ANALYSIS OF SECURITY VULNERABILITIES IN WEB APPLICATIONS USING THREAT MODELING

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Samuel Sudhakar Kondamarri

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

June 2011

Fargo, North Dakota

North Dakota State University
Graduate School

Title

ANALYSIS OF SECURITY VULNERABILITIES IN WEB APPLICATIONS USING THREAT MODELING

By

SAMUEL SUDHAKAR KONDAMARRI

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Kondamarri, Samuel Sudhakar, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, June 2011. Analysis of Security Vulnerabilities in Web Applications using Threat Modeling. Major Professor: Dr. Dianxiang Xu.

Software security issues have been a major concern to the cyberspace community; therefore, a great deal of research on security testing has been performed, and various security testing techniques have been developed. A security process that is integrated into the application development cycle is required for creating a secure system. A part of this process is to create a threat profile for an application. The present project explains this process as a case study for analyzing a web application using Threat Modeling. This analysis can be used in the security testing approach that derives test cases from design-level artifacts.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Dianxiang Xu, for his immense support and guidance. My sincere thanks to Dr. Hyunsook Do, Dr. Kong and Dr. Xiwen Cai for serving on the committee.

I would also like to thank my parents and brothers for their support and encouragement. Finally, I would like to thank my friends, especially Manu Bhogadi, for their continuous encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1. Background and problem definition

Software security issues have been a major concern to the cyberspace community. These issues include attacks that deny service, corrupt data, and disclose confidential information. An attempt to make computer resources unavailable for the intended users is termed 'denial of service'. Corruption that occurs during the transmission, retrieval, or processing, introducing unintended changes to the original data is called 'data corruption'. An application usually consists of confidential information such as username, password, credit card details, and price tags. This crucial information, if disclosed leads to a compromised system.

It has been well-recognized that security issues should be considered as early as possible during the software development life cycle. In particular, secure design is critical for the success of secure software development because design level vulnerabilities are among the hardest defects to handle. In this direction design for security (e.g., design-level Threat Modeling) has recently become a widely accepted practice for building dependable software systems in a cost-effective way. However, secure design does not guarantee secure implementation as vulnerabilities can be introduced in the implementation process.

1.2. Related work

Software testing for security aims at finding security vulnerabilities by executing the software with test cases [13]. In order to find security vulnerabilities we need to test the "Presence of an intelligent adversary bent on breaking the system" [22]. Threat

Modeling has emerged as a viable practice for secure software development. It is the process of producing a simplified, abstract description of how an adversary would perform potential attacks or pose security threats to the system. It can be conducted at various levels of abstraction and granularity or in different development phases, such as requirements analysis, design, implementation, and even testing [26].

While the importance of trustworthy software systems has been well-recognized and tremendous effort has been devoted to enhancing cyber security, companies have still suffered from various cyber-crimes. For instance, one study performed by the WebCohort's Application Defense Center from 2000 to 2004 reported that at least 92% of web applications are vulnerable to some form of hacker attacks [24]. A more recent survey performed by Cenzic [30] reported that the number of attacks continues to increase, and hackers get more sophisticated and organized.

Several notations have been proposed for Threat Modeling, such as threat trees (a variation of fault trees for safety analysis) [7, 11], threat nets (based on Petri nets) [14, 5], and misuse cases (based on use case modeling) [1, 23]. Threat models can be used to generate security tests for exercising whether the implementation is resistant from the identified security threats.

For example, Wang et al. [15] have proposed a threat model-driven security testing approach for detecting undesirable threat behavior at runtime. This approach, utilizes UML sequence diagrams to model threats to security policies. A set of threat traces is extracted from a design level threat model. Each threat trace is an event sequence that should not occur during the system execution; otherwise it becomes a security attack. On the other hand, Microsoft's Threat Modeling approach is based on decomposition of

an application with data flow diagrams. Security threats to the application are modeled with threat trees. The goal is to automatically generate security tests from threat trees. Martin and Xie [8, 9, and 10] have been investigating techniques for test generation from access control policy specifications written in XACML (OASIS eXtensible Access Control Markup Language). They have defined policy coverage criteria [8] and a mutation testing framework for XACML access control policies [10]. To generate tests from policy specifications, they synthesized inputs to a change-impact analysis tool [9]. Masood et al. [2, 3] investigated a state-based approach to test generation for role based access control (RBAC) policy. Their approach first constructs a finite state model of the RBAC policy and then drives tests from the state model. In contrast to all the above approaches, this paper investigates test generation from threat trees.

1.3. Objective

The main objective of this paper is to find a way to analyze a software system so that it can be fully tested at the end of the implementation process which includes two steps: (i) to come up with artifacts of secure design (e.g., threat models) for testing the resultant implementation, (ii) to come up with an enumeration of threats to that particular implementation. To achieve the first step Data flow diagrams, UML diagrams and Flow charts were considered. To enumerate the threat trees, general threats that are available and known vulnerabilities were considered. Further, this two-step process is applied to the applications osCommerce and Drupal.

1.4. Approach

This paper presents an approach to improve software security involving software testing. To date, researchers have developed various security testing techniques. These include techniques that generate test cases or identify vulnerabilities focusing on specific attacks; such as SQL injection or cross-site scripting; generate test cases using use case modeling; generate test cases from control policy specifications.

A security testing technique that is considered in the paper 'Security threat' consists of following activities: (1) building threat trees; (2) generating security tests from threat trees; (3) generating test inputs including valid and invalid inputs; and (4) assigning actual inputs.

First part of the security testing technique, building threat trees is described in this paper. Threat trees can be generated using Threat Modeling. There are three general approaches for Threat Modeling, they are:

Attacker centric: Designers look for the various kinds of attacker motivations.

Asset centric: Starts from assets entrusted to a system like personal information.

Design centric: It is a defensive perspective of Threat Modeling.

The first two approaches are adversarial perspective of Threat Modeling which can be achieved only after implementation is finished. That means if we find any threats in the implementation there will be redesign and rework. Whereas, in the last approach, consideration is given to possible threats and the design is done according to the already known mitigation processes.

In this paper, attacker and asset centric approaches were considered because the applications used for the case study are already implemented and readily deployed web

applications. Consideration was not given to design centric approach because it is good for the applications that are in the process of building but not for the applications that are already built.

As stated in the book 'Threat Modeling' [11], it is defined as a method of assessing and documenting the security risks associated with an application. This methodology can help development teams identify both the security strengths and weaknesses of the system and can serve as a basis for investigating potential threats, testing and investigating vulnerabilities. A threat is a goal an adversary might try to achieve to abuse an asset in the system. Vulnerability is a specific way that a threat can be exploited through an unmitigated attack path.

Therefore, Threat Modeling is, in essence, the act of creating a security design specification and later testing that design. It is integral to the development of any secure system. If performed thoroughly and by a standard methodology, Threat Modeling can improve the built-in security of a system as well as user confidence in that security. Threat Modeling strives to accomplish this heightened security by,

1. Understanding the threat profile of a system
2. Providing a basis for secure design and implementation
3. Discovering vulnerabilities
4. Providing feedback for the application security life cycle
5. Defining a set of possible attacks to consider

Threat Modeling is based on the notion that any system or organization has assets of value worth protecting, these assets have certain vulnerabilities, internal or external

threats exploit these vulnerabilities in order to cause damage to the assets, and appropriate security countermeasures exist that mitigate the threats.

Of all the vulnerabilities identified in Web applications, problems caused by unchecked input are recognized as being the most common. To exploit unchecked input, an attacker needs to achieve two goals: Inject malicious data into Web applications and Manipulate applications using malicious data.

While the former includes Common methods such as: Parameter tampering, URL manipulation, Hidden field manipulation, HTTP header tampering and Cookie poisoning, the latter includes methods such as: SQL injection, Cross-site scripting, HTTP response splitting, Path traversal, and Command injection.

According to Scott W. Ambler five aspects of security Threat Modeling are:

1. Identifying threats
2. Understanding the threats
3. Categorizing the threats
4. Identifying mitigation strategies
5. Testing

Model diagrams such as data flow diagram (DFD), UML diagram or flow chart, can be used to identify entry points to a system such as data sources, application programming interfaces (APIs), web services, and the user interface itself. In order to identify threats we should add privilege boundaries to the model diagram.

To understand the potential threat at an entry point, we must identify any security-critical activities that occur and imagine what an adversary might do to attack or misuse a system.

Security threats can be categorized on the basis of STRIDE classification. STRIDE was documented in Writing Secure Code, Second Edition (Microsoft Press, 2003), by Michael Howard and David LeBlanc. STRIDE stands for S-Spoofing, T-Tampering, R-Repudiation, I-Information disclosure, D-Denial of service, E-Elevation of privileges. Ultimate goal is to find the security vulnerabilities.

Spoofing allows an adversary to pose as another user, component, or other system that has an identity in the system being modeled. Tampering is the modification of data within the system to achieve a malicious goal. Repudiation is the ability of an adversary to deny performing some malicious activity because the system does not have sufficient evidence to prove otherwise. Information disclosure is the exposure of protected data to a user that is not otherwise allowed to access that data. Denial of service occurs when an adversary can prevent legitimate users from using the normal functionality of the system. Elevation of privilege occurs when an adversary uses illegitimate means to assume a trust level with different privileges than he currently has.

To determine how to mitigate a threat, a diagram called threat tree can be created. Every path through the threat tree that does not end in a mitigation strategy is a system's vulnerability. All these threat trees form the threat profile of a system which becomes a plan for testing.

1.5. Results

The main result of this research work is an enumeration of threat trees for osCommerce and Drupal. Enumeration of threat trees can be divided into 6 categories. Categorizing the threats helps us to better understand the threat profile of a system. The

case study is useful for generation of the test sequences which is the next step in the process to the software security technique explained above. For osCommerce, 12 threat trees were developed and for Drupal, 5 threat trees were developed. Threats that are identified in osCommerce are considered in our study of Drupal. Found out that some of the threats that are identified in osCommerce are applicable to Drupal and the others are not applicable. One big reason for that is the design of Drupal. In Drupal, they did not include session information in the URL.

Functional models that are developed helps in future reference and also in different kinds of research like the process that is explained in the paper "A model-based approach to Security Testing" and in "Security test generation using Threat trees". For osCommerce a Data flow diagram and a flow chart are developed and for Drupal a Data flow diagram is developed. This case study gives all the details of where exactly the system is, in the security perspective.

The rest of the paper is organized as follows. Section 2 discusses the approach to Threat Modeling. Section 3 deals with the application osCommerce and the threat trees for osCommerce. Section 4 deals with the web application Drupal and the threat trees for Drupal. Section 5 presents conclusions and discusses future work.

2. APPROACH TO THREAT MODELING

2.1. Overview of the Threat Modeling process

The main objective of this paper is to analyze a web application using Threat Modeling. Threat Modeling by Frank Swiderski and Window Snyder [11] comprises of three high level steps: understanding the adversary's view, characterizing security of the system, and determining threats. Threat Modeling takes an outside-in approach by an adversary when compared to a developer who takes an inside-out approach.

Adversary's view entails enumerating the entry points and assets, as well as cross referencing them with trust levels.

Entry points are locations where data or control transfers between the system being modeled and another system. Examples for entry points are open sockets, remote procedure calls (RPC) interfaces, web services interfaces, and data being read from the file system. Assets are resources for the system under consideration and can be tangible, such as a process token, or more abstract, such as data consistency (a string class that maintains a length field). Trust levels of entry points and assets are cross-referenced.

Privileges for an external entity to legitimately use an entry point or functionality at the entry point is defined by an entry point. An entry point also dictates which assets external entities should normally be allowed to access or affect in some way.

Characterizing security of the system involves bounding the threat model, gathering information about dependencies that are critical to security, and understanding the internal workings of the system. This characterization provides the necessary information for people to understand the threat model. Modeling the system helps the

teams identify design-specific and implementation-specific threats. To characterize a system's security level, one must

1. Define use scenarios

Development teams must ask themselves how the component or system will be used. Conversely, the teams can ask themselves how the component or system will not be used. Use scenarios bind the Threat Modeling discussion, point out situations beyond the security architecture. Use scenarios can be used as mitigation later in the process.

2. Identify assumptions and dependencies

Development teams should collect information such as external dependencies, external and internal security notes, and implementation assumptions. External dependencies document contingencies on other systems that affect the security of the system being modeled. External and internal security notes provide information to the system user and the person reading the threat model, respectively, which is critical to understanding the security of the system. Implementation assumptions provide security information to the developer that is critical to applying parts of the system in a secure manner.

3. Model the system

Data flow diagrams or other diagrams (DFD's), such as process models, are used to understand the actions a system performs at a given entry point. DFD's are visual representations of how a system processes data. These diagrams answer questions such as: How are protected resources affected? Where could an external entity manipulate as asset? What processing occurs beyond an entry point? What action is taken on behalf of the external entity or what transformation is performed on the data supplied? These

answers help development teams understand what an adversary might do at a given entry point.

Enumerating threats creates a threat profile for a system, describing all the potential attacks that architects and developers should mitigate against. The security of a system can be expressed in terms of threats with appropriate mitigation vs. total threats, taking into account the severity of the threats with insufficient mitigation. To create a threat profile for a system, development teams must

1. Identify threats

For each entry point, the development team determines how an adversary might try to affect an asset. Based on what the asset is, the team predicts what the adversary would try to do and what his goals would be this is where the STRIDE classification system comes into play [11].

2. Analyze threats

Development teams model threats to determine whether they are mitigated. Using threat trees, a development team can decompose a threat into individual, testable conditions. Threats that are not mitigated become vulnerabilities—security bugs that must be remedied [11].

2.2. Function modeling

Representation of a system using Data flow diagrams has been described but exclusive use of DFD's in a threat model is not necessary. Some systems are better modeled using other standards like UML or flow charts. Emphasis is on getting to understand how the system under consideration works and how its subsystems fit

together. Data flow diagrams, UML diagrams, flow charts and their advantages are described below.

2.2.1. Data flow diagram

Any business application can be clearly represented using Data flow diagrams. This technique starts off with an overall picture and continues by analyzing each of the functional areas of interest. This analysis can be carried out to precisely the level of detail required. It is a top-down expansion technique to conduct the analysis.

A business model comprises of one or more data flow diagrams [29]. This technique starts off with a context diagram, which is a simple representation of the entire system. This is followed by a level 1 diagram, which provides an overview of the major functional areas of the business. The level 1 diagram identifies the major business processes at a high level and any of these processes can then be analyzed further, giving rise to level 2 business process diagrams. This process of more detailed analysis can then continue through level 3, 4 and so on.

Data flow diagrams are used in Threat Modeling to better understand the operation of a system. They provide a visual representation of how the system processes data. This allows the system to be modeled by focusing on transformations and processes applied to data and requests an adversary might supply. DFDs are also used later in the Threat Modeling process when the document is reviewed or used by security testers [11].

Advantage of using DFD's is that they are hierarchical in nature. Data flow diagramming focuses on data as it moves through the system and the transforms that are applied to that data [11].

2.2.2. UML

Unified modeling language (UML) is a standardized general-purpose modeling language and is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development. UML combines techniques from data modeling, business modeling, object modeling, and component modeling. UML 2.2 has 14 types of diagrams divided into two categories. Seven diagram types represent structural information, and the other seven represent general types of behavior, including four that represent different aspects of interactions.

Structure diagrams emphasize the things that must be present in the system being modeled. Since structure diagrams represent the structure, they are extensively used in documenting the architecture of the software systems. Class diagram, Component diagram, Composite structure diagram, Deployment diagram, Object diagram, Package diagram, Profile diagram are the structure diagram type.

Behavior diagrams emphasize what must happen in the system being modeled. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe functionality of software systems. Activity diagram, UML state machine diagram, Use case diagram, Communication diagram, Interaction overview diagram, Sequence diagram, Timing diagram are the behavior diagram type.

The degree of detail that is needed for the application analysis was not adequate when one of these 14 types of UML diagrams were used. So, decision was taken to draw a flow chart for the system under consideration.

2.2.3. Flow chart

Flow chart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Data flows are not typically represented in a flowchart, in contrast with data flow diagrams; rather, they are implied by the sequencing of operations. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

2.3. Threat trees

Threat trees are used to determine whether the conditions necessary for a threat to be realized exist and are unmitigated. A threat tree consists of a root node, or threat, which has one or more child conditions that must be true for an adversary to realize the threat. Any child condition can in turn have one or more child conditions. Two or more conditions at the same level and sharing the same parent node can be combined, resulting in an AND relationship; otherwise, an implicit OR condition exists. Determining whether one or more vulnerabilities are associated with a threat is simply a matter of starting at a leaf condition (a node in the threat tree with no child nodes) and following it up to the root threat. If a path not broken by a mitigated node or a broken AND condition exists, a vulnerability exists [11].

When analyzing threats by using threat trees, the conditions in the tree have mitigation associated with them. This is because the conditions represent specific actions or situations that must occur, while the threat is less concrete. The conditions also have a risk associated with them, for which Microsoft often uses the DREAD rating

system. DREAD stands for Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability [11].

3. THREAT MODELING OF OSCOMMERCE

3.1. Introduction to osCommerce

osCommerce is an online shop e-commerce solution that offers a wide range of out-of-the-box features that allows online stores to be setup fairly quickly with ease, and is available free as an open source based solution released under the GNU General Public License. It can be used on any web server that has PHP and MySQL installed [20].

osCommerce was started in March 2000 in Germany by project founder and leader Herald Ponce de Leon as The Exchange Project. Since 2000 it has matured to a solution that is currently powering 12,348 registered live shops around the world [20]. osCommerce has attracted a large and growing community that consists of over 247, 600 store owners, developers, service providers, and enthusiasts who support and work with each other on their online business. To date there are over 6,500 ad-ons available for free to customize osCommerce online merchant online stores that help increase sales [20].

Today, osCommerce has been taken to the next level, moving towards an e-commerce framework solution that not only remains easy to setup and maintain, but also making it easier for store administrators to present their store to their customers with their own unique requirements [20].

It is a complete online store solution that contains both a catalog frontend and an administration tool backend which can be easily installed and configured over a web-based installation procedure [20].

3.2. Function modeling of osCommerce

The main artifact for analyzing osCommerce is a flow chart. Normal flow of the Threat Modeling uses a data flow diagram. For our experiment, a decision has been taken to use a data flow diagram and also a flow chart because the information that a data flow diagram gives is not enough to develop the threat trees for osCommerce. Dataflow diagram for osCommerce is as shown below.

Figure 1 represents Level 1 data flow diagram. This is the next level diagram to context diagram. A decision has been taken not to go ahead with the detailed analysis (which includes Level 2, 3, 4 and so on) because, the details that are provided by the diagram are not appropriate for this particular application, osCommerce.

Customer is the main external entity who provides input to the processes. For Login process, input is username and password and output is acknowledgement that takes customer to the home page.

Shopping cart process holds all items together until the items are taken to the checkout. Input is addition or deletion of items and also updating of the cart contents whenever changes are made to it. Output is the cart contents.

Checkout process has cart contents as input and total price including taxes and shipping charges as output. Checkout process sends user information into the database and gets back the saved addresses and credit card information. Payment process is a multiple process because it can be done in two ways. One is just paying by credit card and the other is cash on delivery. The input for it is the total price and credit card information, and the output is full details of the order including the shipping address, billing address, charges and the cart contents. And the credit card details are then

forwarded to the credit card company or a third party for verification and then the approval statement is sent to the shopping cart company. Checkout process which has complete information of the order sends that information into the database and gets the confirmation back. Confirmation process also sends an Email to the customer soon after the order is confirmed.

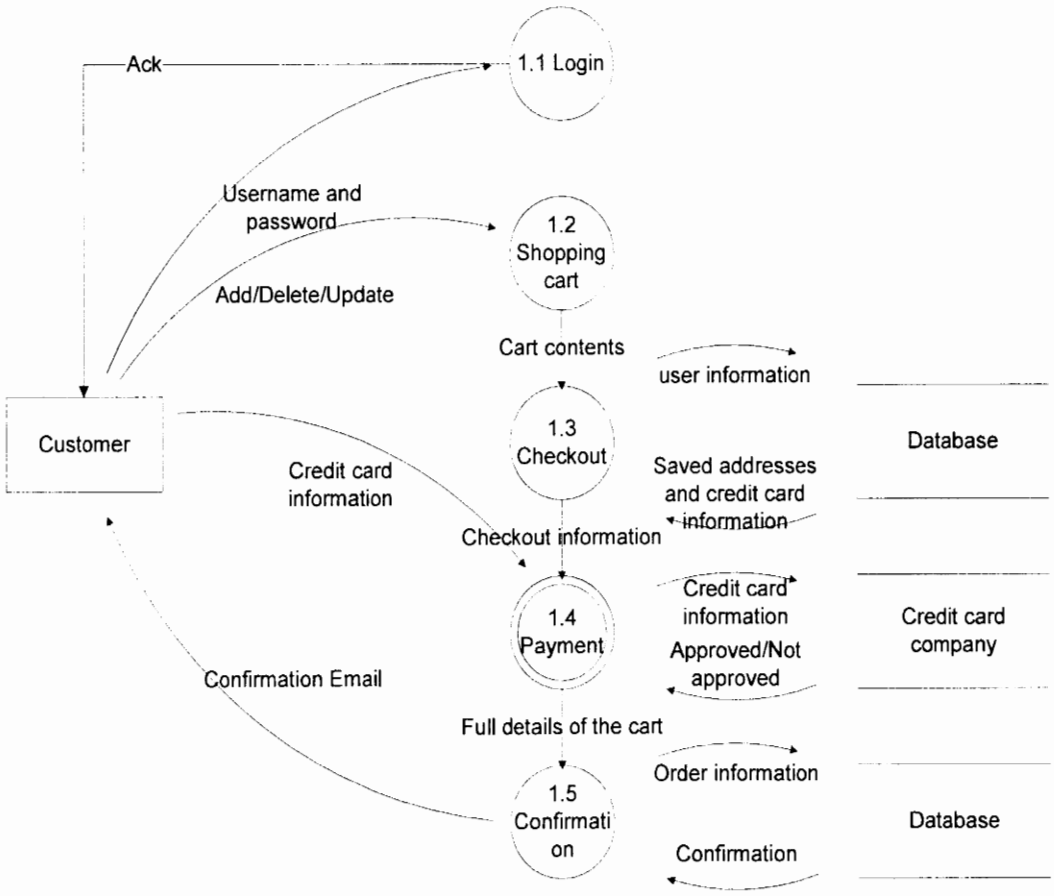

Figure 1. DFD for osCommerce

Figure 2 is the flow chart for osCommerce. It has all the details that were needed for the case study. The attacker paths can be clearly seen in the flow chart. Though some details are definitely missing, the minimum required details are present.

3.3. Threat trees for osCommerce

The threats to the application are listed here in a series. These threats do not imply vulnerabilities. Rather, they are goals that a malicious external entity might have when attacking the system. Each threat is organized in a way to comprise Name, STRIDE classification, Known mitigation, Assets, Entry points, Description and Threat trees.

ID: 01

Name: Gaining access to admin files

STRIDE classification: Tampering and Elevation of privilege

Known mitigation: An authentication layer is added to deny access to admin files.

Assets: Admin files

Entry points: URL construction which is also called as layered entry point

Description: The attacker using brute force gets the information related to the directory construction. And then using that information attacker gets to the admin files of the system. In osCommerce, it is built in such a way that the admin files can be accessed if we know the directory structure. It does not have a special login for the admin files.

Threat tree: Figure 3 shows the threat tree.

ID: 02

Name: Login information vulnerability

STRIDE classification: Information disclosure

Known mitigation: If a session is restricted to one IP address, even if the session information is shared along with the URL, the attacker cannot access the account.

Assets: A particular user's account

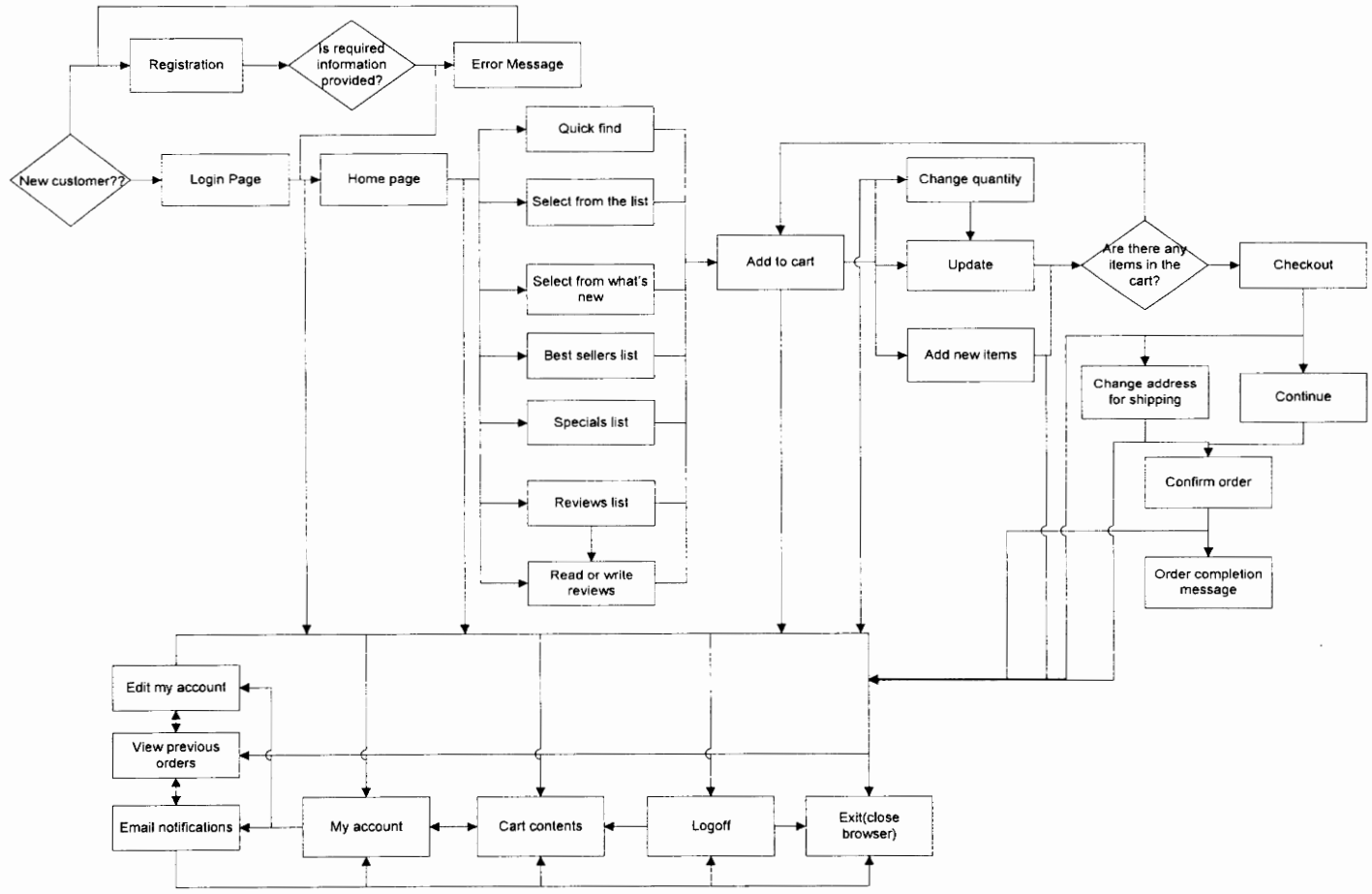

Figure 2. Flow chart for osCommerce

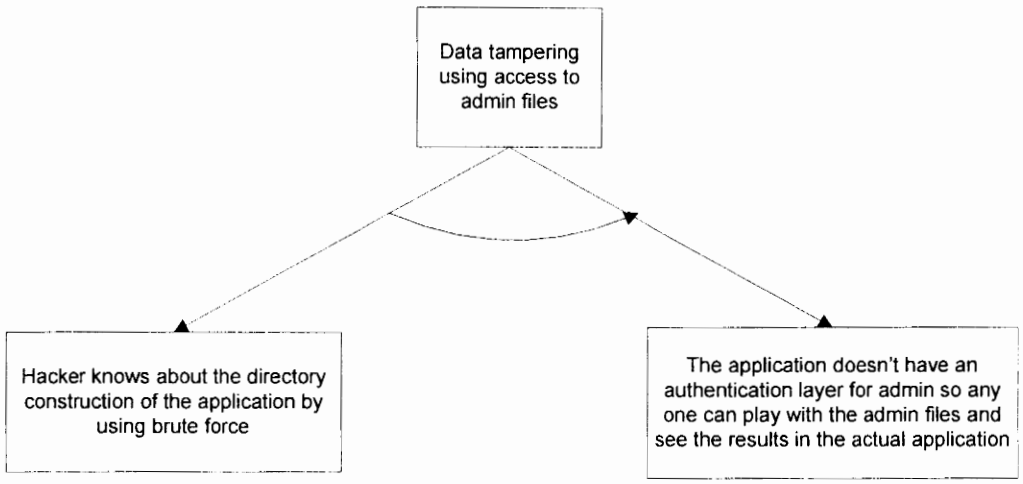

Figure 3. Gaining access to admin files

Entry points: Login page, referrer log

Description: The attacker has the session information so gets access to the victim's account. Once he has the access he goes and changes the username by selecting the 'view or change information' option. Once the username has been changed he can go ahead and change the password. He can change the password separately as well.

But if there is a situation where the actual user (not the attacker) logs out soon after the username has been changed, then the attackers session expires as well. In that case he can actually use the changed username and provide the web application with a new email address where the application will send a system generated password to that email address. And now the attacker can login to the system with the new login information and he can also change the system generated password to the password of his choice. Now, he has control over that account completely.

Threat tree: Figure 4 shows the threat tree.

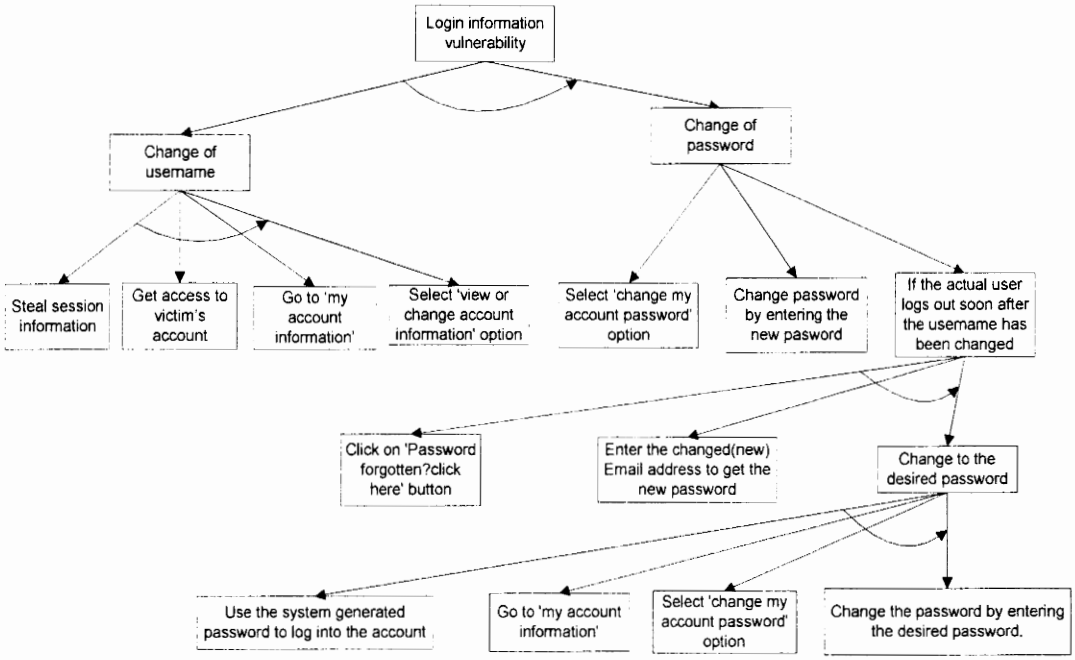

Figure 4. Login information vulnerability

ID: 03

Name: Data Modification

STRIDE classification: Elevation of privilege

Known mitigation: A validation method needs to be added so that the data that is provided by the user does not change the information provided to the user.

Assets: Information like item name, item price and item date of purchase

Entry points: Login page

Description: Adversary can change the item information. For modifying the item price, the adversary follows the following steps.

Select an item (it can be done in 6 different ways), view and save the source code as a text file, find the price tag, change the price tag to the desired amount, save the text

file, reload the text file and complete the order. Completing the order has three steps. The steps are: checkout, provide the payment information and confirm the order.

For modifying the item itself, the adversary follows the following steps. Remember the name of the item which is at a higher price than the one that he is planning to buy, Select an item, view and save the source code as a text file, find the item tag and replace it with the one that has been remembered, save the text file, reload the text file and complete the order. Completing the order has three steps. The steps are: checkout, provide the payment information and confirm the order.

For modifying the date of purchase, the adversary follows the following steps. Select an item, view and save the source code as a text file, find the date of purchase and change it to the desired date, reload the text file and complete the order. Completing the order has three steps. The steps are: checkout, provide the payment information and confirm the order.

Threat tree: Figure 5 shows the threat tree.

ID: 04

Name: Denial of service

STRIDE classification: Denial of service

Known mitigation: A module can be added to the application where it can prevent login process automation. One way to do it is by giving distorted text that only a person can decode and enter the correct text. This method holds good for all the automation attacks.

Assets: Server which provides services to the users

Entry points: Login page

Description: Denial of service in this application can be done in 4 ways.

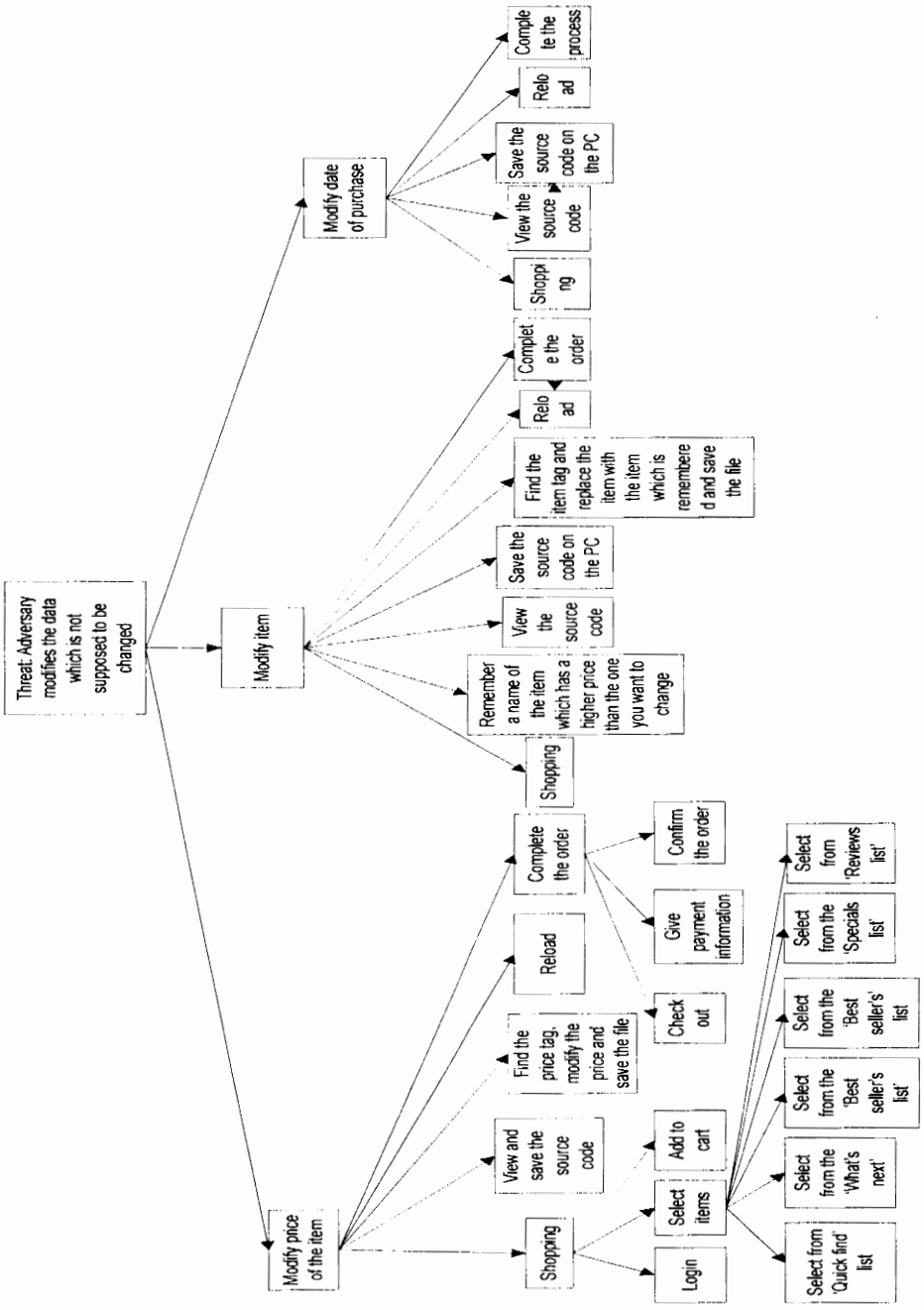

Figure 5. Data modification

Create multiple users: First step by an attacker is to automate the process for creating multiple new accounts. Second step is to apply that to osCommerce.

Password recovery process attack: If there is only a single email ID available, then automate the process so that osCommerce keeps recovering the password for that particular email ID. If there are multiple email addresses available, automate the process so that it uses each email ID for certain number of times and then goes to the next one and continues until the end of the list.

Login process attack: This attack should be done in two steps. The first step involves username and the second step involves the password.

Step 1: This can be done in two ways. First way is to enumerate all the fictitious email addresses. Second way is to get the email address through steal session information vulnerability.

Step 2: This can be done by applying the dictionary attack just for the password field and use the above process for the username.

Starvation attack: Starvation attack can be done in two ways. The first way is to automate the procedure for creating new accounts and then apply it to osCommerce. The second way is to try buying one single item by all the fictitious users.

Threat tree: Figure 6 shows the threat tree.

ID: 05

Name: User Credentials Disclosure

STRIDE classification: Information disclosure

Known mitigation: A validation method can be added to mitigate the SQL injection. Brute force attack can be mitigated by adding another security layer. Example for adding another security layer is by saving user's security question. A security question can be logical information specific to the user.

Assets: Particular user's account which is under attack.

Entry points: Login page and URL construction

Description: SQL injection can be done in two ways as explained below.

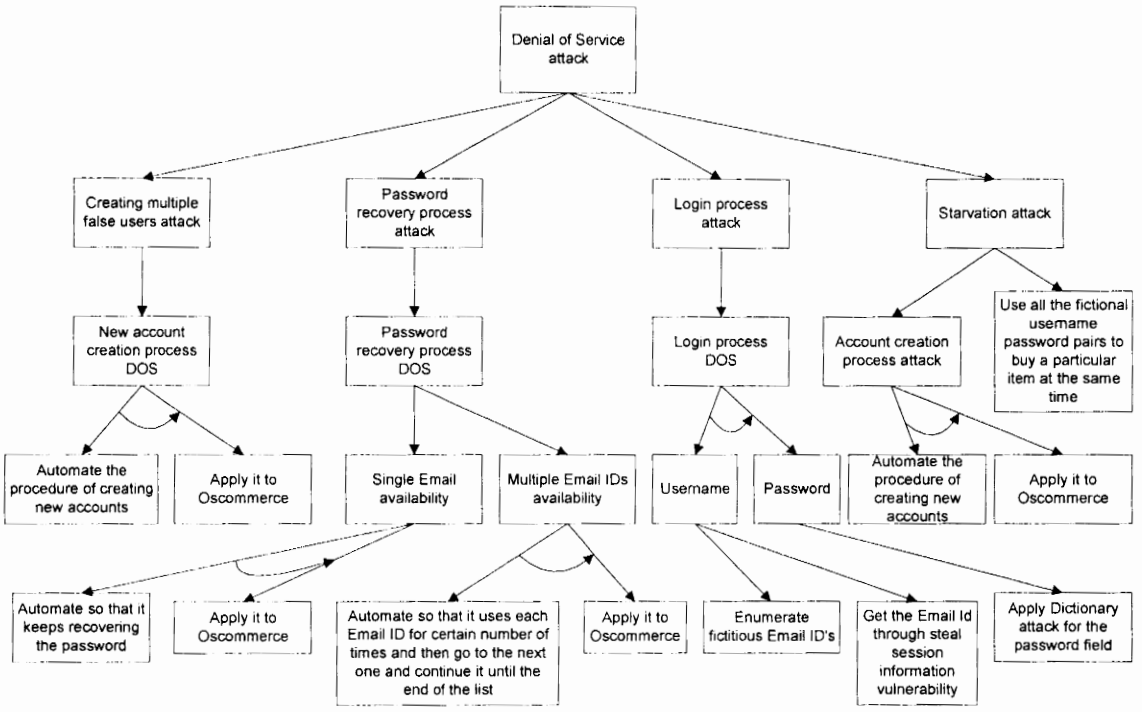

Figure 6. Denial of service

Adversary gets a valid username and password: To get a valid username, adversary uses error string from the login page to determine username validity or adversary gets user to disclose the username. To get a valid password adversary calls retrieve credentials procedure or uses brute force attack on the login page to guess a user's password or guesses the password or gets the user to disclose the password.

Adversary can get in using SQL injection attack. Save the source code on the disk, make changes, reload and give username and password.

Threat tree: Figure 7 shows the threat tree.

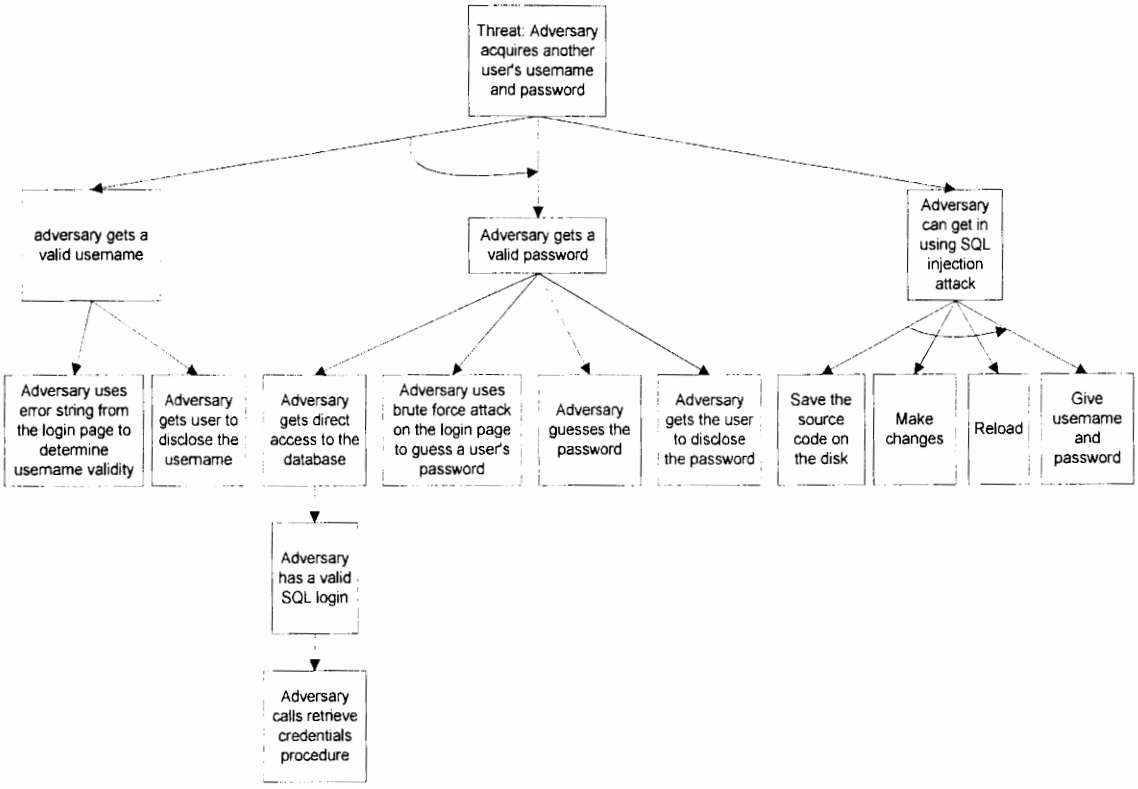

Figure 7. User credentials disclosure

ID: 06

Name: Session ID Exhaustion

STRIDE classification: Denial of service

Known mitigation: A particular IP address is restricted to have a single session ID.

Assets: Server which provides the session information

Entry points: Login page, catalogue page

Description: Visit the catalog page, click on log yourself in, click on the new tab button in your browser, continue the above process until you reach the upper limit of the number of tabs that can be opened in one browser window, open a new browser window and continue above process until you get to a point where session ID's are exhausted,

when the storage capacity reaches maximum and when the processing time is substantially reduced.

Threat tree: Figure 8 shows the threat tree.

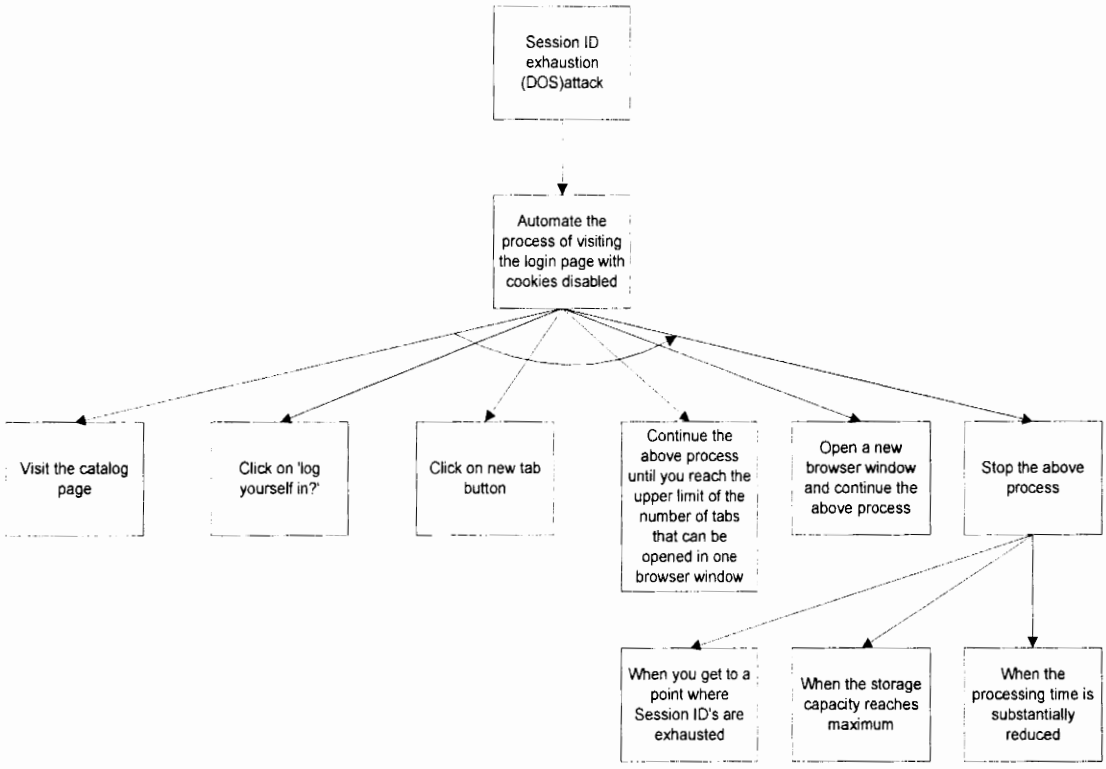

Figure 8. Session ID exhaustion

ID: 07

Name: Session information disclosure

STRIDE classification: Denial of service

Known mitigation: If a session is restricted to one IP address, even if the session information is shared along with the URL, the attacker cannot access the account.

Assets: Referer log page and server which serves the session information.

Entry points: Login page

Description: User (victim) logs-in, the attacker goes through the steps below.

Get access to the referrer log through directory traversal vulnerability present in osCommerce and fix the session.

Access to the referer log is obtained if we follow the following steps: use brute force to find the file names, use'../../' format to reach the location intended, read and download the files on the server.

Session fixation is a three step process. First step is to search for the recent log in the referer log file and steal the session information. Second step is to build a trap and third one is to lure the victim into the trap.

Building a trap can be done in two ways. One of the ways is to design a URL. URL should be exactly the same apart from the session ID. Include an arbitrary session ID. The other way to design a URL is to get a session ID from the server referer log page and include that in the URL.

Lure a victim into the trap can be done in 2 ways.

1. Send an email with a URL which was built that includes the session information, wait until the user logs into his account using the URL being sent through the email, copy and paste the exact URL in the browser to access the information of the victim.
2. User enters the trap, Login information is provided,
3. Copy and paste the exact link from the referer log to get access to the actual users account.

Threat tree: Figure 9 shows the threat tree.

ID: 08

Name: Transaction Resource

STRIDE classification: Denial of service

Known mitigation: None

Assets: The server which contacts the credit card company

Entry points: Credit card information submission page

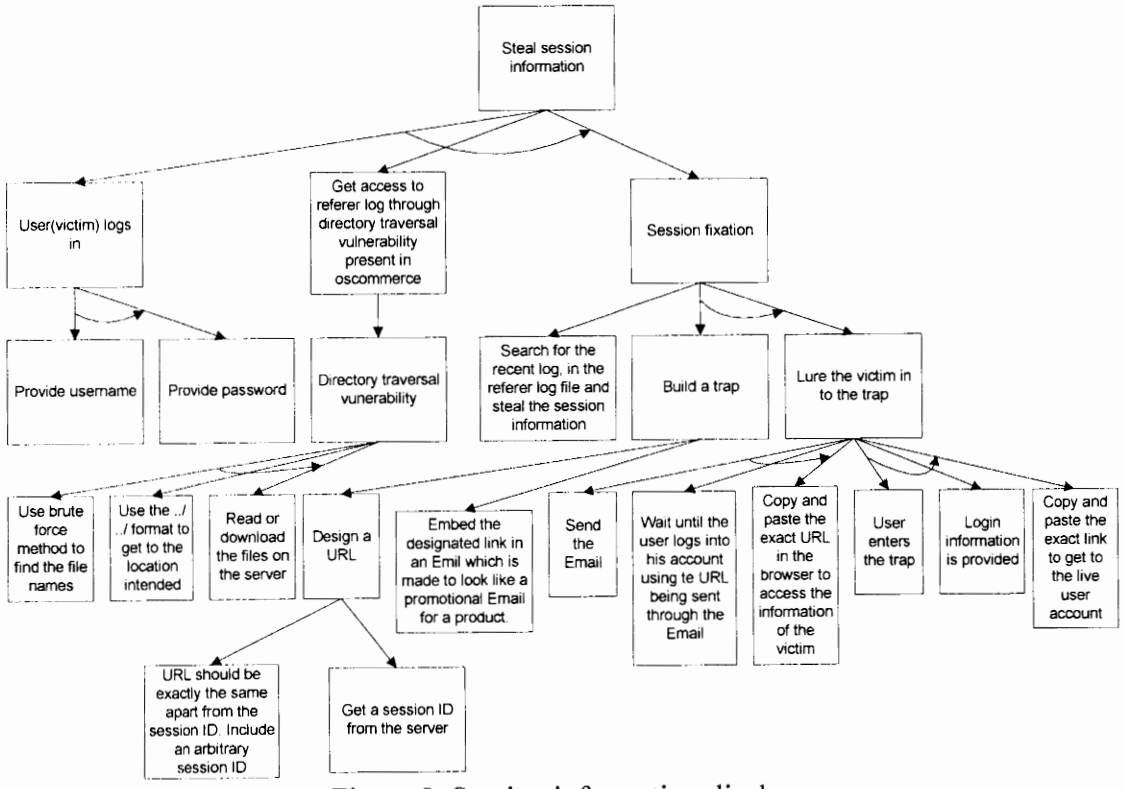

Figure 9. Session information disclosure

Description: Transaction resource attack can be achieved with the credit card authorization process. The steps to achieve that are as follows. Create a new account and login to that account, enumerate credit card numbers which are fictitious, and apply them on the account you are logged in.

Threat tree: Figure 10 shows the threat tree.

ID: 09

Name: User data disclosure

STRIDE classification: Information disclosure

Known mitigation: If an attacker has login credentials to the database, nothing can be done and there is no mitigation to that situation. When it comes to sending requests to the database, they can be validated before the requests hit the database.

Assets: Database

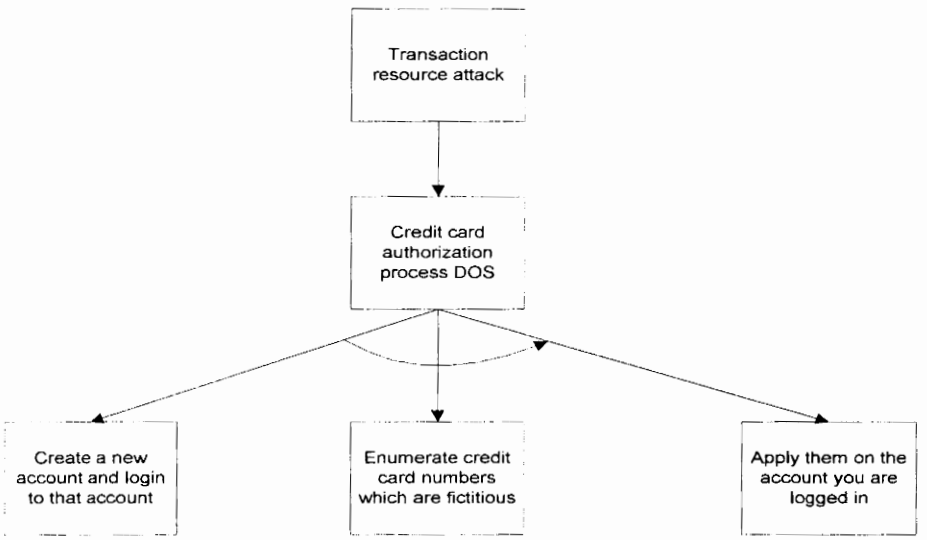

Figure 10. Transaction resource

Entry points: Database login page

Description: This attack can be done in three ways.

Hacker has direct access to the database server which means he has SQL login information and he calls the procedure to pull the user's personal data from the database.

Hacker has already a valid login to the website and also access to the data entry page so hacker injects another username in the query and pulls the user's personal data from the database.

Hacker obtains another user's credentials or gets hold of the session ID.

Threat tree: Figure 11 shows the threat tree.

ID: 10

Name: User data tampering

STRIDE classification: Tampering

Known mitigation: There is no mitigation if an attacker has the login credentials to the database. When it comes to the session information, if a user is restricted to one session for one IP address this attack can be mitigated.

Assets: Database, User information

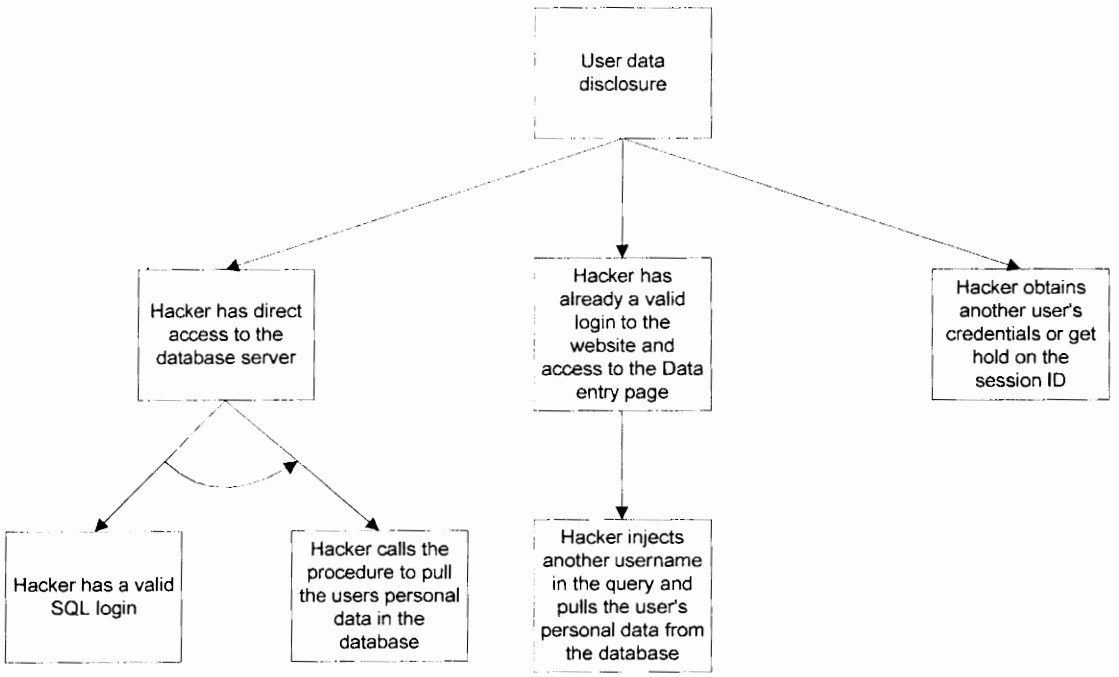

Figure 11. User data disclosure

Entry points: Database login page

Description: This attack can be done in two ways

If the hacker has direct access to the database, he manipulates information by calling the appropriate procedure to store the data.

Hacker changes the username parameter in the HTTP request to inject their information. Hacker has a valid customer account and has access to secure customer input page. Hacker obtains password from the customer by means of social engineering techniques or steals the session ID.

Threat tree: Figure 12 shows the threat tree.

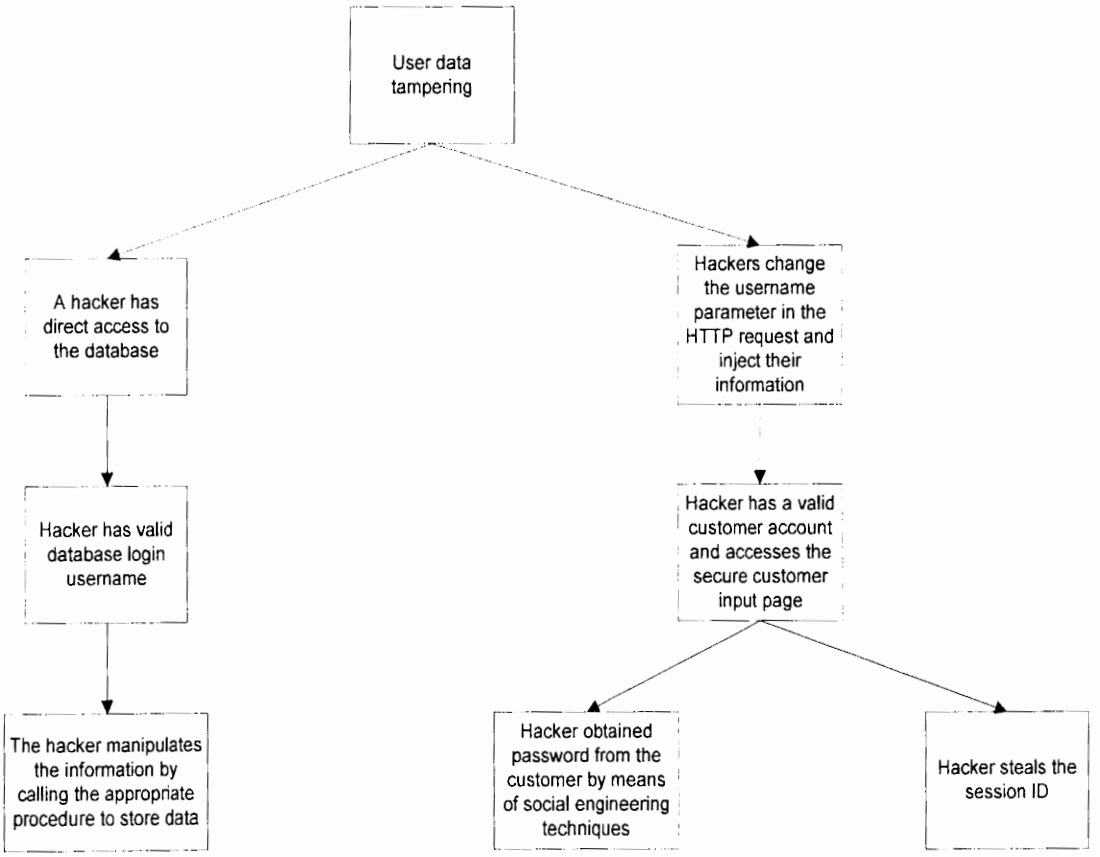

Figure 12. User data tampering

ID: 11

Name: Cross site scripting (XSS)

STRIDE classification: Spoofing

Known mitigation: A validation method needs to be added so that the data that is provided by the user does not change the information provided to the user.

Assets: Users

Entry points: Server which serves the DIV and Iframe

Description: XSS attack can be done in three ways.

1. XSS attack of hidden image: Attackers add script of hidden image in page, Users browse modified page, Users request hidden image and attacker gets the session ID.
2. XSS attack of form: Attackers modify the attribute of form in page, users browse modified page, users request the form and attackers get the session ID.
3. XSS attack of float DIV or Iframe: Attackers add float DIV or Iframe in admin login page, Users browse modified page, Users login and attacker gets the user ID and password.

Threat tree: Figure 13 shows the threat tree.

ID: 12

Name: SQL injection

STRIDE classification: Tampering

Known mitigation: None

Assets: Database

Entry points: Database login, User login

Description: Attackers browse and select product, attackers edit information of product, SQL injection of the id array, attackers add items, attackers checkout, attackers confirm order, inquire database and obtain sensitive information.

SQL injection of the id array can be done in three ways, first one is SQL injection of credit card information, second one is SQL injection of user personal information, and the third one is SQL injection of user login information.

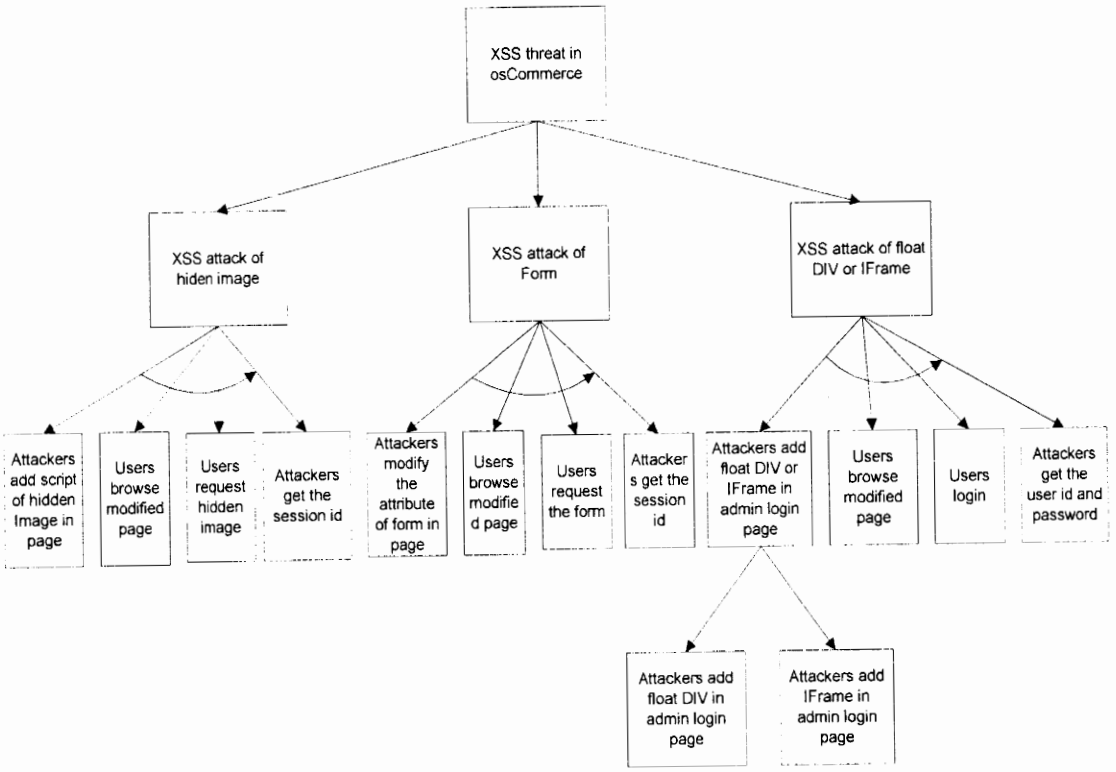

Figure 13. Cross site scripting

Threat tree: Figure 14 shows the threat tree.

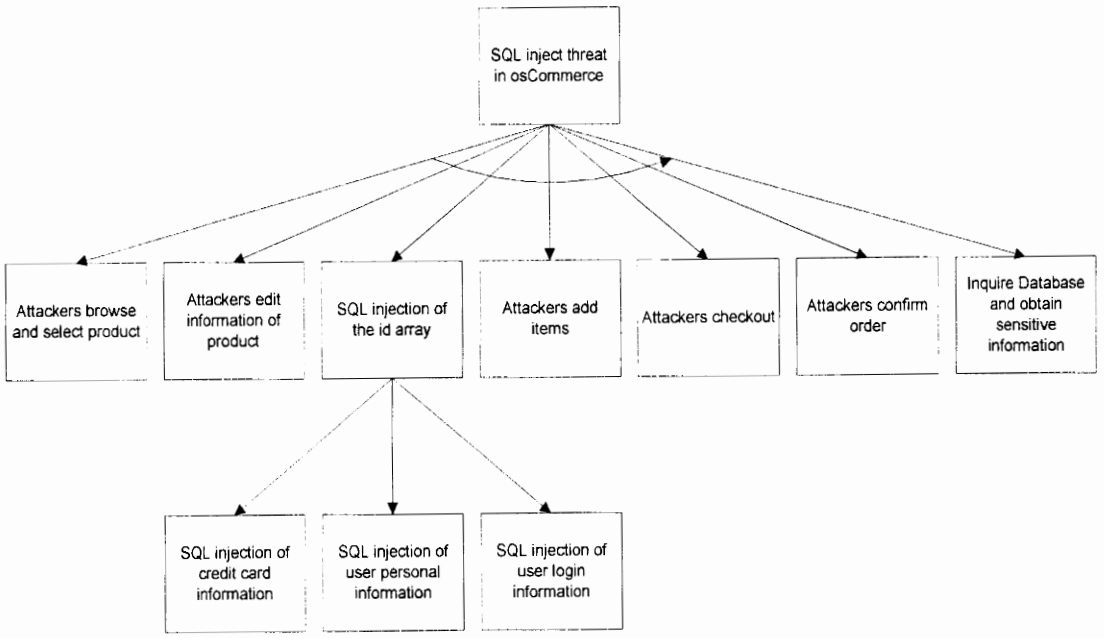

Figure 14. SQL injection

4. DRUPAL

4.1. Introduction to Drupal

Drupal is an open source content management platform powering millions of websites and applications. It is built, used, and supported by an active and diverse community of people around the world. Drupal is free, flexible, robust and constantly being improved. Drupal can be used to build everything from personal blogs to enterprise applications. Thousands of add-on modules and designs let you build any site you can imagine [6].

Drupal is maintained and developed by a community of hundreds of thousands of users and developers. It is distributed under the terms of the GNU General Public License which means anyone is free to download it, share it with others, and contribute back to the project [6].

Drupal works on any server which has Apache or Microsoft IIS, PHP, and MySQL installed and is web based which means it is compatible with any operating system [6].

Manage content with an easy-to-use web interface. Drupal's flexibility handles countless content types including video, text, blog, podcasts, and polls with robust user management, menu handling, real-time statistics and optional revision control. Drupal comes with great options for new user accounts and user permissions. Users can be assigned one or more roles, and each role can be set up with fine-grained permissions allowing users view and create only what the administrator permits. Drupal's presentation layer allows designers to create highly usable, interactive experiences that engage users and increase traffic. Following section will describe how we can model the application [6].

4.2. Function modeling of Drupal

The main artifact for analyzing Drupal is a data flow diagram (DFD). Microsoft's Threat Modeling approach is based on decomposition of an application with data flow diagrams. For this part of our experiment, we have decided to use the same approach where, the details that are provided by the DFD for Drupal is enough for the Security analysis. Dataflow diagram for Drupal is as shown below.

The dataflow diagram for Drupal is shown in Figure 15 and is a Level 1 DFD diagram and it has one external interactor, five processes, and one data store. External interactor is the user. There are five processes, Validation, Inspect forms, Create page, Create story and Logout, and one data store that is the database. Validation process has user information as input from the user, user information is validated and is sent to the database, and an acknowledgment is sent to the user. Inspect forms process has forms (username and password) as the input and improperly filled out forms as the output. Create Page process has page information (title, body, pets) as the input and an acknowledgement is sent to the user after storing it in the database. Create Story process has story information (title, body, pets) as the input and an acknowledgement is sent to the user after storing it in the database. Logout process has user information that needs to be saved as the input and an acknowledgement statement as the output.

4.3. Threat trees for Drupal

ID: 01

Name: User Credentials Disclosure

STRIDE classification: Information disclosure

Known mitigation: A validation method can be added to mitigate the SQL injection. Brute force attack can be mitigated by adding another security layer. Example for adding another security layer is by saving user's security question. A security question can be logical information specific to the user.

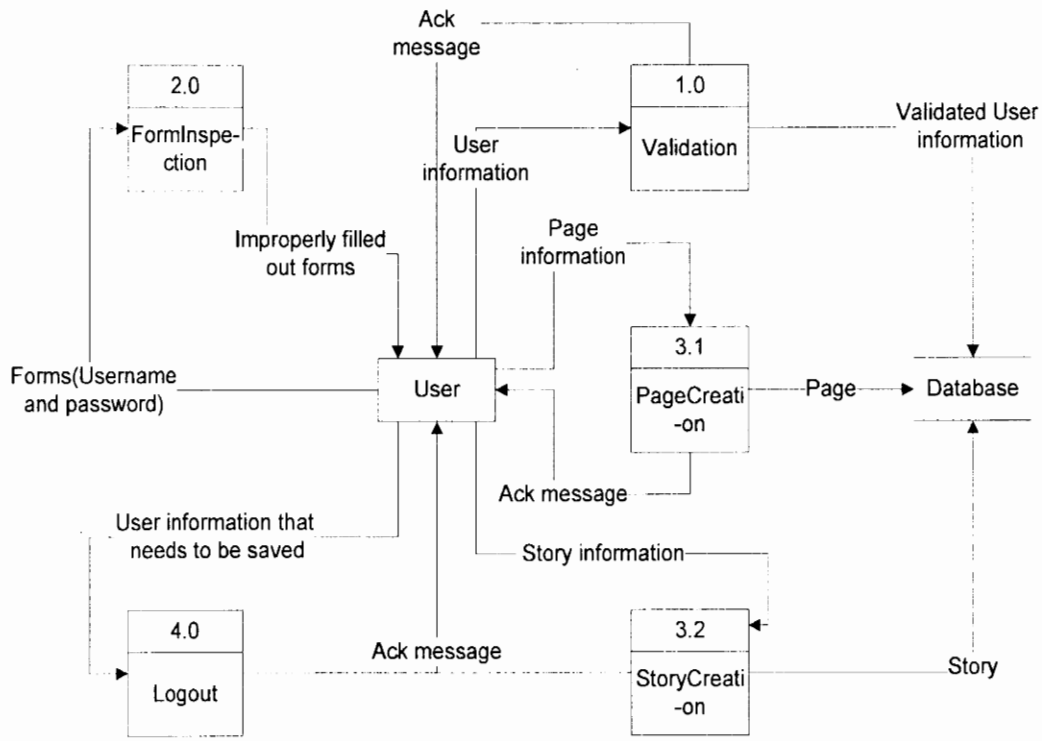

Figure 15. DFD for Drupal

Assets: Particular user's account which is under attack.

Entry points: Login page and URL construction

Description: SQL injection can be done in two ways as explained below.

Adversary gets a valid username and password. To get a valid username adversary uses error string from the login page to determine username validity or adversary gets user to disclose the username. To get a valid password adversary calls retrieve credentials

procedure or uses brute force attack on the login page to guess a user's password or guesses the password or gets the user to disclose the password. Adversary can get in using SQL injection attack. Save the source code on the disk, make changes, reload and give username and password.

Threat tree: Figure 16 shows the threat tree.

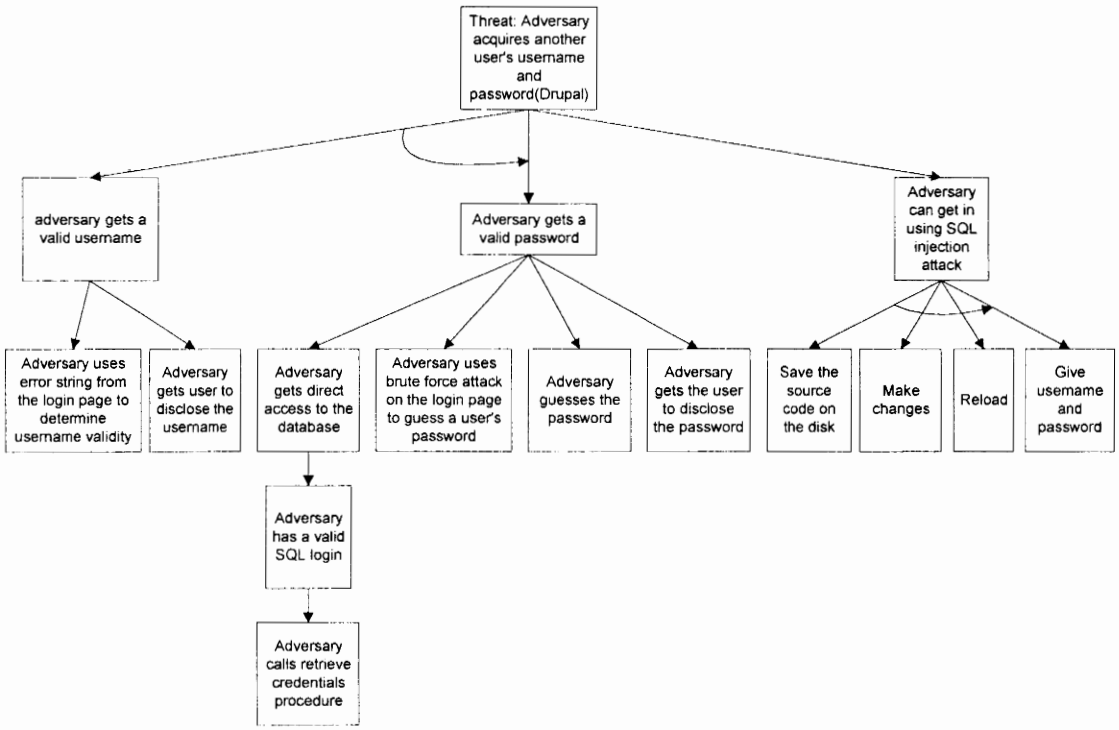

Figure 16. User credentials disclosure

ID: 02

Name: Gaining access to admin files

STRIDE classification: Tampering and Elevation of privilege

Known mitigation: An authentication layer is added to deny access to admin files.

Assets: Admin files

Entry points: URL construction which is based on action parameter

Description: Hacker gets the login credentials of an administrator using other threats like 'Login information vulnerability' and the only authentication layer for admin is the login page, that is if the attacker can crack that he can play with the admin files and see the results in the actual application.

Threat tree: Figure 17 shows the threat tree.

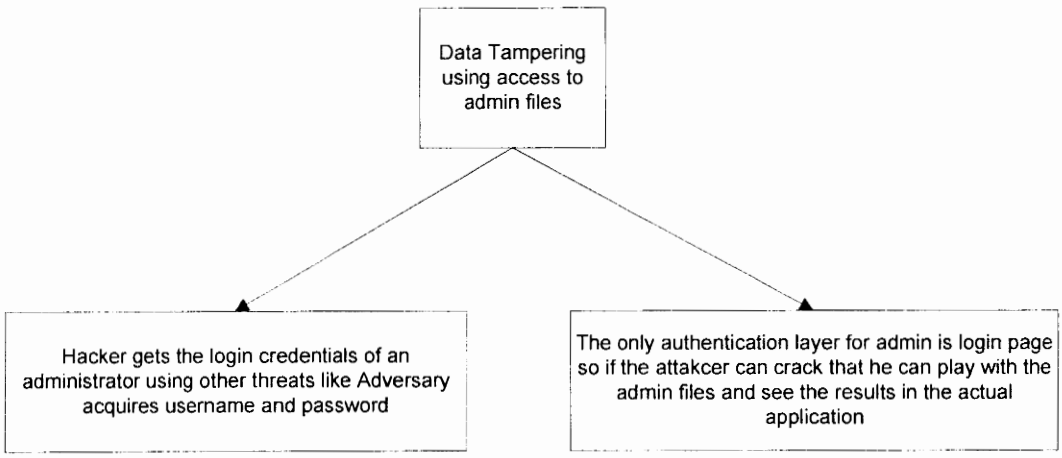

Figure 17. Gaining access to admin files

ID: 03

Name: Denial of service

STRIDE classification: Denial of service

Known mitigation: A module can be added to the application where it can prevent login process automation. One way to do it is by giving distorted text that only a person can decode and enter the correct text. This method holds good for all the automation attacks.

Assets: Server which provides services to the users

Entry points: Login page

Description: Denial of service in this application can be done in 4 ways.

Create multiple users: First step by an attacker is to automate the process for creating multiple new accounts. Second step is to apply that to Drupal.

Password recovery process attack: If there is only a single email ID available, then automate the process so that Drupal keeps recovering the password for that particular email ID. If there are multiple email addresses available, automate the process so that it uses each email ID for certain number of times and then goes to the next one and continue until the end of the list.

Login process attack: This attack should be done in two steps. The first step involves username and the second step involves the password.

Step 1: This can be done by enumerating all the fictitious email addresses.

Step 2: This can be done by applying the dictionary attack just for the password field and use the above process for the username.

Starvation attack: Starvation attack can be done by automating the procedure for creating new accounts and then apply it to Drupal.

Threat tree: Figure 18 shows the threat tree.

ID: 04

Name: User data disclosure

STRIDE classification: Information disclosure

Known mitigation: If an attacker has login credentials to the database, nothing can be done and there is no mitigation to that situation. When it comes to sending requests to the database, they can be validated before the requests hit the database.

Assets: Database

Entry points: Database login page

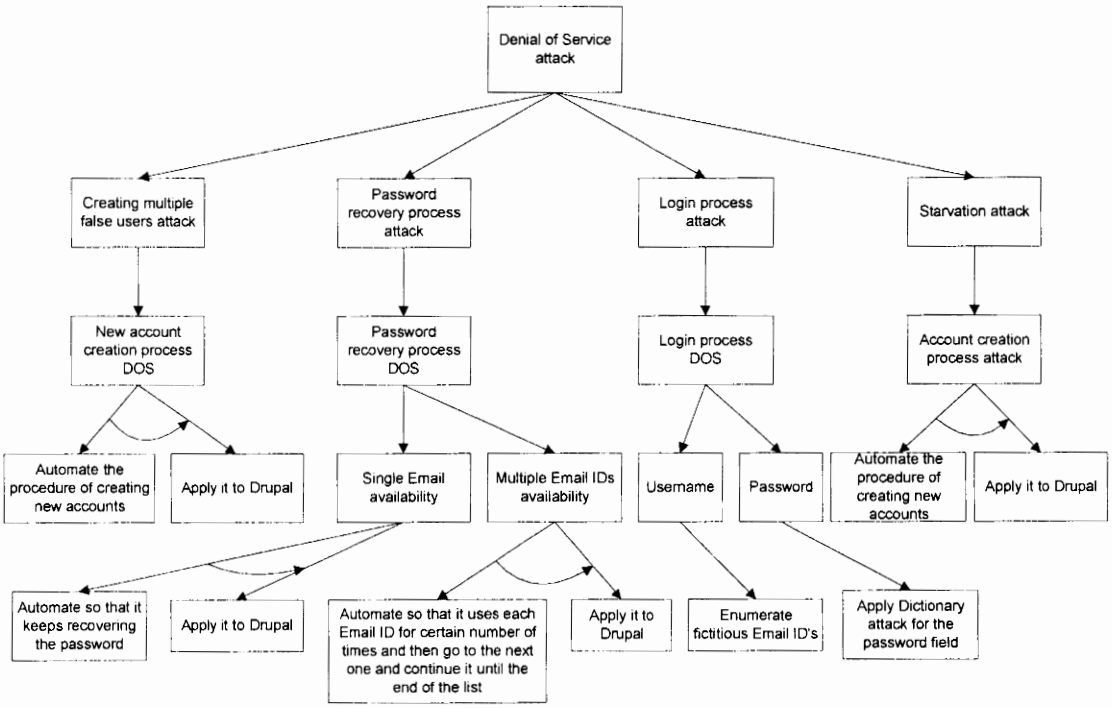

Figure 18. Denial of service

Description: This attack can be done in three ways.

Hacker has direct access to the database server. Hacker has valid SQL login information and he calls the procedure to pull the user's personal data from the database. Hacker already has a valid login to the website and access to the data entry page so hacker injects another username in the query and pulls the user's personal data from the database.

Hacker obtains another user's credentials.

Threat tree: Figure 19 shows the threat tree.

ID: 05

Name: User data tampering

STRIDE classification: Tampering

Known mitigation: There is no mitigation if an attacker has the login credentials to the database.

Assets: Database, User information

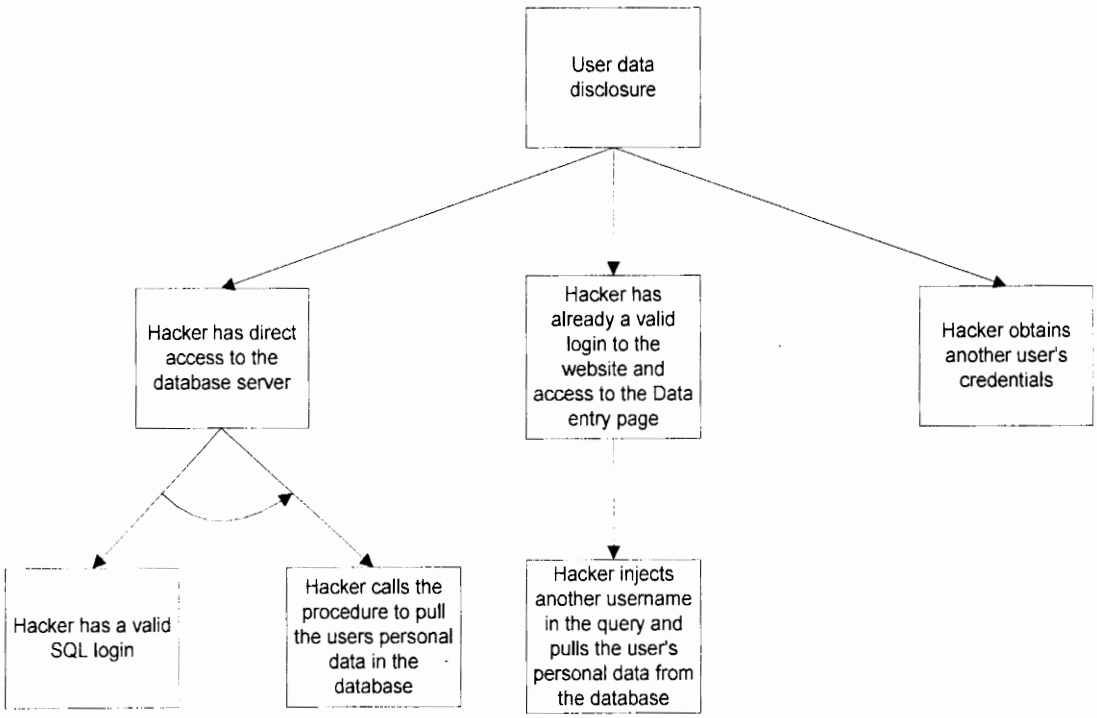

Figure 19. User data disclosure

Entry points: Database login

Description: This attack can be done in two ways

1. If the hacker has direct access to the database, he manipulates information by calling the appropriate procedure to store the data.
2. Hacker changes the username parameter in the HTTP request and injects their information. Hacker has a valid customer account and has access to the secure customer input page. Hacker obtains password from the customer by means of social engineering techniques.

Threat tree: Figure 20 shows the threat tree.

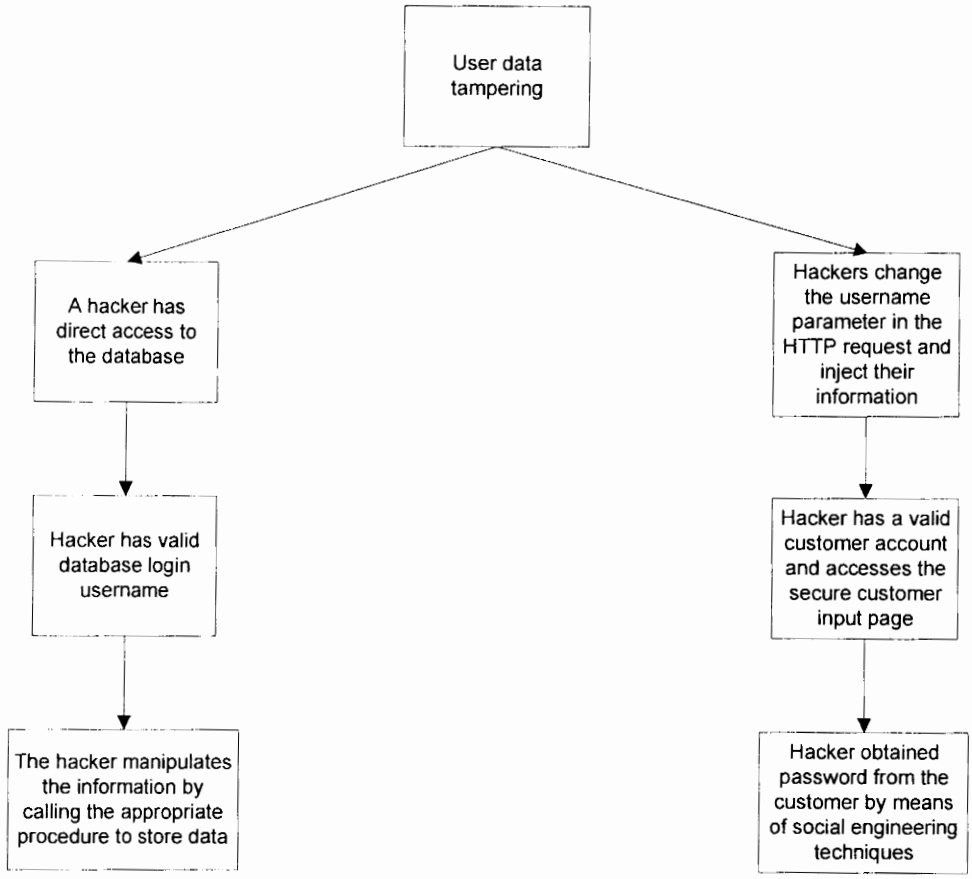

Figure 20. User data tampering

5. CONCLUSION AND FUTURE WORK

5.1. Results

A case study of Analysis of security threat using Threat Modeling has been presented in this paper. Two real world applications, osCommerce and Drupal were used for the case study. Twelve threat trees for osCommerce and five threat trees for Drupal were developed. This case study proves that the approach is powerful in exposing the security vulnerabilities that have been introduced in to the software system and entirely concentrates on how an attacker sees the web applications.

Threat trees developed in this paper are used for the empirical study conducted for writing the papers 'Security Test Generation using Threat trees' and 'A model-based approach to Security Testing'.

The results from our study show that Security test generation using Threat trees can be an effective test case development technique for discovering vulnerabilities. The lack of false positives and negatives also shows that this can be an effective security testing and verification technique. Threat trees based approach identifies the threat goals that attackers want to realize. These threat goals are the targets of the security testing. Different sequences of attack actions can be derived from the threat trees. These sequences represent the ways in which attackers realize the threat goals and thus provide a basis for security test sequences.

The results also show that some threat trees that exposed vulnerability in one application could be used to expose vulnerability in another application after being slightly modified. This could possibly be attributed to a few factors: (1) Both the programs are written in PHP, use a MySQL database for data storage, and are deployed

on the Apache web server; (2) both have some similar functionality, especially when it comes to user management authentication; (3) the programs are both web applications that have an identical high level architectural design. The threats that were not applicable for both applications were typically due to the areas where the programs differ in functionality or implementation. The denial of service threat is a vulnerability that is present in both applications. This is likely due to the similar architectures. The user data tampering threat is an example where osCommerce was vulnerable and Drupal was not. This is likely due to implementation differences in the safeguards of the applications use when accessing the database.

5.2. Limitations

Although the results for the case study are promising, there are some limitations for analysis of the software products that were used in the study. The major limitation is how the threat trees were developed with lack of comprehensive design documentation for osCommerce and Drupal. This limitation forced us to create threat trees from known vulnerabilities and general vulnerabilities instead of design documents. Another limitation is the lack of variety in object programs. An attempt has been made to minimize this threat by performing realistic attacks against a large, readily deployed web application.

5.3. Future work

For future work, these approaches can be applied to different kinds of applications and consider more types of security threats. In this paper, threats were identified

according to the STRIDE category. This research can be extended to model more threat trees by considering various ways of violating security goals, such as authentication, authorization, confidentiality, integrity, and availability.

This research can be extended to model threat trees for different kinds of applications to standardize some threats to certain kinds of applications so that there will be some kind of standard to mitigate those threats. Those mitigation processes can be included in the development process so that a secure system can be built.

REFERENCES

[1] I. Alexander, "Misuse cases: use cases with hostile intent," Software, IEEE, vol. 20, no. 1, pp. 58-66, 2003.

[2] R. B. Ammar Masood, Arif Ghafoor, Aditya Mathur, Model-based Testing of Access Control Systems that Employ RBAC Policies, Purdue University, Sept 2005.

[3] A. G. Ammar Masood, Aditya Mathur, Scalable and Effective Test Generation for Access Control Systems that Employ RBAC Policies, Purdue University, Sept 2006.

[4] B. Beizer, Software Testing Techniques: Van Nostrand Reinhold Co. New York, NY, USA, 1990.

[5] K. E. N. DianxiangXu, "Threat-Driven Modeling and Verification of Secure Software Using Aspect-Oriented Petri Nets," IEEE transactions on software engineering, vol. 32, pp. 265-278, April 2006.

[6] D. webpage, "http://www.drupal.org/."

[7] E. G.Amoroso, Fundementals of Computer Security Technology: Prentice Hall inc, 1994.

[8] T. X. Evan Martin, Ting Yu, "Defining and Measuring Policy Coverage in Testing Access Control Policies," in Proc. 8th International Conference on Information and Communications Security, 2006.

[9] T. X. Evan Martin, "Automated Test Generation for Access Control Policies via Change-Impact Analysis," in Proceedings of the Third International Workshop on Software Engineering for Secure Systems, 2007.

[10] T. X. Evan Martin, "A Fault Model and Mutation Testing of Access Control Policies," in Proceedings of the 16th international conference on World Wide Web, 2007.

[11] F. S. a. W. Snyde, "Threat Modeling," 2004.

[13] H. H. T. a. J. A. Whittaker, "Testing for Software Security," Dr. Dobb's Journal, 2002.

[14] J. McDermott, "Abuse-Case-Based Assurance Arguments," in Proceedings of the 17th Annual Computer Security Applications Conference, 2001.

[15] E. W. Linzhang Wang, Dianxiang Xu, "A Threat Model Driven Approach for Security Testing," in Proceedings of the Third International Workshop on Software Engineering for Secure Systems, 2007.

[16] R. Majumdar, "Paul Ammann and Jeff OffuttIntroduction to Software Testing. Cambridge University Press (2008). ISBN: 978-0-521-88038-1. £32.99. 322 pp. Hardcover," The Computer Journal, vol. 53, no. 5, pp. 615, June 1, 2010, 2010.

[17] M. H. D. L. J. Viega, 19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them: McGrawHill, 2005.

[18] M. M. Monica S. Lam, Benjamin Livshits, John Whaley, "Securing web applications with static and dynamic information flow tracking," in Proceedings of the 2008 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation, San Francisco, California, USA, 2008.

[19] C. K. Nenad Jovanovic, Engin Kirda,, "Precise alias analysis for static detection of web application vulnerabilities," in Proceedings of the 2006 workshop on Programming languages and analysis for security, Ottawa, Ontario, Canada, 2006.

[20] O. C. WEBSITE. "http://www.oscommerce.com/."

[22] B. M. Potter, G., and A. Booz, "Software Security Testing," Security & Privacy, IEEE vol. 2 no. 5, pp. 81-85, Sept.-Oct. 2004.

[23] G. Sindre, and A. L. Opdahl, "Eliciting Security Requirements by Misuse Cases," in Technology of Object-Oriented Languages and Systems, 2000.TOOLS-Pacific 2000.Proceedings. 37th International Conference on Sydney, NSW, Australia, 2000.

[24] W. s. A. c. t. D. Center, Only 10% of web applications are secured against common hacking techniques, Foster City California, 2004.

[25] A. O. William G.J. Halfond "Improving Test Case Generation for Web Applications Using Automated Interface Discovery."

[26] X. Dianxiang, "Software security," Wiley Encyclopedia of Computer Science and Engineering, B. Wah, ed., John Wiley & Sons, Inc, 2008.

[27] F. Y. Yao-Wen Huang, Christian Hang, Chung-Hung Tsai, Der-Tsai Lee, Sy-Yen Kuo, "Securing Web Application Code by Static Analysis and Runtime Protection," in Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, 2004.

[28] Y. Minamide, "Static Approximation of Dynamically Generated Web Pages," in Proceedings of the 14th international conference on World Wide Web, 2005.

[29] "http://www.getahead-direct.com/."

[30] C. C. I. A. L. Tom Stracener, Application Security Trends ReportQ1 2008, CIA (Cenzic Intelligent Analysis) Lab, 2008.