts (*IREAA*).
22. URL Redirection to Untrusted Site (*URL-RUS*).
23. Uncontrolled Format String (*UFS*).
24. Integer Overflow or Wraparound (*IOW*).
25. Use of a One-Way Hash without a Salt (*UOWHS*).

U={*SQLI, OSCI, CLBO, CR-sS, DOCWIC, MACF, MA, UH-cC, MESD, UUFDT, RUISD, EUP, CSRF, ILPRD, IA, IFUCS, IPACR, UPDF, UBRCA, ICBS, IREAA, URL-RUS, UFS, IOW, UOWHS*}.

Eu ={$e_{u,1}$= *occurrence one time,* $e_{u,2}$= *occurrence two times,* $e_{u,3}$= *high damage,* $e_{u,4}$= *low damage,* $e_{u,5}$= *effect on very important asset,* $e_{u,6}$= *effect on important asset,* $e_{u,7}$= *effect on less important asset*}.

In this real case, seven parameters (E_u) have been used and values given according to the security experts' perspectives. For instance if a vulnerability X occurs two times in the system, it should have a high value- see $e_{u,2}$ and vice versa. In other cases, more or less parameters and different values can be adopted.

$$E = \left\{ \frac{e_{u,1}}{0.5}, \frac{e_{u,2}}{1}, \frac{e_{u,3}}{1}, \frac{e_{u,4}}{0.5}, \frac{e_{u,5}}{1}, \frac{e_{u,6}}{0.7}, \frac{e_{u,7}}{0.4} \right\}$$

Now, the function for each parameter i.e. $F(e_{u,1})$ will be found, which means the parameter $(e_{u,1})$ which is "*occurrence one time*" and it has (0.5) value in this system. According to security experts' opinions, all vulnerabilities that occur one time in this system will be part of this function for instance (*SQLI, CLBO)* vulnerabilities appearing in this case study.

U ={*SQLI, OSCI, CLBO, CR-sS, DOCWIC, MACF, MA, UH-cC, MESD, UUFDT, RUISD, EUP, CSRF, ILPRD, IA, IFUCS, IPACR, UPDF, UBRCA, ICBS, IREAA, URL-RUS, UFS, IOW, UOWHS*}.

$F(e_{u,1})$ = F(Occurrence One Time) = {*ILPRD, IA, IFUCS, IPACR, UPDF, UBRCA, ICBS, IREAA, URL-RUS, UFS, IOW, UOWHS*}.

$F(e_{u,2})$ = F(Occurrence Two Times) = {*SQLI, OSCI, CLBO, CR-sS, DOCWIC, MACF, MA, UH-cC, MESD, UUFDT, RUISD, EUP, CSRF*}.

$F(e_{u,3})$ = F(High Damage) = {*SQLI, OSCI, CLBO, CR-sS, DOCWIC, MACF, MA, UH-cC, MESD, UUFDT, RUISD, EUP, CSRF, ILPRD, IA, IFUCS, IPACR*}.

$F(e_{u,4})$ = F(Low Damage) = {*UPDF, UBRCA, ICBS, IREAA, URL-RUS, UFS, IOW, UOWHS*}.

$F(e_{u,5})$ = F(Effect On Very Important Asset) = {*SQLI, OSCI, CLBO, CR-sS, DOCWIC, MACF, MA, UH-cC, MESD, UUFDT, RUISD*}.

$F(e_{u,6})$ = F(Effect On Important Asset) = {*EUP, CSRF, ILPRD, IA, IFUCS, IPACR, UPDF, SQLI*}.

$F(e_{u,7})$ = F(Effect On Less Important Asset) = {*UBRCA, ICBS, IREAA, URL-RUS, UFS, IOW, UOWHS, CR-sS*}.

Then, the fuzzy soft set (F, E) can be viewed as consisting of the following collection of approximations:

(F, E) = {($e_{u,1}$, ({*ILPRD, IA, IFUCS, IPACR, UPDF, UBRCA, ICBS, IREAA, URL-RUS, UFS, IOW, UOWHS*})), ($e_{u,2}$, ({*SQLI, OSCI, CLBO, CR-sS, DOCWIC, MACF, MA, UH-cC, MESD, UUFDT, RUISD, EUP, CSRF*})), ($e_{u,3}$, ({*SQLI, OSCI, CLBO, CR-sS, DOCWIC, MACF, MA, UH-cC, MESD, UUFDT, RUISD, EUP, CSRF, ILPRD, IA, IFUCS, IPACR*})), ($e_{u,4}$, ({*UPDF, UBRCA, ICBS, IREAA, URL-RUS, UFS, IOW, UOWHS*})), ($e_{u,5}$, ({*SQLI, OSCI, CLBO, CR-sS, DOCWIC, MACF, MA, UH-cC, MESD, UUFDT, RUISD*})), ($e_{u,6}$, ({*EUP, CSRF, ILPRD, IA, IFUCS, IPACR, UPDF, SQLI*})), ($e_{u,7}$, ({*UBRCA, ICBS, IREAA, URL-RUS, UFS, IOW, UOWHS, CR-sS*}))}.

The Cagman, Citak and Enginoglu Algorithm (CCEA) applied to the first fuzzy soft part in (F, E) to take the decision from the availability set U incorporating the choice values:

$$\mu_F{}^d{}_X = \frac{1}{25}\left[\mu_x\left(e_{U_{x1}}\right)_{xfx}\left(e_{U_{x1}}\right)\left(V_1\right) + \mu_x\left(e_{U_{x2}}\right)_{xfx}\left(e_{U_{x2}}\right)\left(V_1\right) + \mu_x\left(e_{U_{x3}}\right)_{xfx}\left(e_{U_{x3}}\right)\left(V_1\right) + \mu_x\left(e_{U_{x4}}\right)_{xfx}\left(e_{U_{x4}}\right)\left(V_1\right) + \mu_x\left(e_{U_{x5}}\right)_{xfx}\left(e_{U_{x5}}\right)\left(V_1\right) + \mu_x\left(e_{U_{x6}}\right)_{xfx}\left(e_{U_{x6}}\right)\left(V_1\right) + \mu_x\left(e_{U_{x7}}\right)_{xfx}\left(e_{U_{x7}}\right)\left(V_1\right)\right]$$

$$\mu_F{}^d{}_X = \frac{1}{25}[(0.5 \times 0) + (1 \times 1) + (1 \times 1) + (0.5 \times 0) + (1 \times 1) + (0.7 \times 1) + (0.4 \times 0)]$$

$$= \frac{1}{25}[1 + 1 + 1 + 0.7] = 0.148$$

$$\mu_F{}^d{}_X = \frac{1}{25}\left[\mu_x\left(e_{U_{x1}}\right)_{xfx}\left(e_{U_{x1}}\right)\left(V_2\right) + \mu_x\left(e_{U_{x2}}\right)_{xfx}\left(e_{U_{x2}}\right)\left(V_2\right) + \mu_x\left(e_{U_{x3}}\right)_{xfx}\left(e_{U_{x3}}\right)\left(V_2\right) + \mu_x\left(e_{U_{x4}}\right)_{xfx}\left(e_{U_{x4}}\right)\left(V_2\right) + \mu_x\left(e_{U_{x5}}\right)_{xfx}\left(e_{U_{x5}}\right)\left(V_2\right) + \mu_x\left(e_{U_{x6}}\right)_{xfx}\left(e_{U_{x6}}\right)\left(V_2\right) + \mu_x\left(e_{U_{x7}}\right)_{xfx}\left(e_{U_{x7}}\right)\left(V_2\right)\right]$$

$$\mu_F{}^d{}_X = \frac{1}{25}[(0.5 \times 0) + (1 \times 1) + (1 \times 1) + (0.5 \times 1) + (1 \times 1) + (0.7 \times 1) + (0.4 \times 0)]$$

$$= \frac{1}{25}[1 + 1 + 1] = 0.12$$

And so on for all vulnerabilities.

Then ${}_F{}^d{}_X$ is represented by.

$${}_F{}^d{}_X = \left\{\frac{0.148}{V_1}, \frac{0.12}{V_2}, \frac{0.12}{V_3}, \frac{0.136}{V_4}, \frac{0.12}{V_5}, \frac{0.12}{V_6}, \frac{0.12}{V_7}, \frac{0.12}{V_8}, \frac{0.12}{V_9}, \frac{0.12}{V_{10}}, \frac{0.12}{V_{11}}, \frac{0.108}{V_{12}}, \frac{0.108}{V_{13}}, \frac{0.088}{V_{14}}, \frac{0.088}{V_{15}}\right\}$$

$$\left\{\frac{0.088}{V_{16}}, \frac{0.088}{V_{17}}, \frac{0.068}{V_{18}}, \frac{0.056}{V_{19}}, \frac{0.056}{V_{20}}, \frac{0.056}{V_{21}}, \frac{0.056}{V_{22}}, \frac{0.056}{V_{23}}, \frac{0.056}{V_{24}}, \frac{0.056}{V_{25}}\right\}$$

i.e. the *Fuzzy* decision for this vulnerabilities (VI) are:

1. (V_1) SQL Injection (*SQLI*) = 0.148
2. (V_2) OS Command Injection (*OSCI*) = 0.12
3. (V_3) Classic Buffer Overflow (*CLBO*) = 0.12
4. (V_4) Cross-site Scripting (*CR-sS*) = 0.136
5. (V_5) Download of Code Without Integrity Check (*DOCWIC*) = 0.12

6. (V_6) Missing Authentication for Critical Function (*MACF*) = 0.12
7. (V_7) Missing Authorization (*MA*) = 0.12
8. (V_8) Use of Hard-coded Credentials (*UH-cC*) = 0.12
9. (V_9) Missing Encryption of Sensitive Data (*MESD*) = 0.12
10. (V_{10}) Unrestricted Upload of File of Dangerous Type (*UUFDT*) = 0.12
11. (V_{11}) Reliance on Untrusted Inputs in a Security Decision (*RUISD*) = 0.12
12. (V_{12}) Execution with Unnecessary Privileges (*EUP*) = 0.108
13. (V_{13}) Cross-Site Request Forgery (*CSRF*) = 0.108
14. (V_{14}) Improper Limitation of a Pathname to a Restricted Directory (*ILPRD*) = 0.088
15. (V_{15}) Incorrect Authorization (*IA*) = 0.088
16. (V_{16}) Inclusion of Functionality from Untrusted Control Sphere (*IFUCS*) = 0.088
17. (V_{17}) Incorrect Permission Assignment for Critical Resource (*IPACR*) = 0.088
18. (V_{18}) Use of Potentially Dangerous Function (*UPDF*) = 0.068
19. (V_{19}) Use of a Broken or Risky Cryptographic Algorithm (*UBRCA*) = 0.056
20. (V_{20}) Incorrect Calculation of Buffer Size (*ICBS*) = 0.056
21. (V_{21}) Improper Restriction of Excessive Authentication Attempts (*IREAA*) = 0.056
22. (V_{22}) URL Redirection to Untrusted Site (*URL-RUS*) = 0.056
23. (V_{23}) Uncontrolled Format String (*UFS*) = 0.056
24. (V_{24}) Integer Overflow or Wraparound (*IOW*) = 0.056
25. (V_{25}) Use of a One-Way Hash without a Salt (*UOWHS*) = 0.056

The 25 vulnerabilities above are part of the most dangerous software and web applications' vulnerabilities in the NIST and SANS institute lists (NIST, 8 September 2013; KLOCWORK, 12 November 2013). It has been noted that the maximum value of $\mu_F^d x = 0.148$ among all vulnerabilities; meaning that the *SQL Injection* vulnerability receives the most influence in the proposed website under this parameters (E_u), as well as in SANS institute list *SQL Injection* is the most dangerous software vulnerability. On the other hand, the errors that cause these vulnerabilities have a direct impact on the web applications. So, the next sub section will discuss this issue in further detail.

6.4.7. Quantify Error Index

Every single error or more can lead to one vulnerability or more (M. Khan & Zulkernine, 2008), which are manipulated by attackers to control the system, and to read, modify or destroy it. In this sub section, the EI that may be available in the system will be quantified.

Now, the equation (3) as in page 115 is recalled, as it is used in the previous sub section to calculate the EI. The security experts want to find all error effects (for every single error related to the system) in the system. A universe U will be described as follows according to the NIST and SANS lists (NIST, 8 September 2013; KLOCWORK, 12 November 2013).

$U=\{R_1, R_2, R_3, R_4, R_5, R_6 R_7, R_8, R_9, R_{10}\}$.

Where R denote "errors".

Let $Eu = \{Eu\}$ be a set of decision parameters related to the above universe U "errors" according to the system type and security experts.

After investigating all errors related to this type of systems according to SANS list and supported by NIST, *errors* are:

Note: abbreviations used:

1. Poor SQL commands are used to check user names and passwords (*PSQL*).
2. The program executed will allow arguments to be specified within an input file or from the standard input (*PEAASI*).
3. Code path includes a Buffer Write Operation (*CPBWO*).
4. Buffer is as large as you specify (*BLS*).
5. Replace unbounded copy functions with analogous functions that support length arguments (*RUCFSLA*).
6. Set the session cookie to be not only Http (*SSCH*).

7. Assume that all inputs are not malicious (*AIM*).
8. Input Validation does not consider all potentially relevant properties (*IVPRP*).
9. Do not use proper output encoding, escaping, and quoting (*DPOEEQ*).
10. Encrypt the code with a reliable encryption scheme before transmitting (*ECREST*).

In this case a set of errors containing ten errors is used, depending on NIST and SANS lists (NIST, 8 September 2013; KLOCWORK, 12 November 2013), and in other cases more or less errors can be used, according to the system type. All errors used must have a relationship with the vulnerabilities which are calculated in the previous section, otherwise, security requirements cannot be quantified.

U={*PSQL, PEAASI, CPBWO, BLS, RUCFSLA, SSCH, AIM, IVPRP, DPOEEQ, ECREST*}.

Eu={$eu_{,1}$= *occurrence one time,* $eu_{,2}$= *occurrence two times,* $eu_{,3}$= *occurrence three times or more,* $eu_{,4}$= *effect on one vulnerability,* $eu_{,5}$= *effect on two vulnerabilities,* $eu_{,6}$= *effect on three vulnerabilities or more,* $eu_{,7}$= *effect on very dangerous vulnerability,* $eu_{,8}$= *effect on dangerous vulnerability,* $eu_{,9}$= *effect on normal vulnerability*}.

In this case nine parameters (E_u) have been used and values were given according to security experts' perspectives. For instance if an error X occurs three times in the system, it should have a high value, see $e_{u,3}$ and vice versa. In other cases more or less than this number of parameters and different values.

$$E = \left\{ \frac{e_{u,1}}{0.33}, \frac{e_{u,2}}{0.66}, \frac{e_{u,3}}{1}, \frac{e_{u,4}}{0.33}, \frac{e_{u,5}}{0.66}, \frac{e_{u,6}}{1}, \frac{e_{u,7}}{1}, \frac{e_{u,8}}{0.66}, \frac{e_{u,9}}{0.33} \right\}$$

Now, the function for each parameter i.e. F ($e_{u,1}$) means the parameter ($e_{u,1}$) which is "*occurrence one time*" has (0.33) value in this system. According to the security

experts' opinion, all errors that occur one time in this system will be part of this function like (*ECREST, PEAASI)* errors.

$F(e_{u,1})$ = *F*(Occurrence One Time) = { *ECREST, PEAASI* }.

$F(e_{u,2})$ = *F*(Occurrence Two Times) = { *CPBWO, IVPRP, DPOEEQ, AIM* }.

$F(e_{u,3})$ = *F*(Occurrence Three Times Or More) = { *PSQL, BLS, RUCFSLA, SSCH* }.

$F(e_{u,4})$ = *F*(Effect On One Vulnerability) = { *PSQL, CPBWO, BLS, RUCFSLA, SSCH, AIM, DPOEEQ, ECREST* }.

$F(e_{u,5})$ = *F*(Effect On Two Vulnerabilities) = { *PEAASI* }.

$F(e_{u,6})$ = *F*(Effect On Three Vulnerabilities Or More) = { *IVPRP* }.

$F(e_{u,7})$ = *F*(Effect On Very Dangerous Vulnerability) = { *PSQL, PEAASI, IVPRP* }.

$F(e_{u,8})$ = *F*(Effect On Dangerous Vulnerability) = { *PEAASI, IVPRP, SSCH, DPOEEQ, AIM* }.

$F(e_{u,9})$ = *F*(Effect On Normal Vulnerability) = { *CPBWO, BLS, RUCFSLA, ECREST*}.

Then, the fuzzy soft set (*F, E*) can be seen as consisting the following collection of approximations:

(*F, E*) = {($e_{u,1}$, ({*ECREST, PEAASI*})), ($e_{u,2}$, ({*CPBWO, IVPRP, DPOEEQ, AIM*})), ($e_{u,3}$, ({*PSQL, BLS, RUCFSLA, SSCH*})), ($e_{u,4}$, ({*PSQL, CPBWO, BLS, RUCFSLA, SSCH, AIM, DPOEEQ, ECREST*})), ($e_{u,5}$, ({*PEAASI*})),($e_{u,6}$, ({*IVPRP*})),($e_{u,7}$, ({*PSQL, PEAASI, IVPRP*})), ($e_{u,8}$, ({*PEAASI, IVPRP, SSCH, DPOEEQ, AIM* })), ($e_{u,9}$, ({*CPBWO, BLS, RUCFSLA, ECREST*}))}.

Cagman, Citak and Enginoglu Algorithm (CCEA) applied to the first fuzzy soft part in (*F, E*) to take the decision from the availability set *U*. The choice values are then incorporated.

$$\mu_F{}^d{}_X = \frac{1}{10}\left[\mu_X(e_{U_{,1}})_{\mathcal{Y}fx}(e_{U_{,1}})(R_1) + \mu_X(e_{U_{,2}})_{\mathcal{Y}fx}(e_{U_{,2}})(R_1) + \mu_X(e_{U_{,3}})_{\mathcal{Y}fx}(e_{U_{,3}})(R_1) + \mu_X(e_{U_{,4}})_{\mathcal{Y}fx}(e_{U_{,4}})(R_1) + \mu_X(e_{U_{,5}})_{\mathcal{Y}fx}(e_{U_{,5}})(R_1) + \mu_X(e_{U_{,6}})_{\mathcal{Y}fx}(e_{U_{,6}})(R_1) + \mu_X(e_{U_{,7}})_{\mathcal{Y}fx}(e_{U_{,7}})(R_1) + \mu_X(e_{U_{,8}})_{\mathcal{Y}fx}(e_{U_{,8}})(R_1) + \mu_X(e_{U_{,9}})_{\mathcal{Y}fx}(e_{U_{,9}})(R_1)\right]$$

$$\mu_F{}^d{}_X = \frac{1}{10}[(0.33 \times 0) + (0.66 \times 0) + (1 \times 1) + (0.33 \times 1) + (0.66 \times 0) + (1 \times 0) + (1 \times 1) + (0.66 \times 0) + (0.33 \times 0)]$$

$$= \frac{1}{10}[1 + 0.33 + 1] = 0.233$$

$$\mu_F{}^d{}_X = \frac{1}{10}\left[\mu_X(e_{U_{,1}})_{\mathcal{Y}fx}(e_{U_{,1}})(R_2) + \mu_X(e_{U_{,2}})_{\mathcal{Y}fx}(e_{U_{,2}})(R_2) + \mu_X(e_{U_{,3}})_{\mathcal{Y}fx}(e_{U_{,3}})(R_2) + \mu_X(e_{U_{,4}})_{\mathcal{Y}fx}(e_{U_{,4}})(R_2) + \mu_X(e_{U_{,5}})_{\mathcal{Y}fx}(e_{U_{,5}})(R_2) + \mu_X(e_{U_{,6}})_{\mathcal{Y}fx}(e_{U_{,6}})(R_2) + \mu_X(e_{U_{,7}})_{\mathcal{Y}fx}(e_{U_{,7}})(R_2) + \mu_X(e_{U_{,8}})_{\mathcal{Y}fx}(e_{U_{,8}})(R_2) + \mu_X(e_{U_{,9}})_{\mathcal{Y}fx}(e_{U_{,9}})(R_2)\right]$$

$$\mu_F{}^d{}_X = \frac{1}{10}[(0.33 \times 1) + (0.66 \times 0) + (1 \times 0) + (0.33 \times 0) + (0.66 \times 1) + (1 \times 0) + (1 \times 1) + (0.66 \times 1) + (0.33 \times 0)]$$

$$= \frac{1}{10}[0.33 + 0.66 + 1 + 0.66] = 0.265$$

And so on for all errors.

Then ${}_F{}^d{}_X$ is represented by:

$${}_F{}^d{}_X = \left\{\frac{0.233}{R_1}, \frac{0.265}{R_2}, \frac{0.132}{R_3}, \frac{0.166}{R_4}, \frac{0.166}{R_5}, \frac{0.199}{R_6}, \frac{0.165}{R_7}, \frac{0.332}{R_8}, \frac{0.165}{R_9}, \frac{0.099}{R_{10}}\right\}$$

i.e. the *Fuzzy* decision for this errors (EI) is established below:

1. (E_1) Poor SQL commands are used to check user names and passwords (*PSQL*) = 0.233
2. (E_2) The program executed allows arguments to be specified within an input file or from the standard input (*PEAASI*) = 0.265
3. (E_3) Code path includes a Buffer Write Operation (*CPBWO*) = 0.132
4. (E_4) Buffer is as large as can be specified (*BLS*) = 0.166

5. (E_5) Replace unbounded copy functions with analogous functions that support lengthy arguments (*RUCFSLA*) = 0.166
6. (E_6) Set the session cookie to be not only Http (*SSCH*) = 0.199
7. (E_7) Assume all inputs are not malicious (*AIM*) = 0.165
8. (E_8) Input Validation does not consider all potentially relevant properties (*IVPRP*) = 0.332
9. (E_9) Does not use proper output encoding, escaping, and quoting (*DPOEEQ*) = 0.165
10. (E_{10}) Encrypt the code with a reliable encryption scheme before transmitting (*ECREST*) = 0.099

It is noted that the maximum value of $\mu_F{}^d{}_X$ = 0.332 among all errors; therefore "*Input Validation did not consider all potentially relevant properties*" error receives the most influence in the website under this parameters (E_u), and that all of the above ten errors are part of the NIST and SANS error lists (NIST, 8 September 2013; KLOCWORK, 12 November 2013).

The EI causes the system's vulnerabilities. Vulnerabilities and errors have a direct impact on this web application, but all of them need to find the SI and security requirements index. So, the next sub sections will discuss in depth both the SI and security requirements index.

6.4.8. Quantify Security Index

After calculating both of the VI and EI which are related to the web application in this real case, the next step is to calculate the SI, then finally, to connect SI contents with the security requirements to obtain the result, which is the security requirements index. There is a relationship between VI and EI; this relationship gives us the SI. Using equation (5) the SI can be calculated.

$$SI^{Vi} = f(Vi, Ej)$$

$$SI^{Vi} = \sum_{i,j \in I} \frac{V_i \; E_j}{j} \quad \ldots\ldots(5)$$

Where;

SI^{Vi} : is security index for vulnerability i .

Now, for calculating the SI, need VI and EI shall be calculated in these last two sub sections.

Vulnerabilities Index:

1. SQL Injection (*SQLI*) = 0.148
2. OS Command Injection (*OSCI*) = 0.12
3. Classic Buffer Overflow (*CLBO*) = 0.12
4. Cross-site Scripting (*CR-sS*) = 0.136
5. Download of Code Without Integrity Check (*DOCWIC*) = 0.12
6. Missing Authentication for Critical Function (*MACF*) = 0.12
7. Missing Authorization (*MA*) = 0.12
8. Use of Hard-coded Credentials (*UH-cC*) = 0.12
9. Missing Encryption of Sensitive Data (*MESD*) = 0.12
10. Unrestricted Upload of File of Dangerous Type (*UUFDT*) = 0.12
11. Reliance on Untrusted Inputs in a Security Decision (*RUISD*) = 0.12
12. Execution with Unnecessary Privileges (*EUP*) = 0.108
13. Cross-Site Request Forgery (*CSRF*) = 0.108
14. Improper Limitation of a Pathname to a Restricted Directory (*ILPRD*) = 0.088
15. Incorrect Authorization (*IA*) = 0.088
16. Inclusion of Functionality from Untrusted Control Sphere (*IFUCS*) = 0.088
17. Incorrect Permission Assignment for Critical Resource (*IPACR*) = 0.088
18. Use of Potentially Dangerous Function (*UPDF*) = 0.068
19. Use of a Broken or Risky Cryptographic Algorithm (*UBRCA*) = 0.056
20. Incorrect Calculation of Buffer Size (*ICBS*) = 0.056

21. Improper Restriction of Excessive Authentication Attempts (*IREAA*) = 0.056
22. URL Redirection to Untrusted Site (*URL-RUS*) = 0.056
23. Uncontrolled Format String (*UFS*) = 0.056
24. Integer Overflow or Wraparound (*IOW*) = 0.056
25. Use of a One-Way Hash without a Salt (*UOWHS*) = 0.056

Errors Index:

1. Poor SQL commands are used to check user names and passwords (*PSQL*) = 0.233
2. The program executed allows arguments to be specified within an input file or from the standard input (*PEAASI*) = 0.265
3. Code path includes a Buffer Write Operation (*CPBWO*) = 0.132
4. Buffer is as large as can be specified (*BLS*) = 0.166
5. Replace unbounded copy functions with analogous functions that support lengthy arguments (*RUCFSLA*) = 0.166
6. Set the session cookie to be not only Http (*SSCH*) = 0.199
7. Assume all inputs are not malicious (*AIM*) = 0.165
8. Input Validation does not consider all potentially relevant properties (*IVPRP*) = 0.332
9. Does not use proper output encoding, escaping, and quoting (*DPOEEQ*) = 0.165
10. Encrypt the code with a reliable encryption scheme before transmitting (*ECREST*) = 0.099

Every single vulnerability consists of one error or more, and to calculate the SI for any security requirements the relationships between vulnerabilities and errors and next, the SI must be found using equation (5). Now, the SI will be located by using both the VI and EI as follows:

1. *SQLI ={ PSQL, PEAASI, IVPRP}.*
2. *OSCI ={ PEAASI, IVPRP}.*
3. *CLBO ={ CPBWO, BLS, RUCFSLA}.*
4. *CR-sS ={ SSCH, AIM, IVPRP, DPOEEQ}.*
5. *DOCWIC ={ ECREST}.*

6. *MACF ={ ECREST, PSQL, AIM, IVPRP}.*
7. *MA ={ PSQL, PEAASI, AIM, ECREST}.*
8. *UH-cC ={ ECREST}.*
9. *MESD ={ ECREST, IVPRP}.*
10. *UUFDT ={ PEAASI, BLS, DPOEEQ}.*
11. *RUISD ={ AIM, IVPRP, PEAASI}.*
12. *EUP ={ PSQL, CPBWO, RUCFSLA}.*
13. *CSRF ={ PSQL, PEAASI, IVPRP}.*
14. *ILPRD ={ PSQL}.*
15. *IA ={ ECREST, PSQL, AIM, IVPRP}.*
16. *IFUCS ={ PSQL, PEAASI, AIM, ECREST}.*
17. *IPACR ={ ECREST, PSQL, AIM, IVPRP}.*
18. *UPDF ={ PSQL}.*
19. *UBRCA ={ ECREST}.*
20. *ICBS ={ CPBWO, BLS}.*
21. *IREAA ={ AIM}.*
22. *URL-RUS ={ SSCH}.*
23. *UFS ={ PEAASI, RUCFSLA}.*
24. *IOW ={ RUCFSLA}.*
25. *UOWHS ={ DPOEEQ, ECREST}.*

This indicates that the "SQL Injection (*SQLI*)" vulnerability consists of "Poor SQL commands which are used to check user names and passwords (*PSQL*)" error, "The program be executed allows arguments to be specified within an input file or from standard input (*PEAASI*)" error and "Input Validation did not consider all potentially relevant properties (*IVPRP*)" error. This applies to all vulnerabilities as mentioned above. Now the SI for each vulnerability alone using equation (5) is calculated as follows:

$$SI^{Vi} = \sum_{i,j \in I} \frac{V_i \; E_j}{j} \quad \ldots\ldots(5)$$

$$SI^{SQLI} = \frac{(SQLI \times PSQL) + (SQLI \times PEAASI) + (SQLI \times IVPRP)}{3}$$

$$= \frac{(0.148 \times 0.233) + (0.148 \times 0.265) + (0.148 \times 0.332)}{3}$$

$$= \frac{0.034484 + 0.03922 + 0.049136}{3}$$

$$= \frac{0.12284}{3}$$

$$SI^{SQLI} = 0.041$$

$$SI^{OSCI} = \frac{(OSCI \times PEAASI) + (OSCI \times IVPRP)}{2}$$

$$= \frac{(0.12 \times 0.265) + (0.12 \times 0.332)}{2}$$

$$SI^{OSCI} = 0.036$$

This works in the same way for all vulnerabilities. Concerning the SI for all the vulnerabilities, fuzzy numbers mean that when the number approaches to one, the seriousness of this vulnerability increases; so it should be removed to a high priority, and vice versa. The vulnerabilities are arranged according to their dangerous degrees as follows:

Security Index:

1. SQL Injection (*SQLI*) = 0.041
2. OS Command Injection (*OSCI*) = 0.036
3. Cross-Site Request Forgery (*CSRF*) = 0.030

4. Reliance on Untrusted Inputs in a Security Decision (*RUISD*) = 0.030
5. Cross-site Scripting (*CR-sS*) = 0.029
6. Missing Encryption of Sensitive Data (*MESD*) = 0.026
7. Missing Authentication for Critical Function (*MACF*) = 0.025
8. Unrestricted Upload of File of Dangerous Type (*UUFDT*) = 0.024
9. Missing Authorization (*MA*) = 0.023
10. Improper Limitation of a Pathname to a Restricted Directory (*ILPRD*) = 0.021
11. Classic Buffer Overflow (*CLBO*) = 0.019
12. Execution with Unnecessary Privileges (*EUP*) = 0.019
13. Incorrect Authorization (*IA*) = 0.018
14. Incorrect Permission Assignment for Critical Resource (*IPACR*) = 0.018
15. Inclusion of Functionality from Untrusted Control Sphere (*IFUCS*) = 0.017
16. Use of Potentially Dangerous Function (*UPDF*) = 0.016
17. Download of Code Without Integrity Check (*DOCWIC*) = 0.012
18. Use of Hard-coded Credentials (*UH-cC*) = 0.012
19. Uncontrolled Format String (*UFS*) = 0.012
20. URL Redirection to Untrusted Site (*URL-RUS*) = 0.011
21. Improper Restriction of Excessive Authentication Attempts (*IREAA*) = 0.009
22. Integer Overflow or Wraparound (*IOW*) = 0.009
23. Incorrect Calculation of Buffer Size (*ICBS*) = 0.008
24. Use of a One-Way Hash without a Salt (*UOWHS*) = 0.007
25. Use of a Broken or Risky Cryptographic Algorithm (*UBRCA*) = 0.006

6.5 Destiny Phase

- The developers and the users focus on the expected software, which will be developed and employed.
- Define the activities that have to be considered.
- Support the organization in gathering the information related to the ideas about the system, which becomes apparent to be designed.

- Requirement inspection (the requirements are analyzed for the vagueness of the ultimate security requirements index).
- Build the system, which is in this real case is termed "Online E-Business Website".

Output of Phase 3:

The outputs of this phase are: 'what will be' the initial selected requirements, the documentation of the decision-making process and rationale. Firstly, a step of joining the security requirements with the SI is required to obtain the quantification of security requirements, which is to be called later the security requirements index. Secondly, security requirements are prioritized according to a dangerous degree (Risk Value) as follows:

6.5.1. Categorize Security Requirements Index

At the end of the day, all SI contents will be connected with all security requirements, which cover all software/business requirements (Assets), see table 20. So, each SI should be connected with related security requirements. The security requirements will be arranged according to their dangerous degree as follows:

1. Command Injection Flaws: (*SQLI + OSCI + RUISD + UPDF*) = 0.123
2. Cross Site Scripting (XSS): (*CSRF + CR-sS + EUP + IPACR + URL-RUS*) = 0.107
3. Authentication and Authorization: (*MA + IA + IPACR + IFUCS + IREAA + MACA*) = 0.11
4. Denial Of Service: (*UFS + DH-cC + ILPRD + UUFDT*) = 0.069
5. Buffer Overflows: (*ICBS + IOW + DOCWIC + UPDF + CLBO*) = 0.064
6. Cryptography: (*UBRCA + UOWHS + MESD*) = 0.039
7. Data and Input Validation: (*RUISD*) = 0.030

6.5.2. Prioritize Security Requirements Index

It is presumed that, only a few requirements can be applied; consequently, the most essential requirements have to be determined, but in this case all quantified security requirements will be implemented.

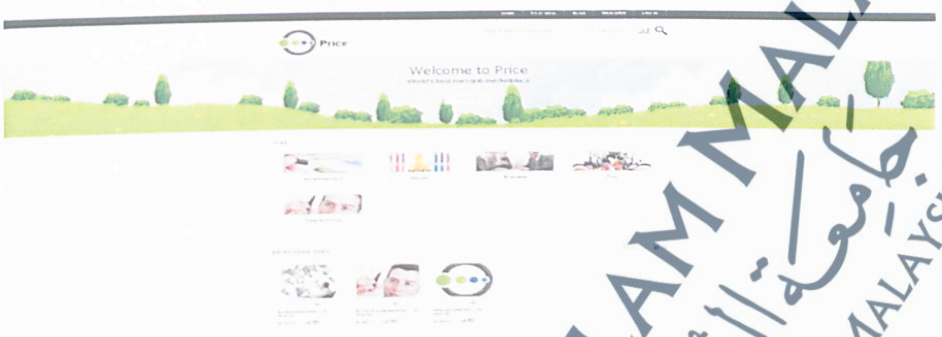
TABLE 20: Software requirements and related security requirements value

Software Requirements	Security Requirements Risk Value	Total Risk Value
Login System	Data and Input Validation: 0.030 Cross Site Scripting (XSS): 0.107 Command Injection Flaws: 0.123 Authentication and Authorization: 0.11 Cryptography: 0.039	0.409
Make Registration	Data and Input Validation: 0.030 Authentication and Authorization: 0.11 Cryptography: 0.039	0.179
View Accounts	Command Injection Flaws: 0.123	0.123
Delete Accounts	Command Injection Flaws: 0.123	0.123
Suggest Changes	Cross Site Scripting (XSS): 0.107 Command Injection Flaws: 0.123	0.23
Post Services	Command Injection Flaws: 0.123	0.123
Edit Services	Command Injection Flaws: 0.123	0.123
Delete Services	Command Injection Flaws: 0.123	0.123
View Uploaded Services	Buffer Overflows: 0.064 Denial Of Service: 0.069	0.133
Reject Services	Command Injection Flaws: 0.123	0.123
Accept Services	Command Injection Flaws: 0.123	0.123
View Admin Feedback	Denial Of Service: 0.069	0.069
View Posted Services	Denial Of Service: 0.069	0.069
Purchase Services	Buffer Overflows: 0.064 Denial Of Service: 0.069	0.133
Make Payment	Denial Of Service: 0.069	0.069

6.6 Website Screens and Main Functions

At the main page, each of the administrator, customer and service seller can make a registration and make a login if they have registered before.

FIGURE 22: Main Page.

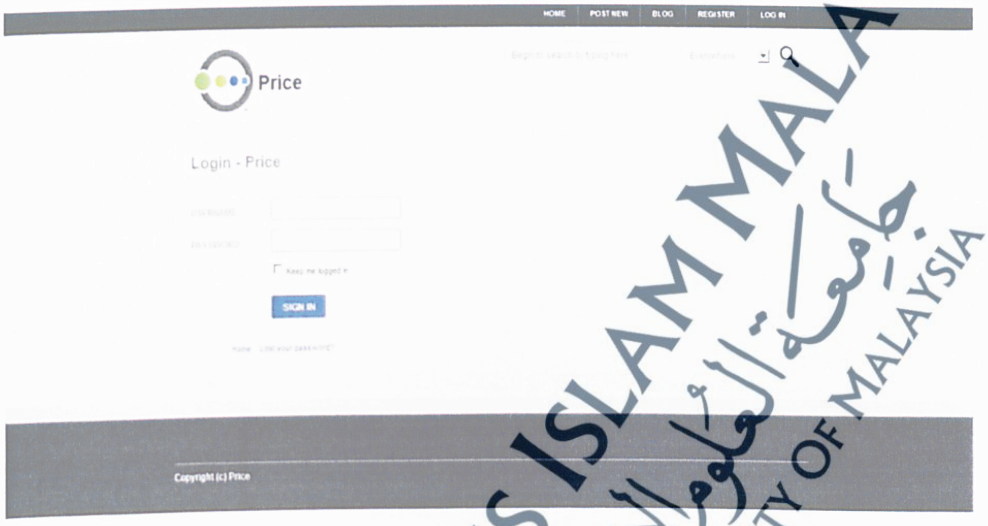


At the Registration Page, both customer and service seller should make a registration to gain access to the system. Customer and service seller should insert a User name and E-mail then the Password will be sent to t their respective emails.

FIGURE 23: Registration Page.

At the Login Page, each administrator, customer and service seller can access the system by entering the valid Username and Password to access the system according to his/her privileges.

FIGURE 24: Login Page.

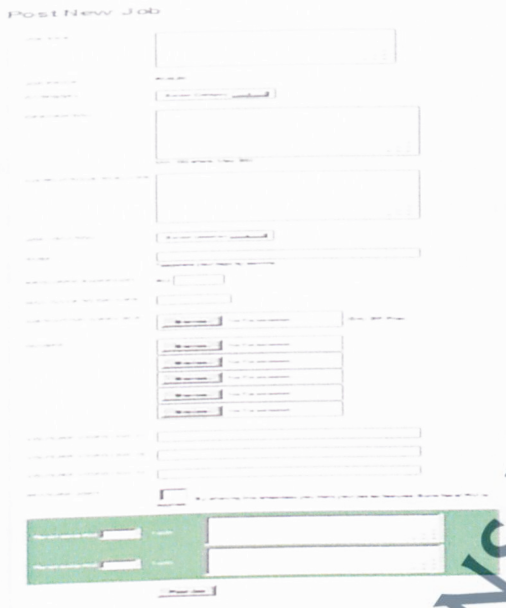


At the Customer Page, the customer can do his or her shopping. i.e. can order any services he or she requests.

FIGURE 25: Customer Page.

At the Post Services Page, the service seller can post his products to this page, by entering all the product details.

FIGURE 26: Post Services Page.



At the Payment Page. Customer and service seller can pay for any service he wants using his account in the website or like normal user (by PayPal).

FIGURE 27: Payment Page.

6.7 Summary

In this chapter, the Secure Appreciative Inquiry Fuzzy Quantification Technique (SAIFQT) was implemented through a real case study. SAIFQT is used in this real case study to elicit each of the software and security requirements and to quantify security requirements which are related to the proposed software (Price Website). Moreover, it is used to make the decisions for all elicited security requirements to prioritize and categorize them in design phase of (SAIFQT). The purpose of building the prototype (Price Website) is to evaluate (SAIFQT), which is used for building the prototype. The next chapter will discuss the findings and evolution method.