

# Building Secure Web-Applications Using Threat Model

**Sobia Usman**

Computer Science Department, COMSATS Institute of Information Technology, Lahore, Pakistan  
E-mail: sobia@ciitlahore.edu.pk

**Humera Niaz**

Computer Science Department, COMSATS Institute of Information Technology, Lahore, Pakistan  
E-mail: humerafaisal@ciitlahore.edu.pk

Received: 29 August 2017; Accepted: 03 December 2017; Published: 08 March 2018

**Abstract**—Ensuring security in web based applications is one of the key issues nowadays. The processes of designing and building a web site have changed. As the online transactions are increasing, increase in type and number of attacks have been observed regarding security of online payment systems. Generally used web development methodologies do not assure security as an umbrella activity. Moreover appropriate threat modeling is also not being conducted against web security objectives. Need of the hour is to have a comprehensive and simple to use web development methodology which caters security throughout the WDLC for web based solutions.

**Index Terms**—Threat, vulnerabilities, STRIDE, DREAD, Security Objectives, Threat Modeling.

## I. INTRODUCTION

The purpose of this study is to examine existing threat modeling process used for the development of web applications and identify the root cause of security threat origin. Moreover we will classify potential security threats according to STRIDE and DREAD model. In this paper we will suggest an easy method for developing Application Security Lifecycle in parallel to the conventional SDLC of web based applications. Furthermore a basic stepwise method will be proposed to check the coverage of security objectives in requirements, use-cases and test cases.

The research paper covers the following

- Identify security objectives against different types of threats faced by web applications
- Design Threat Model based on the security objectives using STRIDE and DREAD model
- Investigate security objectives coverage in requirements, use cases and test cases with reference to the threat models designed

## II. RELATED WORK

Ensuring security in web based applications is one of the key issues nowadays. The processes of designing and building a website have changed. Due to tremendous increase in online transactions, there has been an equal rise in the number and type of attacks against the security of online payment systems. Some of these attacks have utilized vulnerabilities that have been published in reusable third-party components utilized by websites. The different types of vulnerabilities are SQL injection, cross-site scripting, information disclosure, path disclosure, price manipulation, and buffer overflows. Information and path disclosure vulnerabilities typically act as initial stages leading to further exploitation. SQL injection or price manipulation attacks can cripple the website, compromise confidentiality, and in worst cases cause the e-commerce business to shut down completely. It is a good practice to work with security in mind. Every component in the web site, including the Web server and underlying frameworks or platforms, may suffer from security flaws. User authentication, user inputs, remote execution of code and denial of service are common vulnerabilities. Web application security engineering approach aims to design, build and deploy secure Web applications, by integrating security into the development lifecycle and by adapting methodologies which include specific security-related activities [1].

Microsoft uses a specific model to categorize software security threats. It is called STRIDE. The STRIDE Threat Model is a technique used to identify and categorize threats of an application. Each letter in STRIDE acronym specifies a different category of security threat: spoofing identity, tampering, repudiation, information disclosure, denial of service and elevation of privilege. [2]

Microsoft's DREAD model is used to define risk associated with their exploitations by giving points to the threat. DREAD stands for damage potential,

reproducibility, exploitability, affected users, discoverability. The problem with a simplistic rating system is that team members usually will not agree on ratings. To help solve this, new dimensions are added that help to determine the impact of a security threat. These two models i.e. STRIDE and DREAD will be used to classify threats related to our web application under study. [3][4]

Dnyaneshwar K. Patil, Dr. Kailas R. Patil have worked on the detection of the cross site scripting vulnerabilities of the web applications. Their proposed system is capable to discover cross-site scripting in addition to the stored cross-site scripting vulnerabilities. [5] Rupali D. Kombade and Dr. B.B. Meshram have in their paper presented CSRF attack scenario and reasons behind that. They applied CSRF guard and CSRF detector techniques but complete protection against CSRF is not available and techniques require more improvement. [6]

Harish Dehariya, Piyush Kumar Shukla and Manish Ahirwar in their paper have explored SQL injection attack which is a major attack for security of web applications. For security of web application many tools have been developed for detection and prevention. In these most of the tools can attack some types of SQLIAs. Tautology checker, SQL DOM, SQLRand, CSSE, SQL Check and SQLGaurd etc. are the tools which can detect Attacks but not all. Security of web application by SQLIA is an area where a tool requires by which all types of attacks may detect and prevent. [7] Wu Beihua and Wang Yongquan in their paper, their system adopts Web server core embedded technology to embed tamper detection module and application protection module into the Web server, define corresponding strategies for temper-proofing, and realize the real-time monitoring and protection of web pages and the dynamic content in databases.[8]

Abdus Satter and B M Mainul Hossain, in their study, have observed that most of the websites are vulnerable to MITM in compare to XSS, DoS, and SQLIA. These websites share sensitive information without employing any encryption techniques. Most of the websites do not use any certificate while transferring secret information. This security hole paves the way to the attackers to obtain secret information through MITM. Usually, blog sites are considered as common choice for the attackers to perform XSS attack since the users like to visit those sites frequently. Websites developed for business purposes are found mostly protective against this attack due to using different SQL injection and XSS defending frameworks. Four types of attacks are crafted to find different security vulnerabilities. [9]

Pankaj Kumar and C.P. Katti in their paper have worked on the Parallel-SQLIA detector which can detect and prevent all types of SQL injection vulnerabilities as well enhances the overall system performance. Parallel-SQLIA detector simply keeps the information of hot queries and skips the repeated verification; therefore improve the system performance in terms of execution time. [10] Amor Lazzez and Thabet Slimani in their paper have presented a detailed overview about the web

application forensics. First, we defined the web applications forensics as a sub-class of the digital forensics. Then, they presented the appropriate methodology that should be followed to help a successful accomplishment of a forensics investigation of a web application security attack. After that, they discussed how network forensics, digital image forensics, and operating systems forensics may support the achievement of a web application forensics investigation through the provision of further evidence. Next, they presented a detailed presentation of the main considered tools to help a successful accomplishment of a forensics investigation of a hacked web application. Finally, they provided a technical comparison of the main considered web application forensics tools. [11]

Subhash Chander and Ashwani Kush in their paper have discussed certain security issues & vulnerabilities in websites of educational institutes. They have identified various threat levels and high threat level website can be easily compromised by hackers. Stakeholders must take necessary steps to mitigate the risks involved in breach of security of the website. [12]

Vandana Dwivedi, Himanshu Yadav and Anurag Jain, in their paper have reviewed the foremost common existing SQL Injections related problems. [13] Authors have presented a systematic mapping study in order to present the current status of the field, possible gaps and directions for future research. [14] Authors in their paper have overviewed the most critical Web Application vulnerabilities and its fundamental and mitigating solutions, and also presented the common website and web application hacking tools. By recognizing the web application vulnerabilities and solutions, servers could perform better against the attackers and malicious activities. [15]

Apart from others we have presented all-in-one solution. And this solution will be for all kinds of the web application vulnerabilities.

### III. HYPOTHESIS AND RESEARCH SITE

In order to investigate the security objectives coverage in the application under study, a few hypotheses are made. These hypotheses investigate security coverage in existing application, however these hypotheses do not check the validity of the threat model we have proposed for web based solutions.

The hypotheses for the study are derived from the following proposition. Mathematical description is given against each hypothesis.

*P1:*

*There is a relationship between security threats coverage and the number of bugs (Pre-release & Post-release) found in a web application.*

Purpose of this proposition is to analyze the existing security threats coverage of the applications in the requirements, design and test documents. Our objective is to identify the phase in the WDLC where security

threats coverage is lagged, causing major bugs/threats related to security. Furthermore, we will use correlation and tail test to prove the hypothesis.

Following are the hypotheses, which we have assumed for the analysis of the problem.

**A. Hypothesis 1**

Purpose of this hypothesis is to determine, whether requirements cover all the security threats or not. So if the requirements are complete then less number of threats will be faced. Hence quality of the product will increase.

*H0:*

*“There is no relationship between security threats and requirements i.e. every security threat is stated as a requirement in the SRS.”*

$$H0 : \mu_0 \leq \mu_1$$

*H1:*

*“There is relationship between security threats and requirements i.e. every security threat is not stated as a requirement in the SRS.”*

$$H1 : \mu_0 > \mu_1$$

Where  $\mu_0$  = Number of Security Threats  
 $\mu_1$  = Number of Requirements related to security

**B. Hypothesis 2**

The purpose of this hypothesis is to determine if all the security threats catered in the requirements phase are also addressed in the design phase. We intend to determine the relationship of security threats coverage between requirements and design phase.

*H0:*

*“There is no relationship between security threats coverage in requirements and security threats coverage in design i.e. All the security objectives addressed in the requirements phase are also carried to the design phase.”*

$$H0 : \mu_0 = \mu_1$$

*H1:*

*“There exist a relationship between security threats coverage in requirements and security threats coverage in design i.e. not all the security objectives addressed in requirements phase are carried to the design phase.”*

$$H1 : \mu_0 > \mu_1$$

Where  $\mu_0$  = %age Threats handled in Requirements Phase

$\mu_1$  = %age Threats handled in Design Phase

**C. Hypothesis 3**

The purpose of this hypothesis is to determine if all the

security threats catered in the requirements phase are also addressed in the testing phase. We intend to determine the relationship of security threats coverage between requirements and testing phase.

*H0:*

*“There is no relationship between security threats coverage in requirements and security threats coverage in test cases i.e. All the security objectives addressed in requirements phase are also carried to the testing phase.”*

$$H0 : \mu_0 = \mu_1$$

*H1:*

*“There exist a relationship between security threats coverage in requirements and security threats coverage in test cases i.e. not all the security objectives addressed in requirements phase are also carried to the testing phase.”*

$$H1 : \mu_0 > \mu_1$$

Where  $\mu_0$  = %age Threats handled in Requirements Phase

$\mu_1$  = %age Threats handled in Testing Phase

**D. Research Site and Data Collection**

The organization under study is a software development and technology services company. Its working domain lies under the E-commerce and general business application solutions for the capital market, financial, health and industrial sectors. Short details of the organization and the product under study are given below in Table-1 and Table-2.

Table 1. Organization Details

| Organization Details |                                |
|----------------------|--------------------------------|
| Organization size    | 20 (Approximately)             |
| Domain of working    | Business application solutions |
| Standard             | N/A                            |

Table 2. Product details

| Products Details            |   |  |
|-----------------------------|---|--|
| Products under study        | Product A = 7 releases                      |  |
| Products domain             | Product A = Paynet Client Management System |  |
| Average Release duration    | Product A – Release                         | 3 Month  |
| Average number of resources | Product A                                   | Development staff = 13<br>Quality Assurance = 3    |
| Technology used             | Product A                                   | ASP 2/SQL Server                                   |
| Documents Collected         | Product A                                   | SRS<br>Design Document<br>Test Cases<br>Bug Report |
| SDLC followed               | Product A = Spiral Model                    |  |

In order to analyze the hypothesis defined in the earlier section data was collected against seven releases and one project taking into account the following areas:

1. Security Requirements
2. Use Cases incorporating security requirements
3. Data Flow Diagrams
4. Test Cases incorporating security requirements

Above listed information was collected in the following way:

1. *Study of Applications.*

Applications were analyzed to gain complete understanding of the applications requirements, architecture and life cycle methodology opted for development.

2. *Documents Collected*

- Requirement Document
- Design Documents
  - a. Use Case Diagrams
  - b. Data Flow Diagrams
- Test Cases

3. *Interviews & Onsite visits*

Table 3. Interview Details

|                     |  |
|---------------------|--|
| Interviewee         | Team Lead / Project Manager  |
| Nature of Interview | Informal   |
| Purpose             | To get understanding about the working of the applications and details regarding the development of the applications |

IV. THREAT MODELING

This section discusses the steps taken to design a threat model for XYZ Client Management System (XCMS). We have defined security objectives, threat model (application overview, application decomposition, threats, and vulnerabilities) and security objectives coverage analysis in the requirements, design and testing phase of WDLC.

A. *Security Objectives*

There was no specific security objectives defined during the development of this application. However we

can consider following as the implicit security objectives:

- Confidentiality
- Integrity
- Authentication
- Authorization

B. *Threat Model*

1. *Application Overview*

The XYZ Client Management System (XCMS) is the administrative module for the XYZ 2000 application. XCMS is an administrative tool for ABC’s users that will run on Internet. XCMS replaces the Client and Billing System (CABS), Universal Login, and Client Usage Reporting, and integrates their functionality into one Web-based application. XCMS is designed to be easier to use, more efficient, and more flexible than previous XYZ-related administration applications.

2. *Application Assets*

- XCMS User’s login Data
- Contacts personal data
- Client personal data
- System
- Availability of web site
- Services
- Login session
- Trustworthy order information
- Log data
- Access to secure order web page

3. *Application Functionality*

The basic purpose of this application is to provide an administrative interface to the internal ABC Users for managing back office activities for XYZ 2000. In general terms, XCMS includes these main functions:

1. Client Management
2. Product Management
3. Order Management
4. Peer Group Management
5. Reports
6. Data Submission
7. System Management

4. *XCMS Architecture diagram*

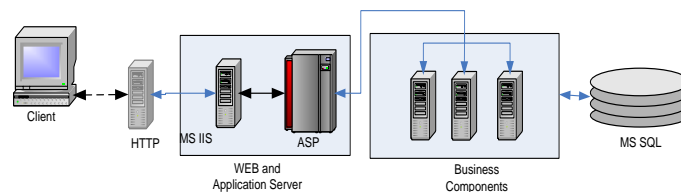


Fig.1. XCMS Architecture Diagram

5. Technologies related to Application

XCMS uses the following technologies:

- Web Server: Microsoft Internet Information Server (IIS)
- Presentation logic: ASP
- Business logic: VB 6
- Data access logic: ADO, SQL Stored Procedures
- Database Server: Microsoft SQL Server 2000

6. Application Roles

Different roles designed for XCMS are:

- Client users
- XCMS administrator
- Server/system administrators
- Contact persons
- Data replication manager

7. Usage scenarios

Following table lists the different usage scenarios of XCMS. Table provides information about the expected use of different modules.

Table 4. Usage Scenarios

| Usage Scenarios | Descriptions   |
|-----------------|--|
| 1.              | The XCMS web application will be installed on a web server. A dedicated team has to maintain security at all times.  |
| 2.              | The XCMS web application will be interacting with database server.   |
| 3.              | In client management module, admin can add client, view detail information, edit, and search and delete clients.   |
| 4.              | In contact management, admin can search contact by client name, add, edit, view delete contacts, assign security, edit security, add log entry, view contact log record, view company contact log.                                 |
| 5.              | In order management, XCMS users can add, edit, search, and delete orders. Users can view orders details, add, edit, delete line items, view order against clients.   |
| 6.              | In parent company management, admin can add, edit, and delete, search and view parent company detail information.  |
| 7.              | In product management, XCMS users can add, edit, view, and delete products groups. Users can add, edit, delete and search products. Users can edit security. Users can add, change and delete price as well as view price history. |
| 8.              | Data base server should not be visible from internet. A firewall should protect direct access.   |
| 9.              | The XCMS web application is configured in a screened subnet where a firewall only allows HTTP and HTTPS traffic towards web server.  |

8. External dependencies

Following table shows the dependencies on other components that can impact security.

Table 5. External dependencies

| ID | Description   |
|----|---|
| 1. | The XCMS web application depends on the security of web server platform.  |
| 2. | The XCMS web application depends on the security of database server.  |
| 3. | The XCMS web application depends on the network between the web server and database server. If the network between the servers is compromised then traffic can be sniffed and altered. Attacked could be launched towards the server. |
| 4. | Session management should be properly configured if not done so then hacker could hijack the user's session.  |

9. Application Security Mechanisms

The most important application security mechanisms known are:

- System should not allow invalid characters, like ' , that can be used to engineer an attack
- System should not transfer critical information, like passwords, in clear text
- All resources in the system will be accessible to the authorized users only
- System will authenticate users before allowing them to perform any activity
- Users are authenticated with Forms authentication.
- Roles are used to authorize access to respective modules.
- Administration can be performed only by physically logging on to the server computer.
- System will show only those modules to the user which are attached to the role of the user.
- System will show only those sub modules to the users which are attached to the selected module and the role of the user.
- System will allow the user to perform only authorize actions in the sub modules of the modules attached to the role of the user.
- The system will catch any unexpected error and handle it through a Handler Class. Error messages will be displayed or written to a log file, depending on the level of the severity of the error.
- In contact management module, users can edit security, assign security, add log entry, view contact log record, and view a company log.
- In product management, XCMS users can edit security.
- Contact management package responsible for the management of Contacts. This would handle adding, deleting, and editing the client contacts and hay office contacts in the system. This will also take care of the Contact Logs for the client contacts.

C. Application Decomposition

1. Trust boundaries

Following table displays the different trust levels describing privileges levels that are associated with entry points and protected resources.

Table 6. Trust boundaries

| ID | Name                          | Descriptions  |
|----|-------------------------------|---|
| 1. | HTTP entity                   | An entity pulling and accessing an HTTP request to the XCMS web application                   |
| 2. | HTTPS entity                  | An entity accessing an HTTP request to the XCMS web application                               |
| 3. | Web server process identity   | The only validated authentication vector that allows web server to and from the database.     |
| 4. | Order Management admin        | Access to secure web pages  |
| 5. | Web server administrator      | An administrator to configure the XCMS web application web server                             |
| 6. | Database server administrator | Server admin responsible for database configuration and user management                       |
| 7. | Data replication manager      | Responsible for data submission module and database level replication activity                |
| 8. | System admin                  | Responsible for system management, user groups, ABC offices, ABC contacts and list management |
| 9. | XCMS admin                    | Responsible for client management, product management and order management                    |

2. Data flows

In XCMS web application the system administrator is responsible for system management. It includes the sub modules like managing ABC contacts information, managing list information, managing ABC offices information and managing user groups. The data replication manager is responsible for replicating the data

on daily basis with XYZ 2000 database and ABC other products database. XCMS administrator is responsible for managing client’s information, managing products information and managing order information. Client users can search the contacts, view their orders and messages related to that. Here the detailed data flow of contact management in XCMS web application has been described. The circles in the middle of DFD shows the processes and the external entities are represented by rectangles. The data store is represented by parallel lines.

- A client user (client contact) logs in to the XCMS system by providing his credentials (user name & password). The login business module authenticates the user by checking credentials in the database. After authentication, the system retrieves user privileges and builds the navigation menu listing only those modules which the user has access to. The user selects Contact Listing module to view a listing of all his peers which he is authorized to see. He clicks on any single contact to see the details of the contact I the system.
- After proper authentication and authorization, the client user opens contact search page. On the contact search page, he provides search information to find his own record in the database. This page calls the business component to retrieve all the contacts in the system based on the search criteria provided. These returned records are displayed in tabular format to the user with options like “View Message”, “View Contact Detail” etc. On this search results page, the user clicks on the “View Message” option to view all the communication that he has done with XCMS internal users.

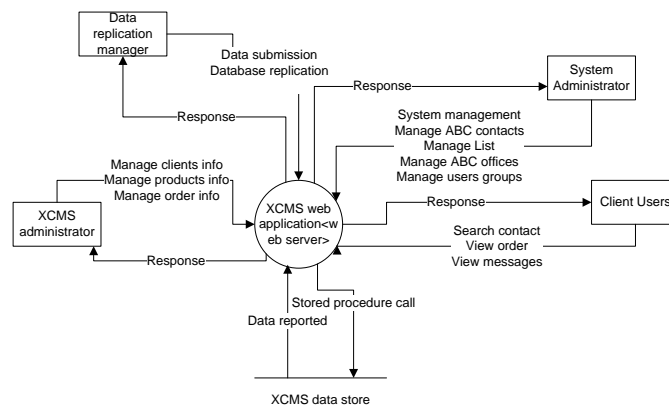


Fig.2. XCMS Data Flow

- The system administrator logs into the system to create a new contact. Login business module checks user credentials in the database and authenticates the user if finds a match. Then the system builds the navigation menu based on the user rights defined in the system. The admin clicks on the “Add New Contact” module to display a

page with input-boxes. The admin provides detailed information about the new contact he wants to create and clicks on the save button when finished. The system validates and saves data in the database.

- The system administrator logs into the system to modify a contact. Login business module checks

user credentials in the database and authenticates the user if finds a match. Then the system builds the navigation menu based on the user rights defined in the system. The admin clicks on the “Search Contacts” module and provides contact information in the search criteria. The page calls business components to search for the contact which returns a list of contacts matching the search criteria. These results are displayed on a page to the user who then clicks on the “Edit Contact” hyperlink against the contact whose information he wants to modify. The admin modifies the information of the contact and clicks on the save button when finished. The system validates and saves data in the database.

### 3. Entry points

The entry point table displays the interfaces through which external entities can interact with the component, either through direct interaction or indirectly supplying it with data. The entry points are linked to the expected trust level that is required for interfacing.

Table 7. Entry Points

| ID | Name                              | Description   |
|----|-----------------------------------|---|
| 1. | Web server listening port (HTTP)  | A web request on Port 80 for normal pages   |
| 2. | Web server listening port (HTTPS) | A web request on Port 443 for secure pages.   |
| 3. | User input page                   | Web pages used to accept client order entry, product groups and product information and feed into the database.   |
| 4. | Login page                        | The logon page, which is accessible to all Internet users. Logon is validated by using client-side and server-side validation controls, together with a common validation library.  |
| 5. | Add & Modify contact pages        | Add & Modify contact pages are available to the admin users only. Login validation page receives all the requests before they are routed to the database and checks whether the user is properly authenticated and has the proper rights to perform the operation he is asking for. |
| 6. | Database stored procedures        | Store and retrieve all data in database related to clients and their orders.  |
| 7. | Database listening port           | The port listening for database requests. The database service is running on that port.   |
| 8. | Web pages on hard disk            | The web pages in www root are entry points. The web server process replies on the request by fetching the html files and sending them back to the requesting entity.  |
| 9. | Welcome page                      | Presentation of the XCMS web application and redirect to the login page   |

### 4. Protected Resources

Following table shows the cross-reference between application assets with minimum trust levels.

Table 8. Protected Resources

| ID  | Name                            | Description   |
|-----|---------------------------------|---|
| 1.  | XCMS User's login Data          | User name and password credentials  |
| 2.  | Contacts personal data          | Private data related to contacts account  |
| 3.  | Client personal data            | Private data related to client account  |
| 4.  | System                          | Assets related to underlining layers of application   |
| 5.  | Availability of web site        | If the website goes down, users cannot place order.   |
| 6.  | Services                        | Assets related to processes that are used to run the XCMS web application   |
| 7.  | Login session                   | The web sessions are assigned to authenticated users.   |
| 8.  | Trustworthy order information   | Integrity should be preserved on the order information  |
| 9.  | Log data                        | In contact management module, admin can add log entry, view contact log report, view company contact log. It comprises of comments. |
| 10. | Access to secure order web page | Only authenticated client should be authorized to access secure order web pages   |

### 5. Exit points

Exit points of XCMS application are listed below:

- The search page, which writes the contact's search string and the corresponding results.
- The contact-detail page which displays contact details.
- The contact-saved page which displays new or modified-contact db saving information.

### D. Threats Identified

According to our analysis following is a list of potential threats could affect XCMS application, defined against the security objectives listed in earlier section:

1. Cross-site scripting
2. SQL injection
3. Canonicalization attacks
4. Query string manipulation
5. Form field manipulation
6. Network eavesdropping
7. Brute force attacks
8. Dictionary attacks
9. Disclosure of confidential data
10. Data tampering
11. Retrieval of clear text configuration secrets
12. Over-privileged process and service accounts
13. Accessing sensitive data in storage
14. Accessing sensitive data in memory (including process dumps)
15. Information disclosure
16. Session hijacking
17. Session replay
18. Revealing sensitive system or application details

- 19. Denial of service attacks
- 20. User denies performing an operation
- 21. Attacker exploits an application without trace

1. Threats Classification

Threat classification has been done on ten threats, from the above listed of threats. Following table discusses the classification of threats with reference to STRIDE model. Classification of threats helps the project team to analyze threats and its potential damage more clearly.

Table 9. Threats Classification

|                       |   |
|-----------------------|---|
| THREAT ID             | 1   |
| Name                  | Hacker gets hold on PCMS user without being logged  |
| Description           | If no audit trail is enrolled over the whole PCMS web application then there will be no evidence in tracing back intrusions. Investigators will have no chance indicating where the attack came from. |
| STRIDE classification | Repudiation   |
| Mitigated             | Yes   |
| Entry points          |   |
| Protected Resources   |   |
| THREAT ID             | 2   |
| Name                  | Crashing the PCMS web application   |
| Description           | A hacker with malicious intent crashes the website which results in denial of service. Clients will not be able to make orders and company will suffer financial losses.                              |
| STRIDE classification | Denial of service   |
| Mitigated             | No  |
| Entry points          | Web server listening port (HTTP)  |
| Protected Resources   |   |
| THREAT ID             | 3   |
| Name                  | Hacker deletes a client or PCMS user account  |
| Description           | By the removal of client or PCMS user accounts, the hacker causes a denial of service. The client or PCMS user will not be able to work with web application.   |
| STRIDE classification | Denial of service<br>Elevation of privileges  |
| Mitigated             | Yes   |
| Investigated Notes    | Only the system administrator can connect directly to the database.   |
| Entry points          | Database stored procedures  |
| Protected Resources   | Client's personal data<br>Contact's personal data   |
| THREAT ID             | 4   |
| Name                  | Unauthorized use of secure client web page  |
| Description           | A malicious hacker who is able to successfully authenticate to the PCMS user web page is able to retrieve client sensitive data.  |
| STRIDE classification | Elevation of privileges   |
| Mitigated             | Yes   |
| Entry points          | Order management web pages  |
| Protected Resources   | Access to order management web pages  |
| THREAT ID             | 5   |
| Name                  | Direct access to database   |
| Description           | Hacker with malicious intent accesses the backend PCMS database directory.  |
| STRIDE classification | Tampering<br>Repudiation  |

|                       |   |
|-----------------------|---|
|                       | Information disclosure<br>Elevation of privilege  |
| Mitigated             | Yes   |
| Entry points          | Database listening port   |
| Protected Resources   | Trustworthy order information   |
| THREAT ID             | 6   |
| Name                  | User data tampering   |
| Description           | A hacker with malicious intent is able to alter the order information send to the PCMS application.   |
| STRIDE classification | Tampering<br>Spoofing<br>Elevation of privilege   |
| Mitigated             | No  |
| Entry points          | User input pages: Client management, product management   |
| Protected Resources   | Client's personal data  |
| THREAT ID             | 7   |
| Name                  | Session ID theft  |
| Description           | A malicious hacker acquires the session ID of another user  |
| STRIDE classification | Elevation of privilege  |
| Mitigated             | Yes   |
| Entry points          | Web server listening port (HTTPS)   |
| Protected Resources   | Order management  |
| THREAT ID             | 8   |
| Name                  | Disclosure of login information   |
| Description           | Hacker gets hold on login credentials.  |
| STRIDE classification | Information disclosure<br>Elevation of privilege  |
| Mitigated             | No  |
| Entry points          | Login page  |
| Protected Resources   | PCMS Administrator login data<br>Client's login data  |
| THREAT ID             | 9   |
| Name                  | Command injection flaws   |
| Description           | A hacker exploits this flaw by injecting malicious commands via the PCMS web application to another business component i.e., database server. By not properly validating the incoming SQL syntax, execution on database may change, resulting in information leakage, the system becomes unstable or could lead to a database crash |
| STRIDE classification | Tampering<br>Elevation of privilege   |
| Mitigated             | No  |
| Entry points          | Login page<br>Usage input page  |
| Protected Resources   | Dataflow access towards backend database  |
| THREAT ID             | 10  |
| Name                  | User data disclosure  |
| Description           | The PCMS web application deals with confidential and private data. If an attacker can sniff user login data, this can lead to elevation of privileges. Disclosing another user's personal data raises privacy issues.   |
| STRIDE classification | Spoofing<br>Information disclosure  |
| Mitigated             | No  |
| Entry points          | User input pages<br>Database stored procedures  |
| Protected Resources   | Contact personal data<br>Client personal data   |



2. Threat Trees

In order to gain a better understanding of the whole occurrence scenario of a threat, threat trees are formed.

Threat 6: User Data Tampering

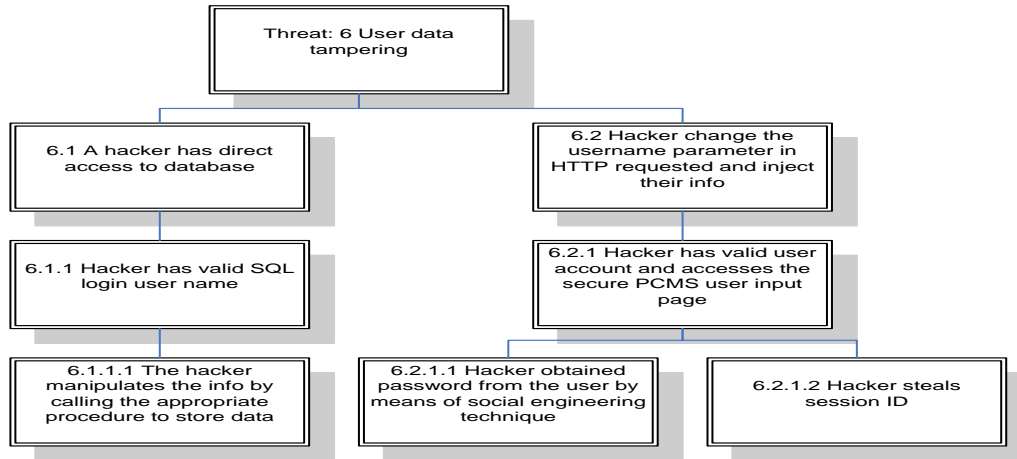


Fig.3. User Data Tampering

Threat 8: Disclosure of Login Information

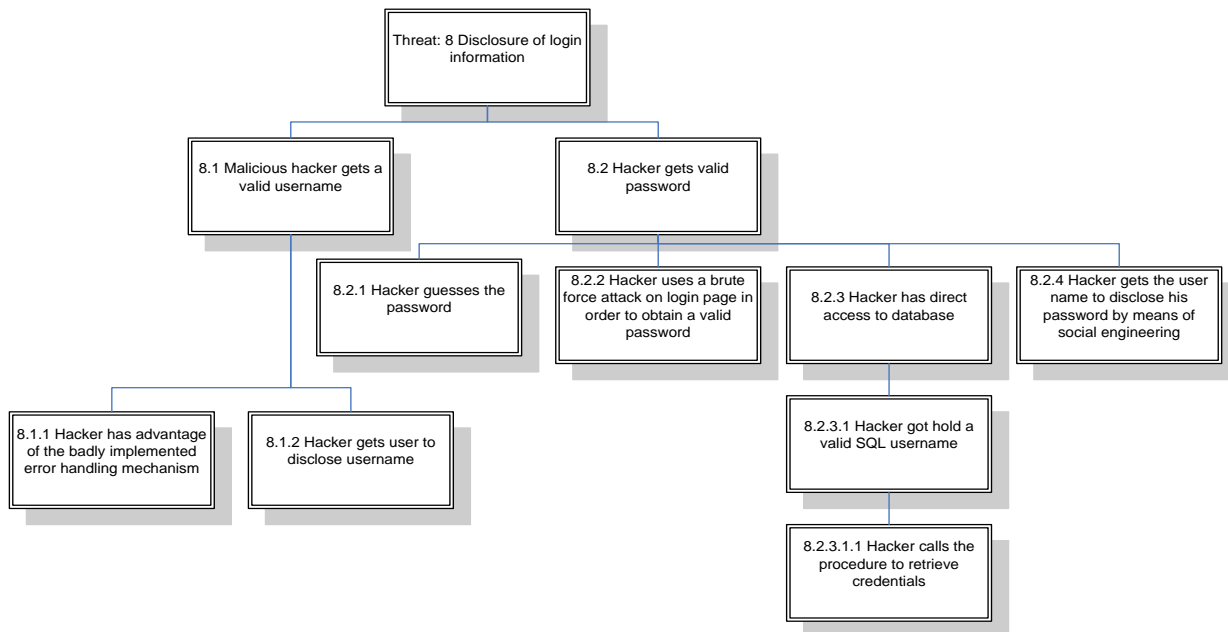


Fig.4. Disclosure of Login Information

E. Vulnerabilities Identified

Following is the complete list of vulnerabilities that could affect the application, defined against the threats listed above:

1. Using non-validated input in the Hypertext Markup Language (HTML) output stream
2. Using non-validated input used to generate SQL queries. Relying on client-side validation
3. Using application-only filters for malicious input
4. Trusting data read from databases, file shares, and other network resources
5. Failing to validate input from all sources including

cookies, query string parameters, HTTP headers, databases, and network resources

6. Storing clear text credentials in configuration files
7. Passing clear text credentials over the network
8. Permitting over-privileged accounts
9. Permitting prolonged session lifetime
10. Relying on a single gatekeeper
11. Failing to limit database access to specified stored procedures
12. Storing clear text configuration data
13. Using over-privileged process accounts and service accounts
14. Storing secrets in clear text

15. Passing sensitive data in clear text over networks
16. Passing session identifiers over unencrypted channels
17. Permitting prolonged session lifetime
18. Revealing too much information to the client
19. Failing to audit failed logons

(related to security)

which shows that there is relationship between security threats and security related requirements i.e. all the security threats are not stated/handled in the SRS.

For *hypothesis 2*, i.e. there is no relationship between security threats coverage in requirements and security threats coverage in design, we have analyzed that

$$\begin{aligned} \%age \text{ Threats handled in Requirements Phase} \\ = \%age \text{ Threats handled in Design Phase} \end{aligned}$$

The above statement shows that all the threats handled in the requirements phase are also handled in the design phase. Moreover, we have also ensured that the same threats are carried forward to the design phase.

For *hypothesis 3*, i.e. there is no relationship between security threats coverage in requirements and security threats coverage in test cases, we have analyzed that

$$\begin{aligned} \%age \text{ Threats handled in Requirements Phase} \\ = \%age \text{ Threats handled in Testing Phase} \end{aligned}$$

The above statement shows that all the threats handled in the requirements phase are also handled in the testing phase. Moreover, we have also ensured that the same threats are carried forward to the design phase.

It is clear from the hypothesis results that the development team ignored a lot of security threats during the development of the XCMS application. Whereas all those security threats once identified by the team during the requirements phase were carried out to the design and testing phase.

## VI. RECOMMENDATIONS

In order to design a secure application, following are the recommendations, which any developer should be familiar with and should employ when creating security strategies: Depend on verified security systems. Never believe external inputs. External systems are not secure, suppose it. Apply the principle of slightest privilege. Decrease available components and data. Do not facilitate services, account rights, and technologies that are not explicitly required. Do not trust on protection by insignificance. STRIDE principles should be followed.

## VII. CONCLUSION AND FUTURE WORK

In this paper we have applied threat modeling for web applications. We have identified security objectives against different types of threats faced by web applications. We have designed Threat Model based on the security objectives using STRIDE and DREAD

In order to prioritize the threats and vulnerabilities, it is essential to analyze the potential damage and probability of occurrence of a particular threat. Based on DREAD model, we classified five of the above listed vulnerabilities.

### 1. Security Lags

After the completion of Threat Model, we will investigate the XCMS application to access the coverage of threats handled in requirements, design and testing phase.

%age coverage of threats handled is calculated as given below, using formulas defined in the research methodology:

$$\begin{aligned} \%age \text{ Threats handled in Requirements Phase} \\ = 5 / 22 * 100 = 22.73\% \end{aligned}$$

$$\begin{aligned} \%age \text{ Threats handled in Design Phase} = \\ = 5 / 22 * 100 = 22.73\% \end{aligned}$$

$$\begin{aligned} \%age \text{ Threats handled in Testing Phase} = \\ = 5 / 22 * 100 = 22.73\% \end{aligned}$$

Where total number of threats is 22

## V. DISCUSSION OF RESULTS

In this section we present the main results of our hypotheses 1, 2 and 3. We have analyzed our data on one application i.e. XCMS (due to unavailability of web-based projects for analysis). Following data was gathered for the XCMS application:

Number of security objectives = 4  
 Total number of threats identified by designing Threat Model = 22  
 Number of Threats handled in the requirement phase = 5  
 Number of Threats handled in the design phase = 5  
 Number of threats handled in the testing phase = 5  
 %age Threats handled in Requirements Phase = 22.73%  
 %age Threats handled in Design Phase = 22.73%  
 %age Threats handled in Testing Phase = 22.73%  
 Number of requirements identified by XCMS team related to security = 5

Based on the above data, for *hypothesis 1* i.e. there is no relationship between security threats and requirements, we analyzed that

$$\text{Number of security threats} > \text{Number of Requirements}$$

model. We have proposed a method to investigate security objectives coverage in requirements, use cases and test cases with reference to the threat models designed. This research can act as basis for further research in many ways:

1. Due to limited applications we could only analyze our threat model on one application; this study can be extended to multiple projects to generalize the results.
2. Moreover the threat modeling process proposed can be applied to real time projects to analyze its effectiveness.
3. Threat modeling process can be generalized to specific types of *e-products* e.g. e-banking, e-commerce etc.
4. Analysis of threats coverage with respect to post and pre release defects (related to security) can also be conducted.

#### REFERENCES

- [1] J.D. Meier, Alex Mackman, Blaine Wastell, Prashant Bansode, Kishore Gopalan, August (2005), "Patterns & Practices Web Application Security Engineering Index", Microsoft Corporation
- [2] Stijn Vande Casteele, (2004), "Threat Modeling for Web Applications Using the STRIDE Model", MIS Thesis, Information Security group, Royal Holloway, University of London
- [3] J. Offutt, March/April (2002), "Quality Attributes of Web Software Applications", IEEE Software: Special Issue on Software Engineering of Internet Software, 19(2):25-32
- [4] Meier, j.D., Mackman, A, Dunner, M., Vasireddy, S., Escamilla, R. and Murukan, A. Ni date.(2003) "Improving Web Application Security Threats and Countermeasures" Microsoft
- [5] Dnyaneshwar K. Patil, Dr. Kailas R. Patil, (2016), Automated Client-side Sanitizer for Code Injection Attacks, I.J. Information Technology and Computer Science, 4, 86-95
- [6] Rupali D. Kombade, Dr. B.B. Meshram,( 2012), " CSRF Vulnerabilities and Defensive Techniques", I. J. Computer Network and Information Security, 1, 31-37
- [7] Harish Dehariya, Piyush Kumar Shukla, Manish Ahirwar, (2016) "A Survey on Detection and Prevention Techniques for SQL Injection Attacks", I.J. Wireless and Microwave Technologies, 6, 72-79
- [8] Wu Beihuaa, Wang Yongquan, (2012), "The Research and Application of Webpage Temper-proofing System", I.J. Wireless and Microwave Technologies, 3, 16-20
- [9] Abdus Satter, B M Mainul Hossain, (2016), "Vulnerabilities Assessment of Emerging Web-based Services in Developing Countries", I.J. Information Engineering and Electronic Business, 5, 1-8
- [10] Pankaj Kumar, C.P. Katti,( 2016), " A Parallel-SQLIA Detector for Web Security", I.J. Information Engineering and Electronic Business, 2, 66-75
- [11] Amor Lazzez, Thabet Slimani, (2015), "Forensics Investigation of Web Application Security Attacks", I.J. Computer Network and Information Security, 3, 10-17
- [12] Subhash Chander, Ashwani Kush, (2013), "Vulnerabilities in Academic E-governance Portals, I. J. Computer Network and Information Security", 3, 56-62
- [13] Vandana Dwivedi, Himanshu Yadav, Anurag Jain, (2014),

- "Web Application Vulnerabilities: A Survey", International Journal of Computer Applications, 1, 108
- [14] Sajjad Rafique, Mamoona Humayun, Bushra Hamid, Ansar Abbas Muhammad Akhtar, Kamil Iqbal, (2015), "Web Application Security Vulnerabilities Detection Approaches: a Systematic Mapping Study", IEEE, 978-1-4799-8676-7
  - [15] Hasty Atashzar, Atefeh Torkaman, Marjan Bahrololum, Mohammad H. Tadayon, (2012), "A Survey on Web Application Vulnerabilities and Countermeasures", IEEE, 978-89-88678-55-8.

#### Authors' Profiles



**Miss Sobia Usman** is Assistant Professor in Department of Computer Science at COMSATS Institute of Information Technology, Lahore, Pakistan. She earned a master's degree in Computer Science in 2003, from Institute of Management Sciences, Pak-AIMS, Lahore. In 2009, she completed her MS (Software Project Management) from National University, FAST, Lahore. Her research interests include software engineering, software project management and software process measurement.



**Ms. Humera Faisal** is Assistant Professor in Department of Computer Science at COMSATS Institute of Information Technology, Lahore, Pakistan. She received Master degree in Computer Science from International University of Islamabad, Pakistan in 2001. She completed her MS (Adhoc Social Networks) from COMSATS Islamabad in 2010. In 2001, she joined the COMSATS Institute of Information Technology as Assistant System Analyst. Her research interests include social networks, Adhoc networks and web semantics and services.

**How to cite this paper:** Sobia Usman, Humera Niaz, "Building Secure Web-Applications Using Threat Model", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.3, pp.52-62, 2018. DOI: 10.5815/ijitcs.2018.03.06