Honours Project Report

A Comparative Analysis of Threat Modeling HACKMI2 web application

Using the SensePost CTM tool

A mini-dissertation submitted to the Department of Computer Science of the University of Cape Town in partial fulfilment of the requirements for the award of an Honours Degree in Computer Science.

By

Sanele Macanda (MCNSAN002)

Under the supervision of Dr. Anne V.D.M Kayem, Senior lecturer of Information Security, University of Cape Town

October 2012

		Min	Max	Chosen
1	Requirement Analysis and Design	0	20	13
2	Theoretical Analysis	0	25	12
3	Experiment Design and Execution	0	20	5
4	System Development and Implementation	0	15	10
5	Results, Findings and Conclusion	10	20	5
6	Aim Formulation and Background Work	10	15	5
7	Quality of report Writing and Presentation	10		10
8	Adherence to Project Proposal and Quality of Deliverables	10		10
9	Overall General Project Evaluation	0	10	10
Total marks		**80**		**80**

Acknowledgements

My profound gratitude goes to God the almighty, my family and friends, and all the individuals who helped me during the preparation and write up of this dissertation.

Special thanks to Dr. Anne Kayem for taking the time to supervise me and for having the belief that this could be accomplished and I am grateful for the insightful comments receive for the draft of this thesis report.

Special thanks to the National Research Funding (NRF) for funding me this year, without them I wouldn't have pursued my studies.

I want to thank my group members Rotodwa Ratshidaho and Molulaqhooa Maoyi for the co-operation during the project. For putting many hours on developing the social network and the whole project in general.

Contents

Appendix A: Table of figures

Abstract

The popularity of web applications has increased rapidly in recent few years [1] and consequently, securing in web applications such as social networks has become a strong requirement for these applications. Identifying security vulnerabilities in applications is challenging and when security patches are implemented, can lead to other more serious vulnerabilities [2]. Therefore, there is a growing consensus to analyze and model the potential security threats in a system/application before implementation [3]. This project report presents a method of modeling security threats in an experimental social network web application namely, HACKMI2. More specifically, we use a threat modeling tool namely, SensePost Corporate Threat Modeler (CTM), to highlight the security vulnerabilities that emerge in building the "HACKMI2" social networking application. We discuss the vulnerabilities that are typically found in social networks and propose code solutions to mitigate these vulnerabilities. A comparative analysis of the SensePost CTM threat modeling tool in relation to the Microsoft SDL Threat Modeling tool and Microsoft TAM, indicates that the CTM ranks threats according to its impact on the system and it is very subjective when modeling the application from a software-centric approach.

Chapter 1. Introduction

The growing number of internet users attests to the fact that the world is becoming increasingly technology driven [1]. Social networks are quite popular and they encourage data sharing and seamless user interactions facilitate the use of internet. This study is centered on social networks because they are an example of a web application that involves multiple user interaction and raises a lot of issues that are centered on securing access to shared data. This makes them one of the prime targets to attack because these web applications typically handle considerable amount of personal information. The HACKMI2 social network can be exposed to different types of attacks which make the information on which the application manipulates vulnerable. Threat modeling is a method of identifying threat and providing mitigations to threats. We will use the SensePost CTM tool to identify threats on the HACKMI2 web application. The web application will be viewed from a software-centric approach where it evaluates weaknesses in security controls of the application. The objectives of this study is model the HACKMI2 web application and identifying threats on it, provide code mitigation to stop these threats from occurring and compare the different threats models on how the they model and identify threats on the web application

1.1 Research questions

- The main research questions for this study are finding out what are the social networking security risks? Identifying threats that could cause a severe impact on the application and what type of attacks can be perpetrated by a malicious code. Once the vulnerabilities have been identified we look at what countermeasures could be employed by the software-centric approach to identify threats and mitigate them. Finally, we look at How do the attack centric and software centric models differ with respect to identifying threats and vulnerabilities.

The plan of the study is as follows. Chapter 2 looks at the key components of the Elgg framework and how they work. Chapter 3 shows how the HACKMI2 web application was built using the Elgg framework and key features of this web application. Chapter 4 outlines the method of modeling threats namely, Threat Modeling and how it can be used to model threats on web applications. It also provides models such as the STRIDE and DREAD models which categorizes and rates threats respectively. The chapter concludes by outlining the importance of threat modeling web application. In chapter 5, I used the SensePost threat modeling tool to model the HACKMI2 web application. How I mitigate the threats discovered is dealt with in chapter 6. Chapter 7 provides a comparative analysis of different threat models on which advantages and disadvantages of each model are discussed. Chapter 8 concludes the report by stating a brief summary of the whole report and providing work that can be done on the future.

Chapter 2. Elgg

This section presents an overview of the Elgg framework. We explain what Elgg framework is, and describe its components in order to show how the components were used to construct our experimental social network ("HACKMI2").

2.1 The Elgg framework

Elgg is an open source social networking framework that is built on the Linux, Apache, MySQL and PHP (LAMP) system; it offers the basic functionality that enables a user to create a customized version of a social network. This framework is also the engine that makes it possible to run the social network externally, from where it can be accessed from all over the world, or internally, in the case where it is on a local network [4]. The Elgg framework is quite generic and comprises many other social networking components as social intranet for organizations, which can also be modified to behave in a customized fashion. We also note that the Elgg framework is very extensible making it is easy to modify or change some of its functionality through plugins.

2.2 Elgg Overview

Amongst the Elgg framework's important features, we have chosen to focus on the data model, events, views, and plugins because they were the ones we found to be most useful in creating our social network (HACKMI2). When these features interact together they create a better and realistic social networking environment. Below we describe how each feature works.

i. **Data Model:** This Elgg version 1.8v supports the latest version of MySQL 5.5. Elgg uses a flexible data model to support custom functionality. If there is new database

functionality then there won't be need to change or update the plugins.[1] [5]. Figure 1 below shows an overview of the data model.

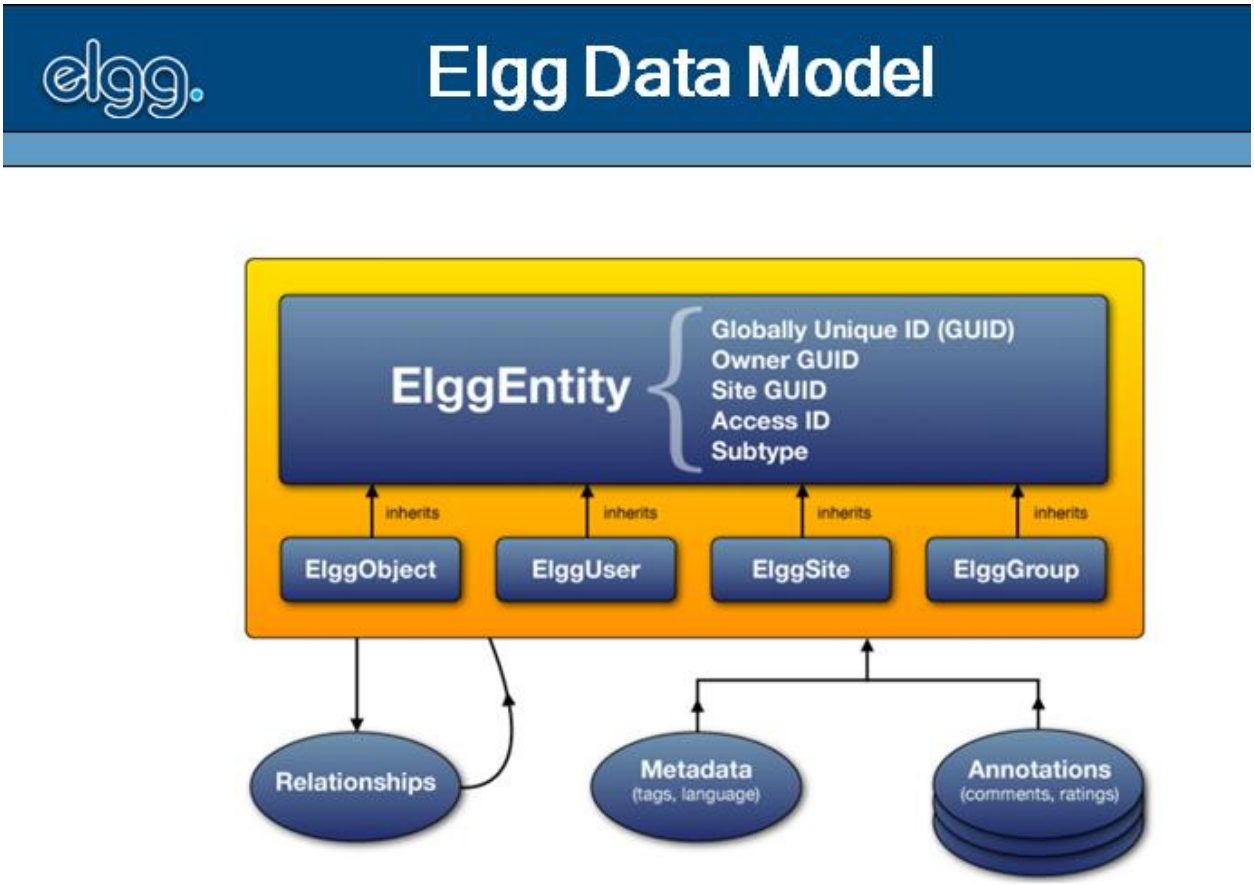

Figure 1. Data Model

ii. **Events:** Elgg events occur when a user action has occurred; these actions include login or posting a status update on the wire, these actions will trigger the add login and add comment actions which will then authenticate the user and add a comment on the application [4]. Actions are generally triggered when a user submits a form.

iii. **Views**: Another feature of the Elgg framework is that it caters for different people with diverse needs. A site built on the Elgg framework can support many different interfaces such as the normal html standard which can be optimized for mobile devices and also offers accessibility features for the blind and visually impaired users.

[1] Plugins will be discussed in detail later in this chapter.

iv. **Plugin:** Plugins are the most important feature of Elgg. Through plugins, one can replace or extend the core Elgg functionality. Plugins interact to the systems through Events and Views. They can be used to create new themes or add new functionality such as a chat system. By adding plugins one can change the entire Elgg framework. These plugins exist under the 'mod' directory of the framework files.

When creating a plugin directory, a number of sub-directories are necessary. An actions directory will trigger the event you want to do prior to the plugin being activated. A language directory which is for internationalism allows the use of different languages. A graphics directory is dedicated to images that you have used for your plugin and a views directory allows your Elgg plugin to function properly. Below is an example of how one can create their own blog plugin.

a. Creating a blog plugin

When writing an Elgg plugin one needs to use PHP programing language as this framework is developed in this programing language. One needs to create an object view, which affects how users will view see your blog. One also need to create a form which allows users to interact on the blog i.e. comment, add title of the blog, save and delete comments. The developer must create a save file so that users can save their blogs and create a display page which allows users to display

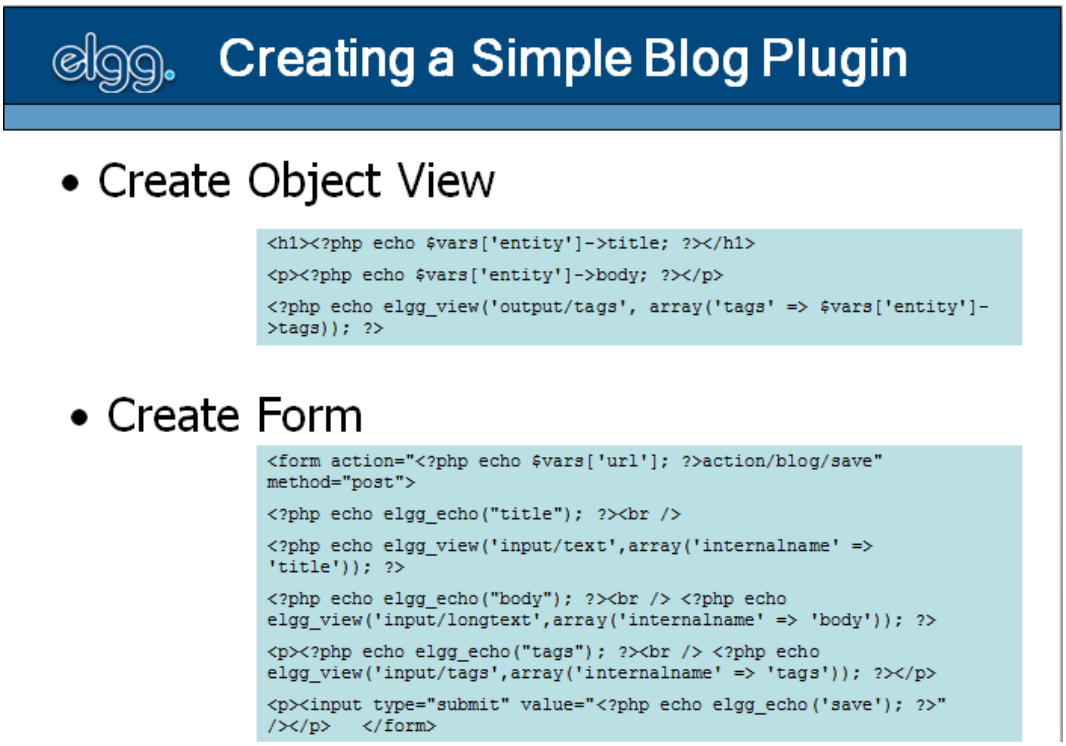

Figure 2. Creating a blog plugin

their blog posts. Figure 3 and 4 below shows how you create a blog plugin in Elgg.

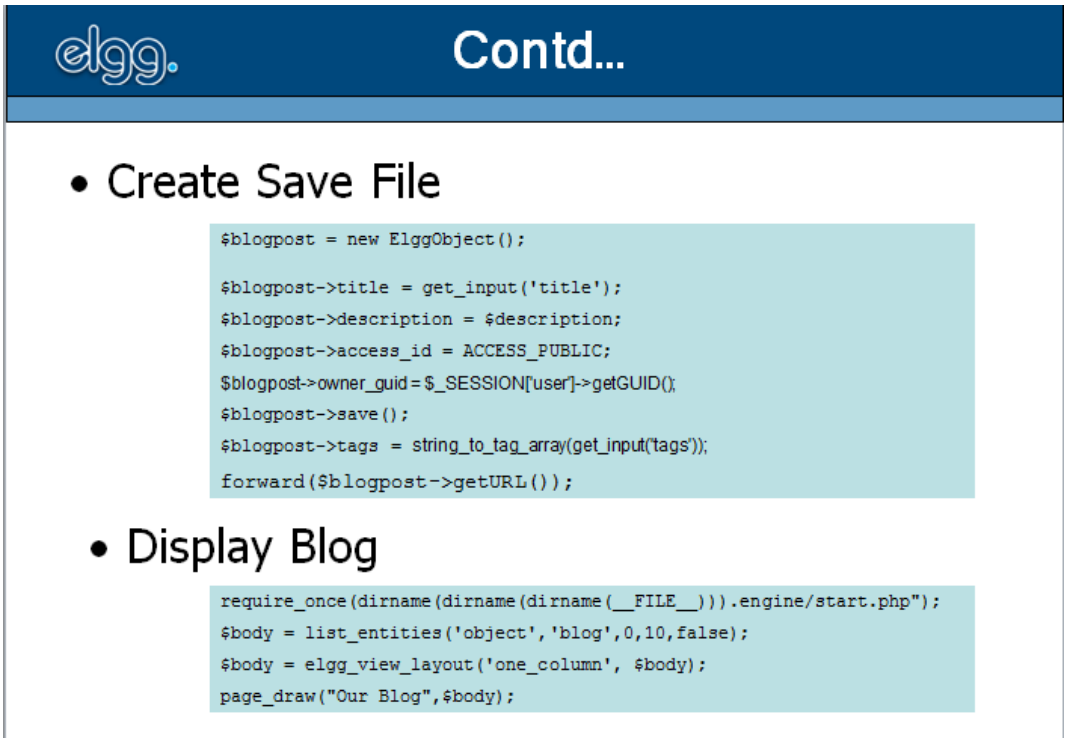

Figure 3. Creating a blog plugin continued

v. **Notifications:** The Elgg framework contains a notification directory. A notification is triggered when an event has occurred, and it serves to notify the user(s) that an event has occurred. For example, when a user has sent a friend request to another user, the user will receive a friend request notification. It also notifies users after they have sent a message that their message sent was sent successfully or not.

2.3 Components of Elgg

Elgg can be used for three different purposes; it can be used for education (like an E-learning for a University and school application), as a social intranet for organizations, and also to create a custom social network [6]. As previously mentioned, this study focuses on the application of Elgg on custom social networks. Elgg provides organizations with components needed to create a social networking. These components include messaging, blog, comments, widgets, forum,

friends/groups, files, files, RSS, pages and tags which can be used for social networking purpose. These components work together to form an efficient social networking platform. Below is a description of the what each component

- **Messaging:** This component allows friends on the social network to exchange text messages.
- **Blog:** It allows users to create, delete, and edit their blogs and other users such as friends can also comment on the blog.
- **Comment:** It allows users to comment on each other's blogs and status updates.
- **Widgets:** Shows advertisement on the social network
- **Forum:** Allows users to post topics and other users can comment on the topic, one after the other.
- **Friends/groups**: it allows users to send friend request to other users and it also allows a group of friends to form a group on which they can send each other text messages.
- **Files:** This component allows users to upload or download files that are shared on the social network. Users can also upload their avatar and can change it.
- **RSS:** This component allows RSS feeds to the social network, where it will provide information on various things from the public i.e. news.
- **Pages**: This directory contains different pages, such as an account page, a friends' page where one can see all one's friends, a profile page for when a user view his/her profile, and a settings page so that users can change their settings such as username or password.
- **Tags**: Allows users to tag each other on pictures

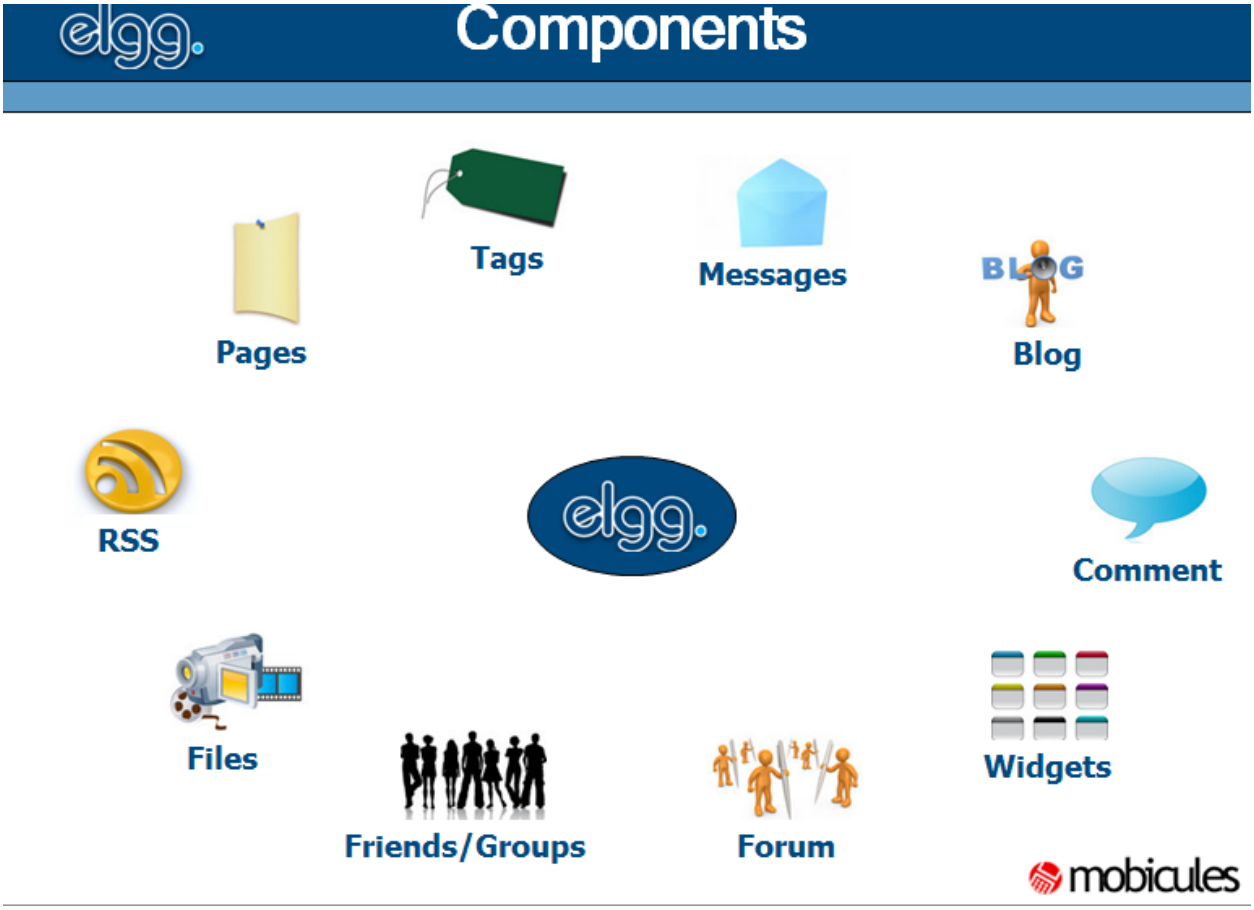

Figure 4. Elgg Components

Chapter 3. Hackm2

3.1 Introduction to HACKMI2 web application

HACKMI2 is a social network that has been built from an open source social network Elgg framework. The name "HACKMI2" comes from the fact that security attacks such as hacking will be used on the social network to try and explore some of the vulnerabilities that will be found on it. HACKMI2 serves as a case study social network that we use to study the attack types that will be highlighted by the threat modeling tool. We opted to build an experimental social network because testing on renowned social networks like Twitter and Facebook is not

legal. In this chapter the study presents the architecture of HACKMI2 and some of it general features.

3.2 Architecture of HACKMI2

HACKMI2 uses third tier architecture which consists of the client, a server and a database. This architecture structure is typical on web based applications. The web application server provides the application logic layer in the three-tier architecture and it enables client components to interact with data resources and legacy applications [7]. Collectively, the three-tier architectures are programming models that enable the distribution of application functionality across three independent systems. Each tier of the architecture is described below.

First tier: Client

The client interacts with the web browsers to allow end-users to view or access content on the on HACKMI2. This allows the client to interact with the second tier which is the server in a secure manner [7]. When users log in to HACKMI2 they need to authenticate themselves to the server by providing their username and password. The server will then communicate with the database to check if the username and password are correct or not so that they can give access to the client, as you can see the client does not have a direct access to the database.

Second tier: Server

The second tier allows access to the third tier and it is where most of the processing occurs as it must allow multiple clients to login to the system at the same time [7]. The second tier is also called the application logic layer.

The HACKMI2 server has 3TB hard disk space as many people are expected to join and use the social network. It has an i7 processor which can handle multiple users logging in to the system simultaneously. These high technical specifications enable instant communication between the server and the database.

Third tier: Database

The database is protected from direct access from clients and can only be accessed through the server. These tiers communicate with each other in order to provide efficient web application architecture. Figure 5 below shows the architecture of HACKMI2.

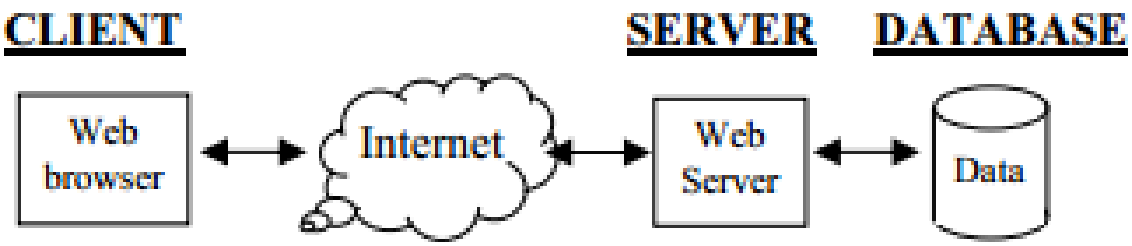

Figure 5. HACKMI2 architecture

3.3 General features of HACKMI2

Login: Registered users are required to login using their username and password. The username can be their email address or a username they have chosen. If users forget their password they are required to click on the "forget password" link so that they can retrieve them from their email address. Figure 6 below shows HACKMI2 login screen.

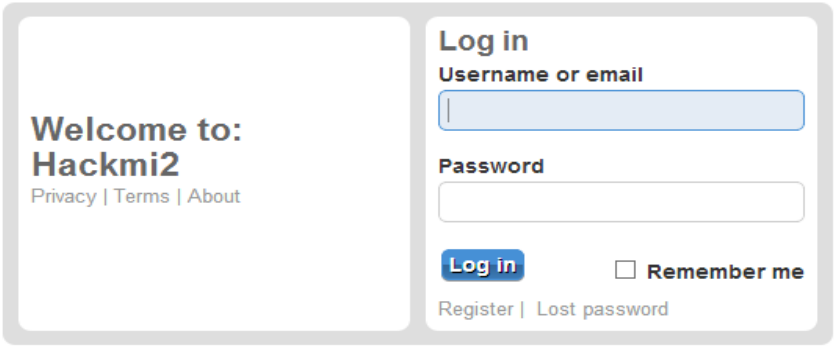

Figure 6. HACKMI2 Login Screen

Profile: HACKMI2 allows users to register first, using a username and password which will then create their profile. A user can upload a picture of them as their profile picture. They can edit their profile picture to change their username or password. Figure 7 below shows a profile on HACKMI2

Figure 7. Creating a Profile on HACKMI2

Chat: HACKMI2 allows users to chat whilst their online, users can send each other messages to multiple users at the same time through the group feature. When a user is online it will show a green colour and when they are offline then shows red.

Friend: Becoming friends on HACKMI2 is a matter of one sending a friend request, the receiving party must accept that request in order two users to become friends on HACKMI2. Once users are friends they can start sending each other text messages. Figure 8 below shows the a user's message inbox on HACKMI2

Messaging: HACKMI2 allows users to send each other inbox messages. Users can send each other message even if the other user is offline. Users can also delete messages from their inbox without affecting the counterparty's inbox. Figure 8 below shows a message inbox in HACKMI2

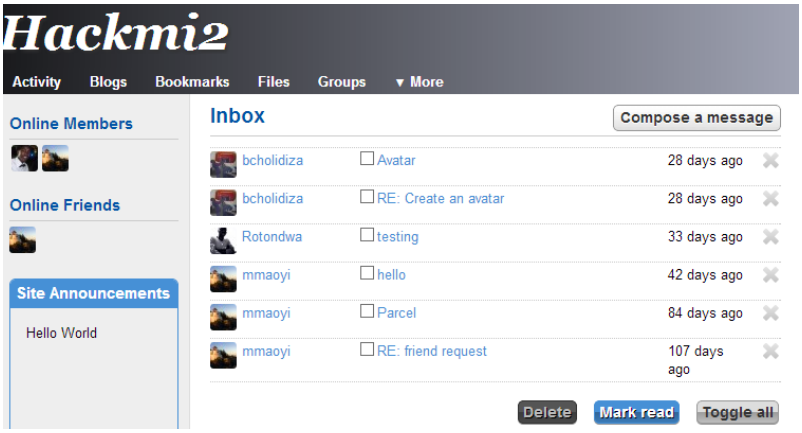

Figure 8. Message Inbox on HACKMI2

Comment: Allows friends on HACKMI2 to post comments on each other's profile picture, blogs and status updates. Figure 9 below shows how to post a comment on HACKMI2.

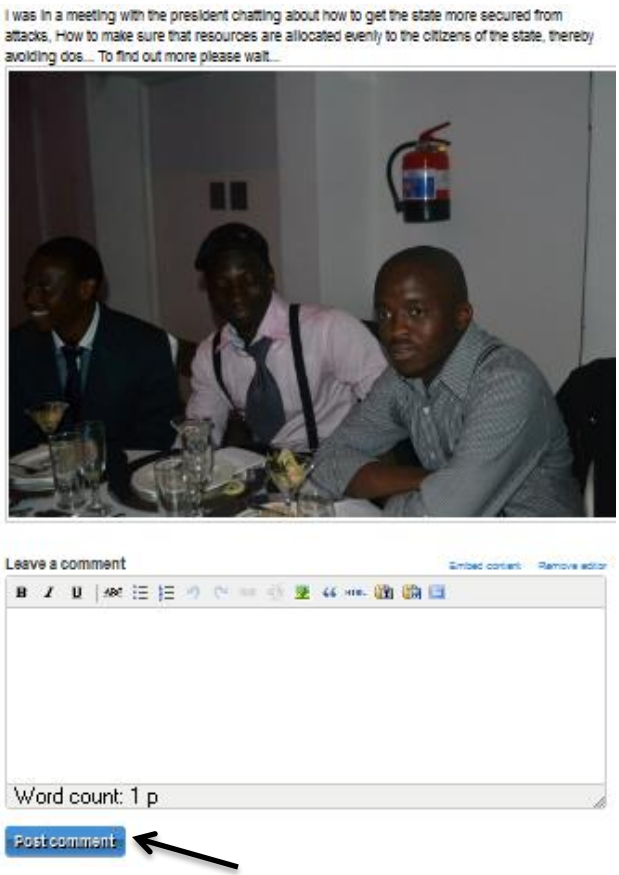

Figure 9. Posting a comment on Hackim2

Notification: HACKMI2 notifies users every time events like friend request, message or incorrect login details occur. When one user comments on another's picture, a notification will occurs to the receiving user. The notification will appear until a user has acknowledged it (clicked on that event). Figure 10 below shows a message notification on HACKMI2.

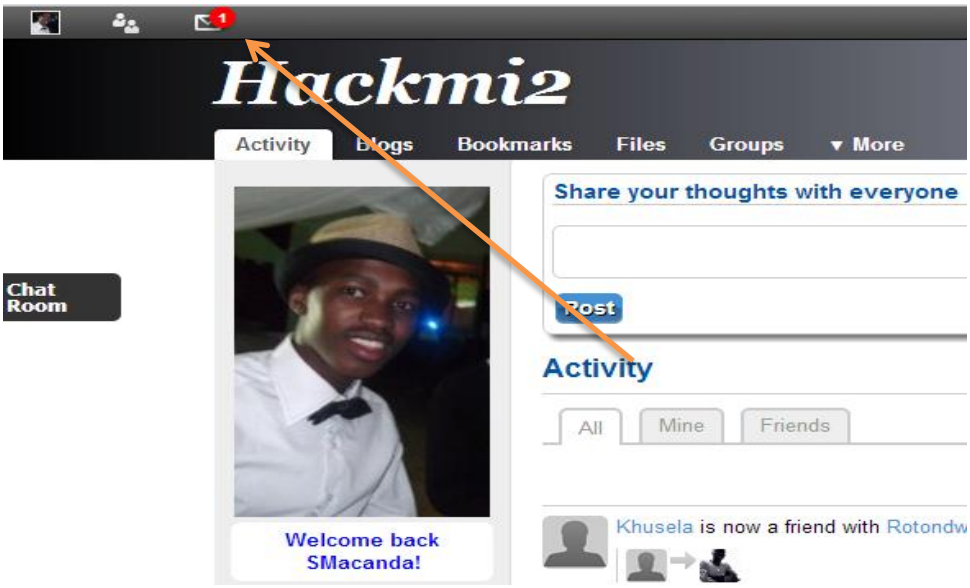

Figure 10. Message Notification

The Wire: Also known as "status updates" allows users to post messages for their friends to read. Friends can comment on the status update or click on the 'like' button. The most recent updates will be on top of the user's profile page. Figure 11 below shows where you post your status update on the wire

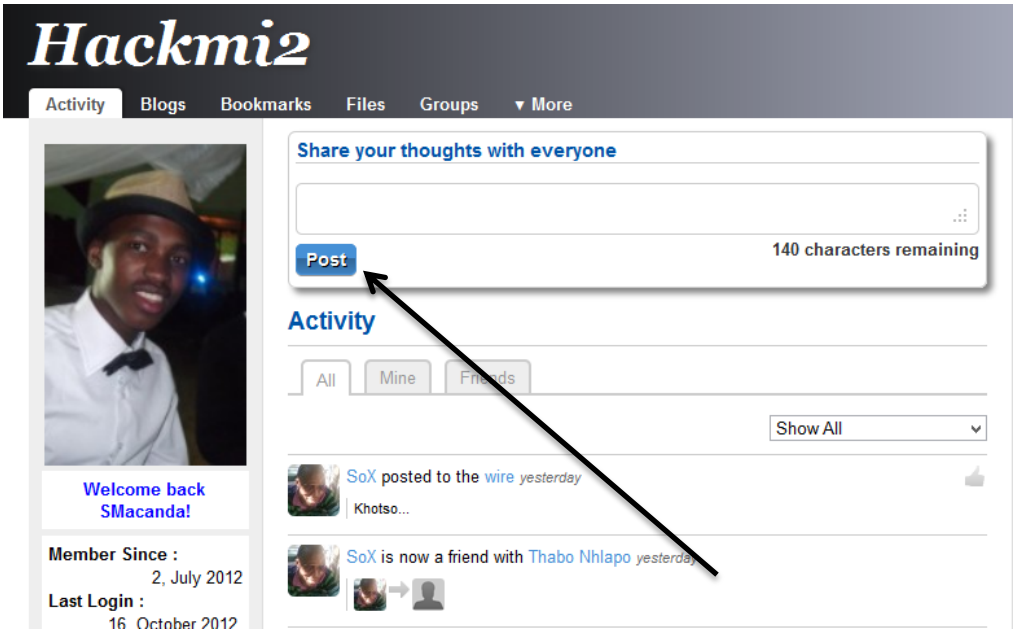

Figure 11. Status update on the Wire

Like: The like button is made for positive feedback on something. Users can like each other's status update, pictures and comments.

Files: HACKMI2 allows users to upload and download files, these files include, pictures, songs or any other document. Figure 12 below shows how to upload a file on HACKMI2.

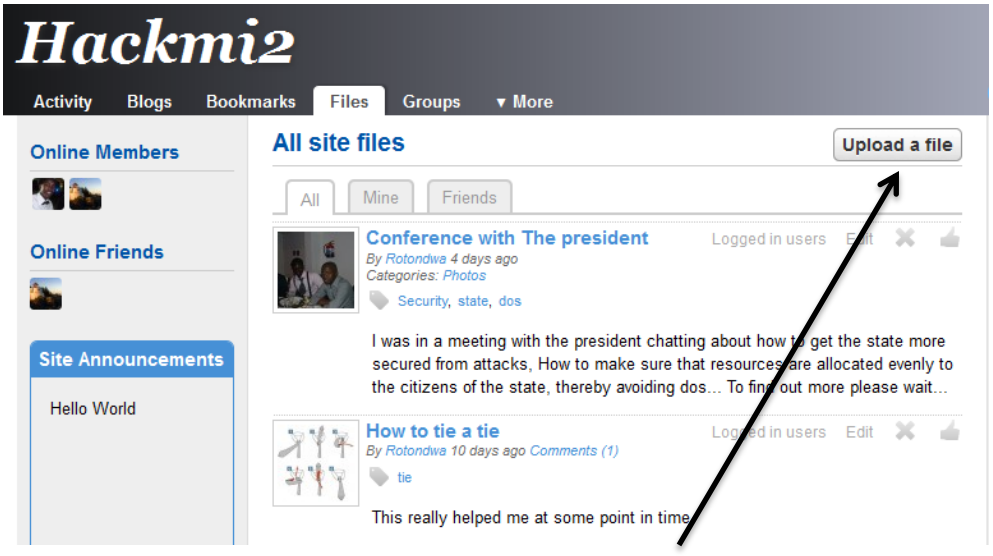

Figure 12. Uploading a file on HACKMI2

Chapter 4. Threat Modeling

4.1 Introduction to Threat Modeling

Web applications are difficult to build and are even more difficult to secure. Building a secure web application is important as developers need to know what types of threats the system is vulnerable to and how these vulnerabilities can be exploited by a an attacker. Threat modeling is a procedure for improving security by identifying the objectives and vulnerabilities, and then describing countermeasures to prevent or ease the effects of threats to a system [8]. It is an analysis technique that is used at the design stage of a system. It is typically used during the construction and design phase of the Secure Development Lifecycle (SDLC) to identify and reduce risk within a system [9]. The advantage in this

approach is that the vulnerabilities can be handled before the application is deployed and in this way it provides stronger information security to users.

Threat modeling is a tool typically used to proactively identify threats that can be classified as potential vulnerabilities in a system and provide countermeasures to prevent these attacks from being exploited to cause damage to the system [3]. Threat modeling can be used at the end of the design cycle but this makes applying security patches a cumbersome task. Threat modeling helps shape your application design to meet security objectives

The important thing to consider when using threat modeling is to determine where the most effort should be applied to keep a system secure. This can be difficult, as other factors such as when applications are upgraded, added and user requirements change or develop. Threat modeling is an iterative process where a developer defines the organizations assets, identifies applications which could affect the assets, creates a security profile for each application, identifies potential threats, prioritizes the potential threats and documents countermeasures to the threats [10].

4.2 Threat modeling approaches

There are three main approaches to threat modelling. This study employs the software centric approach as it look to strengthen security on a system. According to N. Sportsman [9] a system could be approached in the following ways:

- **Attack centric**: views the system in an attack perspective. In this approach you look at your system and try to see where it is vulnerable.
- **Software centric**: evaluates weaknesses in security controls. This approach looks to strengthen these weakness found on the system.
- **Asset centric**: evaluates assets classification and value. This approach looks at the assets on the system and classifies them according to their importance. High classification means higher security measures on the asset.

4.3 Threat Modelling Process

The threat modeling process is an iterative process that is implemented from the design phase of the application to the end of the application life cycle. This process cannot be done in a single pass as it cannot guarantee that it will identify all threats in an application. Since applications are rarely static and need to be enhanced to meet new business requirements, the threat modeling process should be repeated as the application evolves. Figure 13 below depicts the threat modeling process [11] .

i. Identify Assets

In this step you identify assets, i.e. confidential data of all users in the database of the social network, availability of data, and web server.

ii. Create an architecture overview

In this step a high level structure of the application is created showing the physical deployment components, i.e. the server of a web application

iii. Decompose the application

In this step, the application is broken down to create security profile for the application on typical area where it vulnerable, for example authentication, data flow, entry points and privilege code.

iv. **Identify Threat**

In this step, you identify threats that can affect your application assets. Threats can be identified using the STRIDE model. [2]

v. **Document threats**

In this step, you document the threat by; the type description, where the threat is targeting, attack techniques and countermeasure that could be taken to mitigate the threat. The table below shows the how this is done.

[2] This model will be explained in detail later in this chapter.

Threat Description	Attacker obtains authentication credentials
Threat target	Web application user authentication process
Risk	Low
Attack techniques	Use network monitoring software
countermeasures	Use SSL to provide encrypted channel

vi. Rate Threats

In this step, threats are rated according to risk. For example if a threat has a high risk then it must be mitigated first.

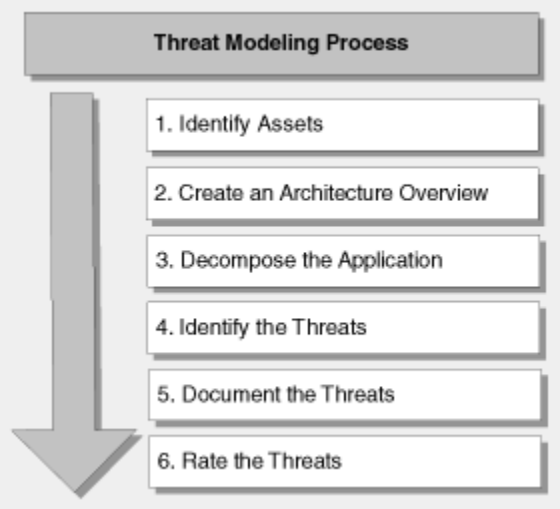

Figure 13. Threat modeling Process

4.4 Open Web Application Security Project (OWASP)

"The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications that can be trusted" [12]. The aim of this organization is to raise awareness about application security by classifying the top ten most critical web application risks. According to OWASP the threats below are raked according to their security risk.

1. SQL- injection attack – it's when an attacker insert a malicious SQL query into the web application to manipulate data or even gain access to the back end of the database
2. Cross-Site Scripting – it's when attacker inserts a malicious script into the web browser without proper validation and escaping, this script can steal data i.e. user's session
3. Broken Authentication and Session Management – It's when authentication is not applied correctly, therefore an attacker can attack it to compromise passwords, session tokens etc.
4. Insecure Direct Object Reference – Direct object reference occurs when leaks a reference an object such as files, directory or database keys. If these are not protected by means of access control then an attacker can attack these references.
5. Cross-Site Request Forgery (CSRF) – This attack occurs when a logged-on targeted user's web browser sends a fake HTTP[3] request which can include the users session cookies or other authenticated information about the user.
6. Security Misconfiguration – security configuration should be kept secure, if an attacker were to get all the security settings defined, implemented and maintained on an application then they have control on the application.
7. Insecure Cryptographic Storage – important data such as credit cards, authentication credential must be kept secure using appropriate encryption and hash functions. With weak security these can be attacked by the attacker to conduct credit card fraud or other crimes.

[3] Hypertext transfer protocol

8. Failure to restrict URL access – It is when an attacker forges a URL access into the application. The applications need to perform access control checks each time pages are accessed.
9. Insufficient Transport Layer Protection – It is when web applications cannot successfully authenticate, encrypt and protect the integrity and confidentiality of data on a network.
10. Unvalidated redirects and forwards – it when a web application directs and forward users to other web sites. An attacker can redirect users to phishing sites.

4.5 Microsoft SRTRIDE MODEL

The Microsoft threat model is defense centric, and it uses the STRIDE model to categorize threats [13] which is an acronym for spoofing, tempering, Repudiation, Information Disclosure, Denial of service and Elevation of service. This is important because each category has a specific set of mitigants once threats are analyzed and categorized [3] .The table below shows each threat and its corresponding category.

Threat	**Security property**
Spoofing	Authentication
Tempering	Integrity
Repudiation	Non-repudiation
Information Disclosure	Confidential
Denial of service	Availability
Elevation of service	Authorization

- **Spoofing** – To deceive someone for the purpose of getting access to their resources. Example: a fake internet address so that the user would think it's the real address and then will provide their credentials, whilst the credentials are sent to the attacker. This type of threat typically happens at the authentication of a web application.

- **Tempering with data** – occurs when an attacker modify data maliciously in order to hamper the system. Example if an attacker would change a patient's data in a hospital by inserting a malicious code into the database of the hospital. The attacker is said to be violating data integrity security property when they achieve.

- **Repudiation** – occurs when an attacker performs an action and claims that they did not do it. Example when a user purchases an item at a store, he must then sign for it as it will be proof for the vendor that the customer actually received their goods.

- **Information disclosure** – occurs when an attacker gains permission without a valid authentication. For example when users read files that they were not supposed to get access to. This type of attack will be violating the confidentiality security property.

- **Denial of service (Dos)** – occurs when an attacker deny service to valid users. For example when the attacker make a web application temporarily unavailable. These attacks violate the availability in security property.

- **Elevation of service** – occurs when an unprivileged user gains privilege access to a system. These threats include when the attacker penetrates all the defenses of the system and become one of the trusted systems itself.

4.6 DREAD MODEL

DREAD is a rating model that rates threats according to their impact. According to A.Shrivastava after an application has been analyzed and application assets and security threats have been discovered, it is important to note that not every threat and asset have the same priority [10]. The prioritization of threats and vulnerabilities on a system can be done by using the DREAD model the acronym for (Damage, Reproducibility, Exploitability, Affected users and Discoverability) [14]. The DREAD model is a subjective classification as

it depends on the rating the developer chooses. Typically DREAD rating scale is between [0-10] with 1 being a low occurrence and 10 high a high occurrence. The DREAD algorithm is given below and we discuss each category of DREAD.

DREAD Rating = (Damage + Reproducibility + Exploitability + Affected users + Discoverability)/5

- Damage Potential

  If a threat is possible then we would like to know how much damage to the asset this threat can do. Example: an attacker can do a lot of damage if they can gain control to the database of system or administrative rights as opposed to when he can lead information disclosure which might get a low occurrence.

- Reproducibility

  An attacker tries to reproduce a threat exploit. If it can be easily done then it the threat will get a high reproducibility rate. If the attacker depends on some race condition or time frame then the vulnerability will be difficult to exploit giving it a low occurrence.

- Exploitability

  The developer checks what resources an attacker will need to exploit the threat. Like an advance attack tool or some understanding of complex networking (low occurrence). An attacker might just need a web browser to exploit the threat (high occurrence)

- Affected Users

  The developer checks how many users will be affected should the threat be exploited. A low occurrence is when a few users will be affected and high occurrence when almost all the users get affected.

- Discoverability

  If the threat it was easy to discover then we assign a low occurrence probability if the threat was almost impossible to discover and a high occurrence if the threat could be discovered on the search engines or if information is visible from the web browser.

4.7 The importance of threat modeling web Applications

In web applications one seems to think everything is good and secure when they have their firewalls, Secure Socket Layer (SSL) and strong password are in place and get the impression that the application is secure. This is a dangerous assumption as they have not carefully looked at the components of their system and what is at risk. Threat modeling can help model the system and discover vulnerabilities within the web application. According K. Beaver [15] threat modeling will help you with the following.

- **Determine security goals**

  This can be done outlining what is in the scope, what is important and what are the assets of the application.

- **Document the architecture of the application**

  Look at how information flows from one component to the other. It is important to model how information flows from the client (web browser) to the web server then to the database.

- **Outline what needs to be protected**

  These include things like login credentials, user information and session information.

- **Find entry points and trust zone.**

  Here we look at how users will interact with the system, the entry points include user authentication, system logging, server maintenance and database interfaces.

- **Discover possible attacks from a malicious code**

It won't be easy finding out every possible attack on your system and it is almost impossible to find all the vulnerabilities in any system web application. In information security one can never get a truly secure system. Web applications are mostly prone to vulnerabilities such as input validation, system configuration and abusing of privileges, these include the following:

1. Cross-site scripting – input validation
2. SQL injection attack – input validation
3. Weak passwords – privileges
4. Session key and cookies not expiring
5. Dictionary attack – input validation

- **Determine what is important and urgent**

These are determined by the likelihood and the impact of the vulnerabilities found. It is important to first focus on vulnerabilities that would affect the important parts of the system. Some of the vulnerabilities can have low likelihood but a high impact. Others can have a high likelihood but low impact. The vulnerabilities that must be dealt with first are the ones that will have both high likelihood and high impact. This can be done using a DREAD model which is discussed at a later stage. The analysis of the system must be kept constant throughout the application as this will allows better results.

- **Determine how to mitigate these risks**

One needs to determine how to countermeasure the vulnerabilities found. It can be done by making changes to a code or other security control, like rearranging the structure of the web application components.

Chapter 5. Threat Modeling: HACKMI2

5.1 SensePost Corporate Threat Modeling (CTM)

This chapter draws on theory from the previous chapters to do a threat modeling implementation, using the SensePost CTM on HACKMI2. The motivation behind this exercise is identifying vulnerabilities that could harm the HACKMI2 web application. Since users will be using the social network for communication, it is important to protect the information that they share on the social network. Before modeling the system using the SensePost tool, it is important to have a brief overview of the SensePost CTM tool. Figure 14 show the SensePost threat modeling tool.

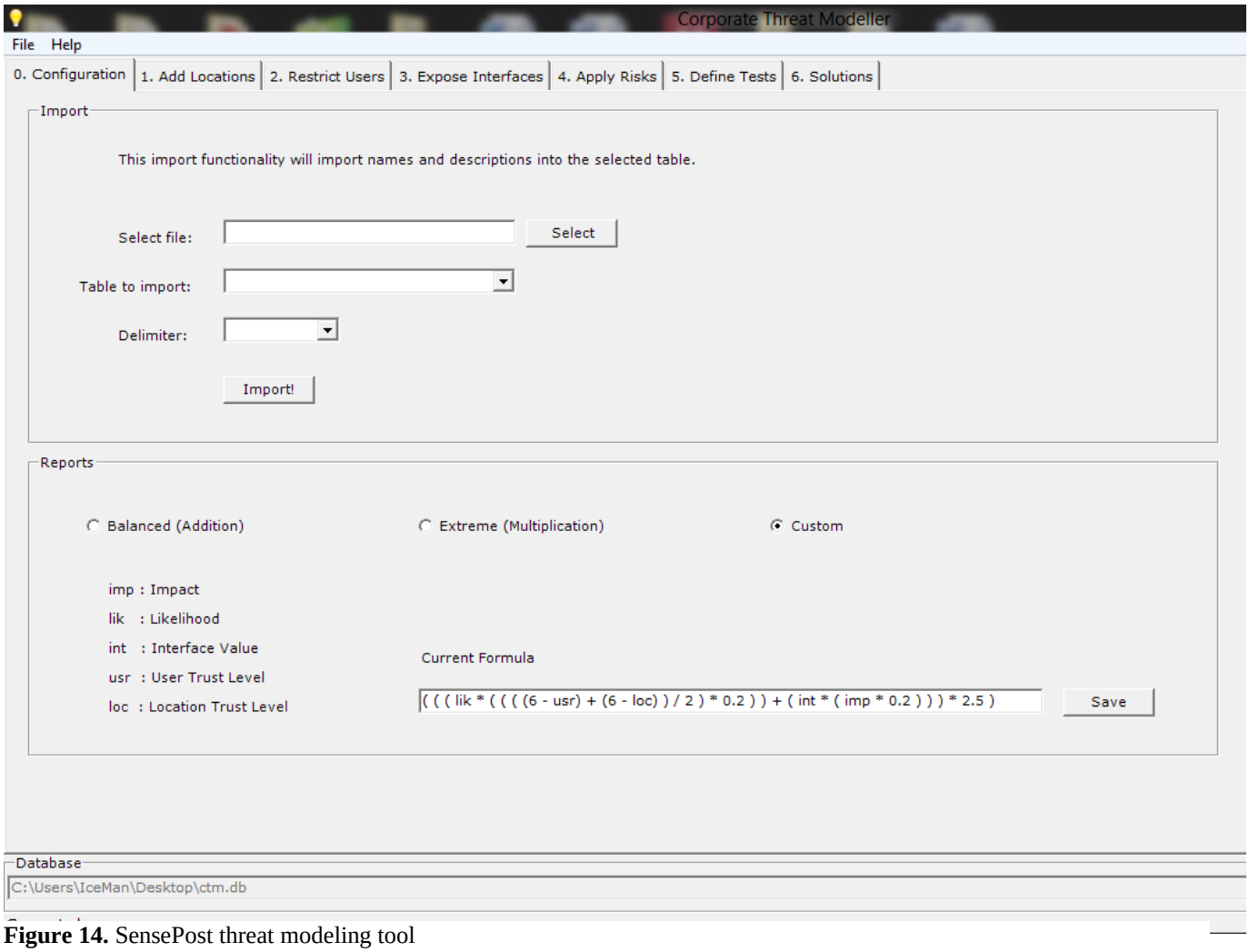

Figure 14. SensePost threat modeling tool

5.2 SensePost CTM Risk Equation

The SensePost CTM tool uses a rating model. It is very similar to the DREAD model as it rates threats according to its impact [16]. The CTM ratings is calculated by the following values

1. imp = the impact of the risk identified
2. lik = likelihood of the risk occurring
3. int = the value of an asset (at a particular interface)
4. usr = trust levels on the user
5. loc = trust levels on the location

Figure 15 below shows the SensePost CTM risk equation. What this equation means is that the risk of an attack is given by the likelihood of that attack reduced by the average of the trust level in the user and location combined with the value of the asset reduced by the potential impact [17] . Another way to view this is

Risk = applied likelihood + value at risk

Then the likelihood = attack likelihood (reduced by) the user trust + location

Value at risk = value of the asset (reduced by) the amount of asset exposed by attack

The values 0.2 and 2.5 are just there to ensure that the CTM tool fits the scales. Since the likelihood scale of any threat is between 1-5 and has a percentage between 0-25.

$$\left(\left(lik \times \frac{(6-usr)+(6-loc)}{2} \times 0.2\right) + \left(int \times imp \times 0.2\right)\right) \times 2.5$$

Figure 15. SensePost CTM risk equation

5.3 SensePost modeling overview

The CTM consist of four entities which are the locations, users, interfaces and threats. This tool uses these entities to map threats to users, location and interface [16]. Below is a description of each entity.

- **Location**

  This is will denote how much trust control the developer has at this location, as interfaces will be visible at locations and also users will exist at locations. The CTM has three defined location types.

  Physical location – the physical material of the system i.e. organization's head offices

  Network – The network of the system that will exist i.e. Internet network

  Logical – This can be the system or application or some levels of authorization

- **Users –** A system can have many different types of users. Users will be available at locations and interfaces are exposed to users via the locations. A system can have

  External users ➔ anonymous

  Internal normal ➔ administrators.

- **Interface –** The interfaces are a way users interact with a system/application. These expose the value of an asset and must be kept consistent throughout the system/application. The structure of the interface entity is that it must have the interfaces and functionality.

  Functionality - What functionality are you expecting to occur on your system? i.e. administrator or authentication functionality.

  Interfaces – What type of interface will the system have i.e. web interface, backend interface and physical interface.

- **Threats –** What are the threats to the system? The developer must assess the type of attacks that can occur at an interface that could expose the asset of the system. The most important thing one must look for at this entity is likelihood and impact.

  Likelihood – What is the likelihood of the threat? to the developer has to consider things that will make an influence, such as population of users, attractiveness, and capability. The trust level of the users and location should not influence decisions made on here i.e. administrators, then they are not likely to do some damage on the server. The CTM risk equation has already taken care of the user and location trust.

  Impact - What damage can the threat do, in a scale of 1-5 (1 being insignificant and 5 being severe)? The developer has to think about the value of the asset at risk and also take into account the value of the interface.

When the CTM tool has finished rating these threats, each threat will be associated with a percentage and colour denoting the severity of the threats. The threats that have a high likelihood and high severe impact will show the colour red meaning that we should try and protect our system from that attack. When a threat shows the colour green it means that the threat has a less likelihood of occurring and your system is safe from that particular threat.

5.4 SensePost CTM Entity Mapping

Mapping will be denoted by the symbol ' ➔'

- Users to locations

  Admin users' ➔admin functionality
  Normal users to ➔authenticated functionality
  Anonymous ➔ unauthenticated functionality

- Interface to Location

  Functionality interface ➔ system locations
  Physical interface ➔ physical locations
  Client and backend ➔ network locations

5.5 SensePost CTM: Implementation on HACKMI2

The SensePost CTM was applied to the HACKMI2 social network to identify all possible threats on the social network. The SensePost CTM has 7 steps but this study will only focus on the first 5 steps namely[4]:

0. Configuration
1. Add Locations
2. Restrict users
3. Expose Interfaces
4. Apply Risks

Step 1: Configuration:

On the configuration tab the developer needs to use the standard SensePost database file which is a standard for using this tool. After that is selected the developer can then ignore everything else and write the equation needed to calculate threats that will be found on the system. The SensePost CTM risk equation was applied. Figure 16 below shows where the threat rating equation is calculated.

[4] The other two steps are not required for this exercise

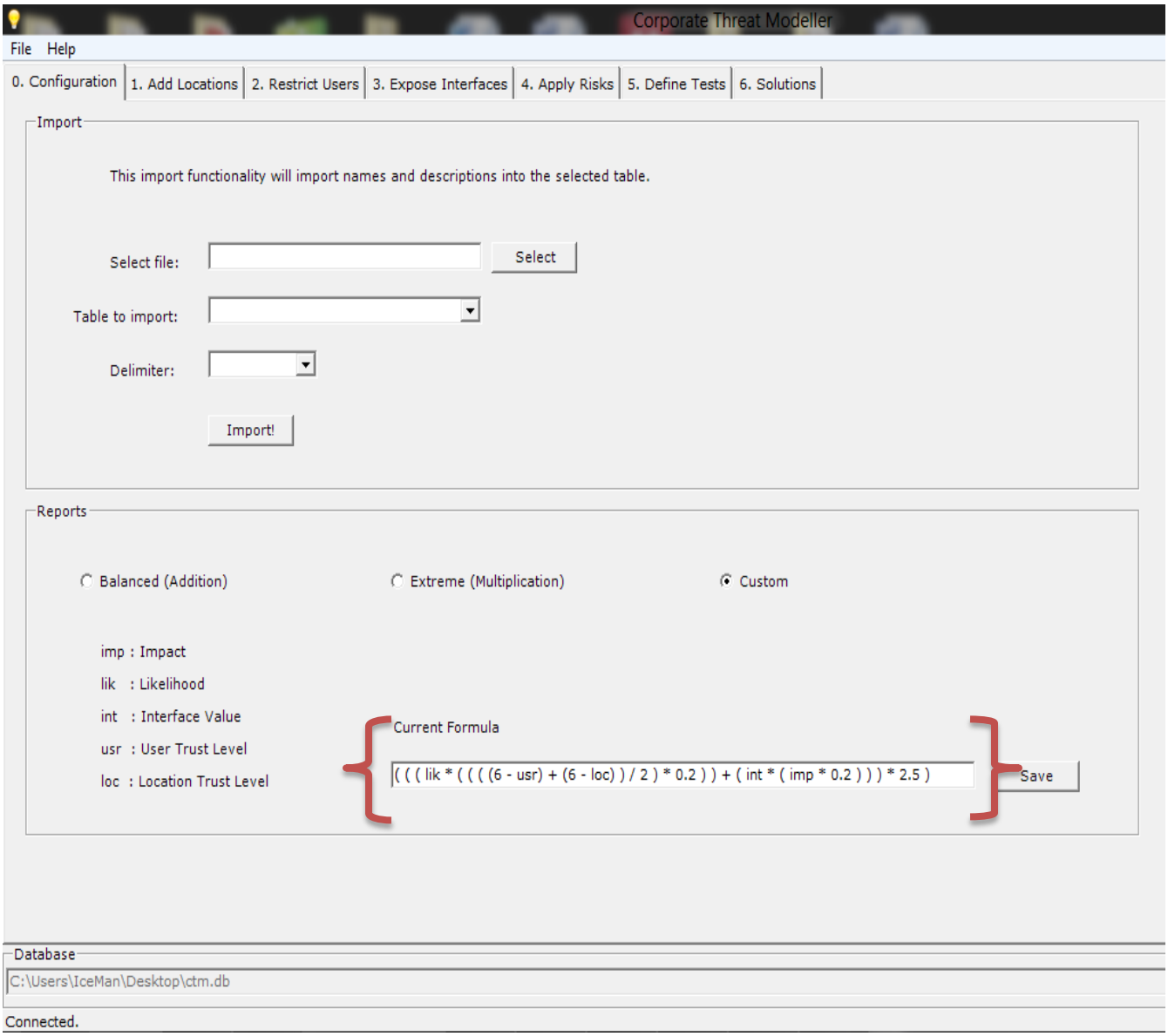

Figure 16. Threat Rating Equation

Step 2: Add locations

The HACKMI2 social network location types will be defined in terms of functional, network and Physical.

Physical location

In this location there will be a server room of the HACKMI2 web application where the servers are kept running without any interruptions.

Network

HACKMI2 has two types of networks i.e. a public network and a server room network. The public network is the network that users of the system access. They can access the Hackim2 site from any location. The server room network is where the administrators can connect to the server via the secure shell (SSH) network protocol.

Functional

The HACKMI2 web application will have some levels of authorization. These levels have been defined as administrative access, authenticated access and unauthenticated access. Administrators can access the social network via SSH or authenticated access. Users are expected to register into the system and therefore have authenticated access. Some users may access the system unauthenticated i.e. through hacking. Figure 17 below shows the add location threat modeling step.

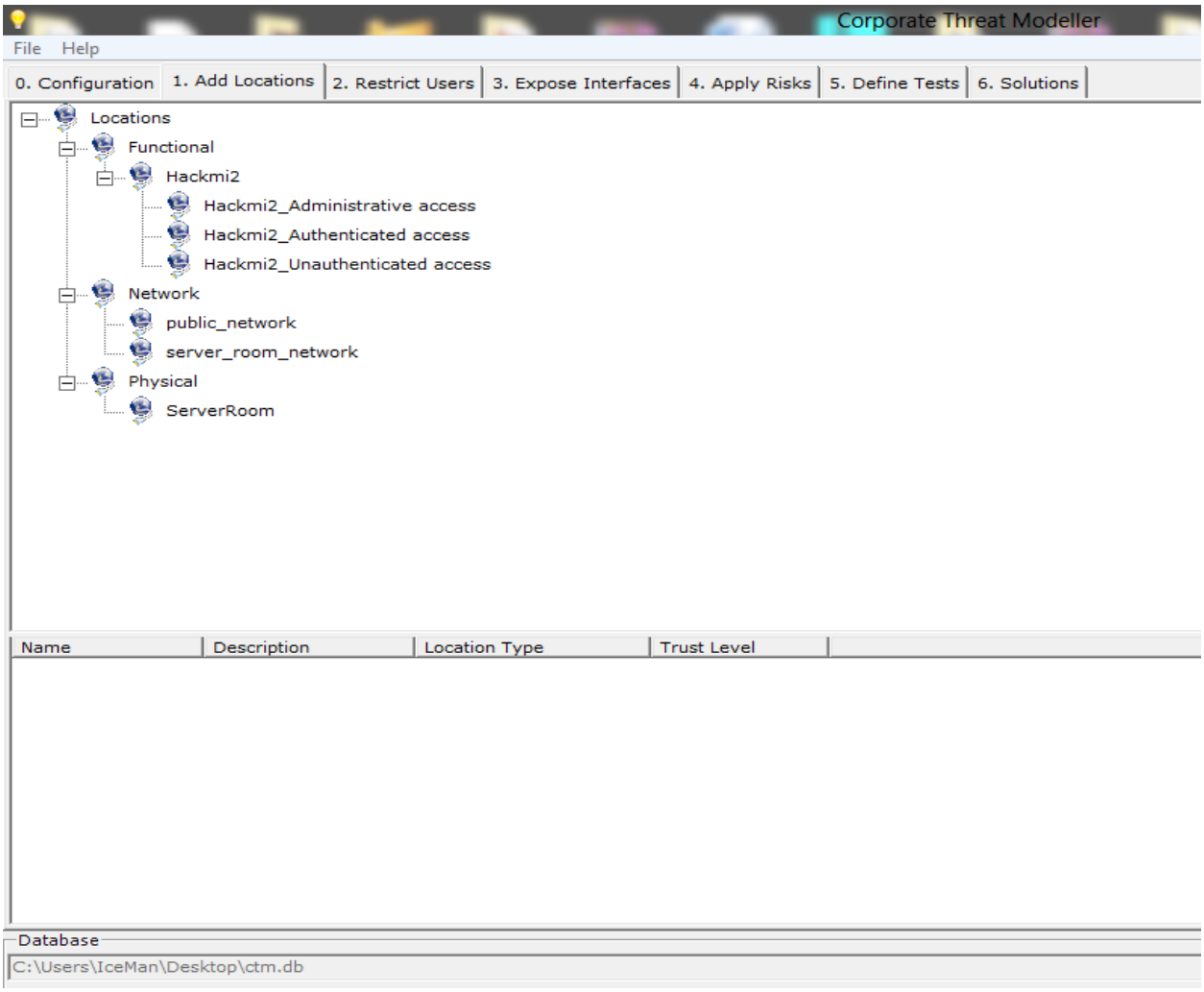

Figure 17. Add Locations

In each location the developer needs to set the level of trust. The trust level for each location is given by a scale of 1-5 (with 5 being absolute and 1 being deficient). The location trust of the server room is expected to be excellent (high trust level). The server of HACKMI2 is on an environment where there will not be any interruptions on it. The server room is always locked and only people who have authorization have access to the room. These are the things we have to consider before giving a location a rating. Figure 18 below show the server room location trust. The ratings of other locations are show in figure 19.

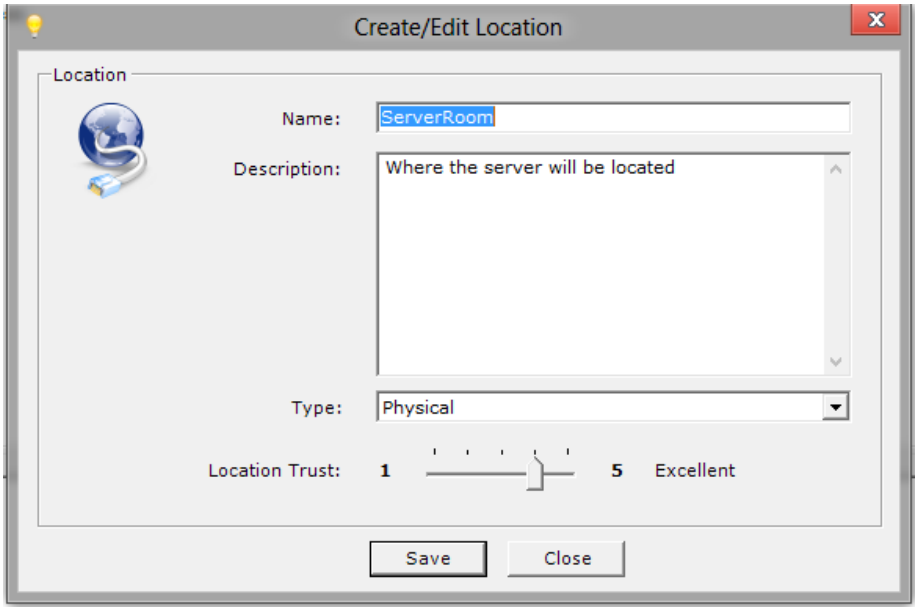

Figure 18. Server room trust level

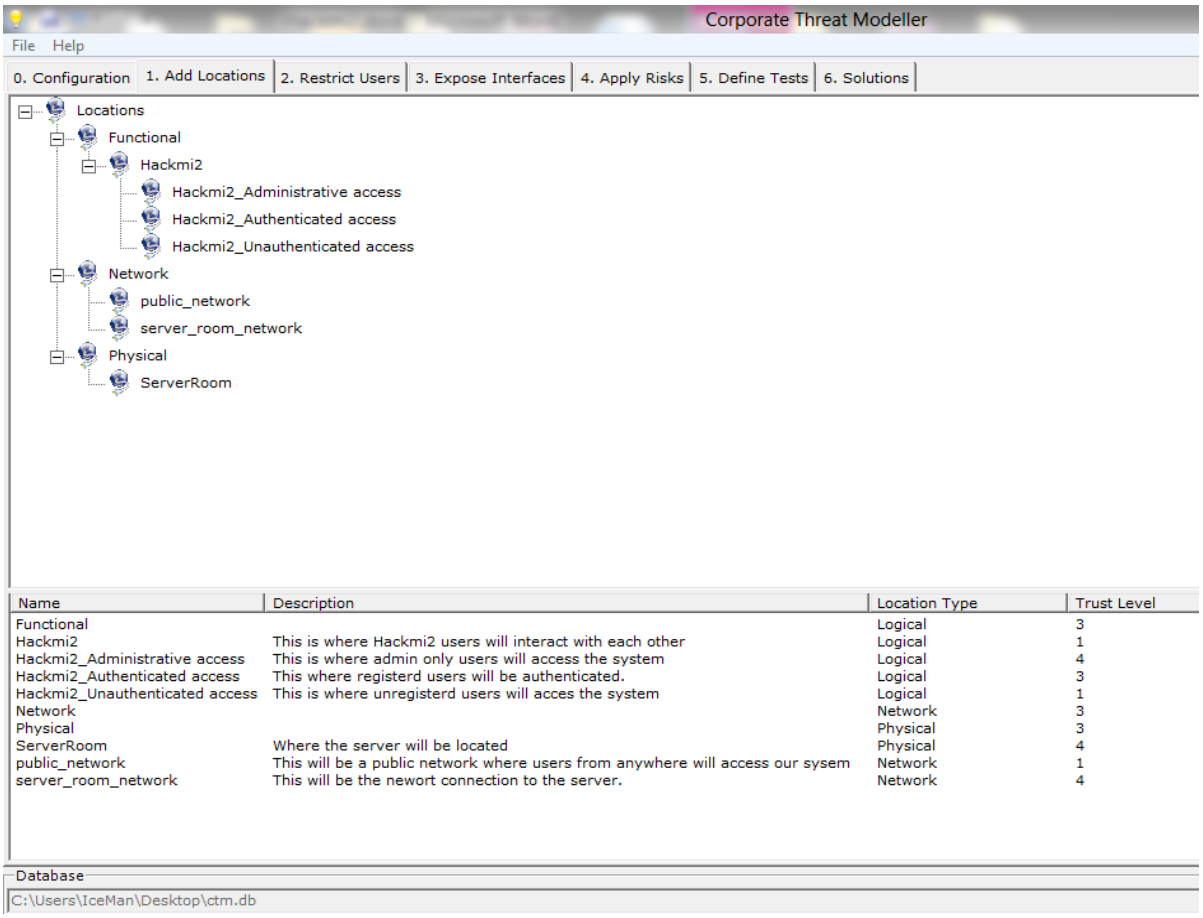

Figure 19. Trust level of all locations

Step 3: Restrict Users:

There are three types of users defined on the HACKMI2 web application; these include administrators, normal users and anonymous users. Figure 20 below shows the users defined on Hackim2.

Administrators

These will be users who will be configuring, updating and adding new features on the social network.

Normal users – these will be users who will be communication and using other features on the social network.

Anonymous users – These will be unregistered users on the social network and do not want to be known.

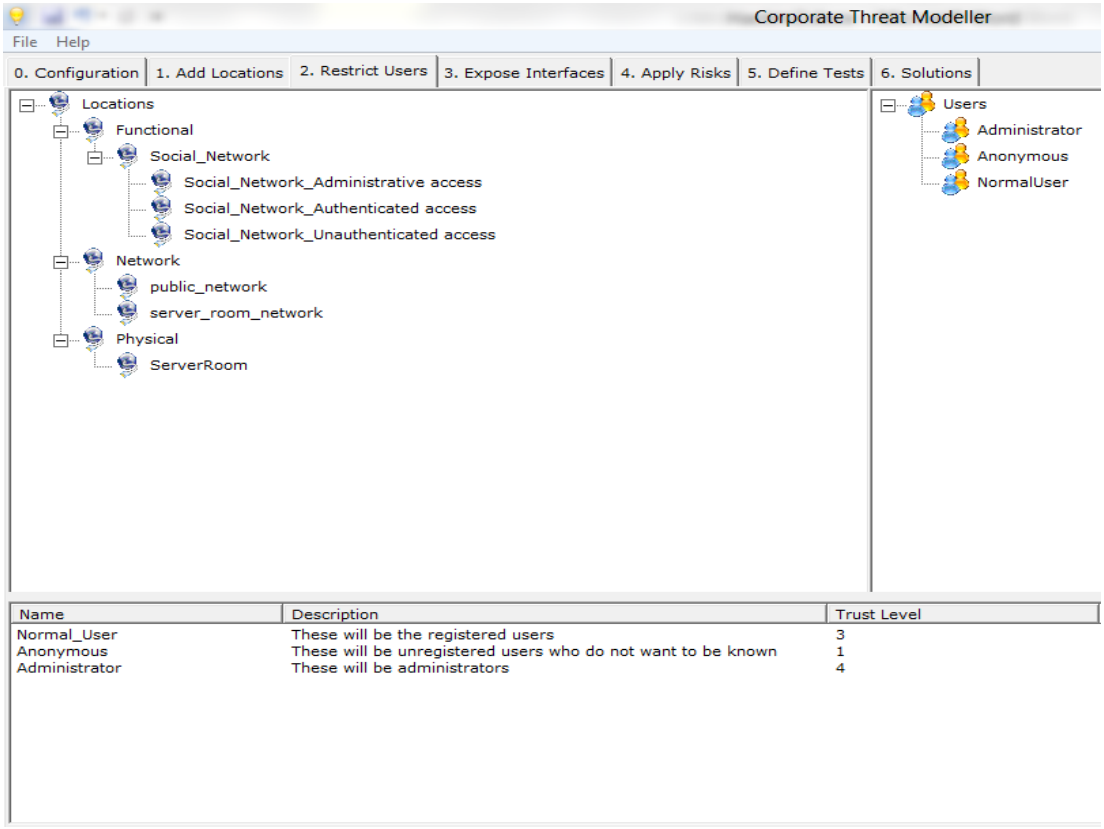

Figure 20. HACKMI2 Users

After creating each user, a trust level is assigned to each user. Administrators have a trust level of 4, a trust level of 3 for normal users whilst the anonymous users have a low trust level of 1.

Now that users for the social network have been defined, each user is mapped to a location. The mappings of users to locations for the HACKMI2 web application are shown below.

Functionality:

Administrators' ➔ administrative functionality, administrator users will access the Hackim2 and will add, updates or configures the web application.

Normal users ➔ authenticated access; these users must authenticate themselves to the server before permission will be granted.

Anonymous users' ➔unauthenticated access: these users are expected to enter the web application without authenticating themselves to the server.

Network:

All the users' ➔ public network: all the users can access the web application on a public network.

Administrators ➔ Server room network: administrators are expected to access this network. They can access the server via the SSH network protocol.

Physical:

Administrators ➔ Server room: Only these users can be allowed in the room.

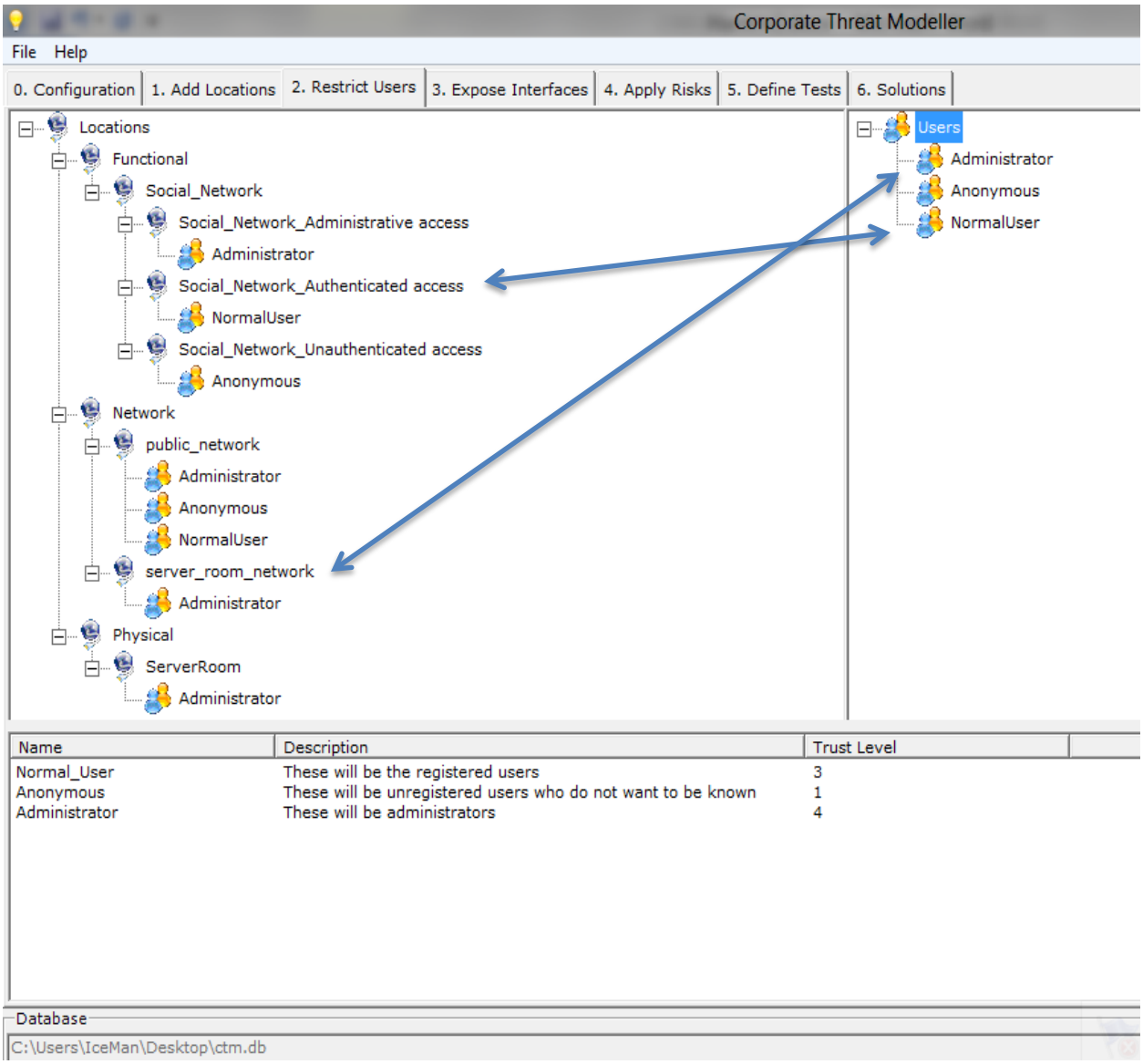

Figure 21. Mapping of users to location

Step 4: Expose Interface:

Interfaces for the HACKMI2 web application are divided into three sections namely, network interface, logic and logical.

Network interface

Web interface: allows users to connect to the HACKMI2 web application via a web browser.

SSH interface: is where only administrators are expected to use when they configure the server.

Database: allows administrators to view or add data on it.

Logical

Logical interface is where the functionality in the social network is defined; these include an administrative functionality, some administrative functions on the web application i.e. adding security restrictions on users, authenticated and unauthenticated functionalities.

Physical

The only hardware the social network has is a server. The only interface here is the server of Hackim2.

Figure 22 below shows the interface and functionalities defined on HACKMI2 web application.

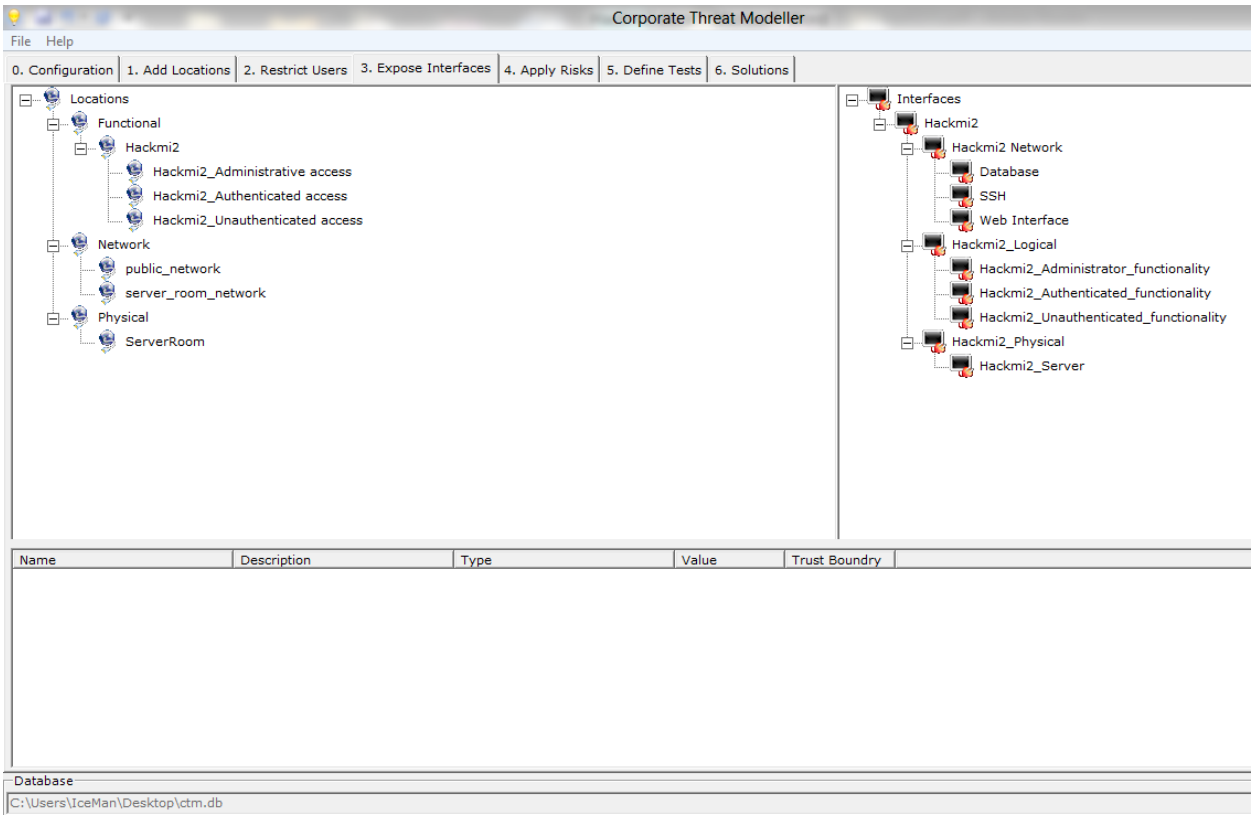

Figure 22. Expose interface

After defining each interface, the value of the interface is given and its trust boundry is specified. A trust boundry is where the developer need to specify whether they trust the interface or not, i.e. web interfaces cannot be trusted, since the interface is vulnerable to web application threats. The interface value should be kept consistent throughout the application. This value has a scale of 1-5 and a trust boundary is either true or false. The value of the interfaces on the HACKMI2 web application is imperative (interface value of 5) and false trust boundry. Since the HACKMI2 interfaces are consistent then the interfaces must have the same value and trust boundry. These interfaces are imperative as users will be using them to interact with the web application. If the interface value is low then it means that the interface value is insignificant. This would mean that the CTM would rate users hacking on these interfaces less likely since the interface is insignificant. Figure 23 below shows all the interface values and a trust boundry for each interface defined on HACKMI2 web application.

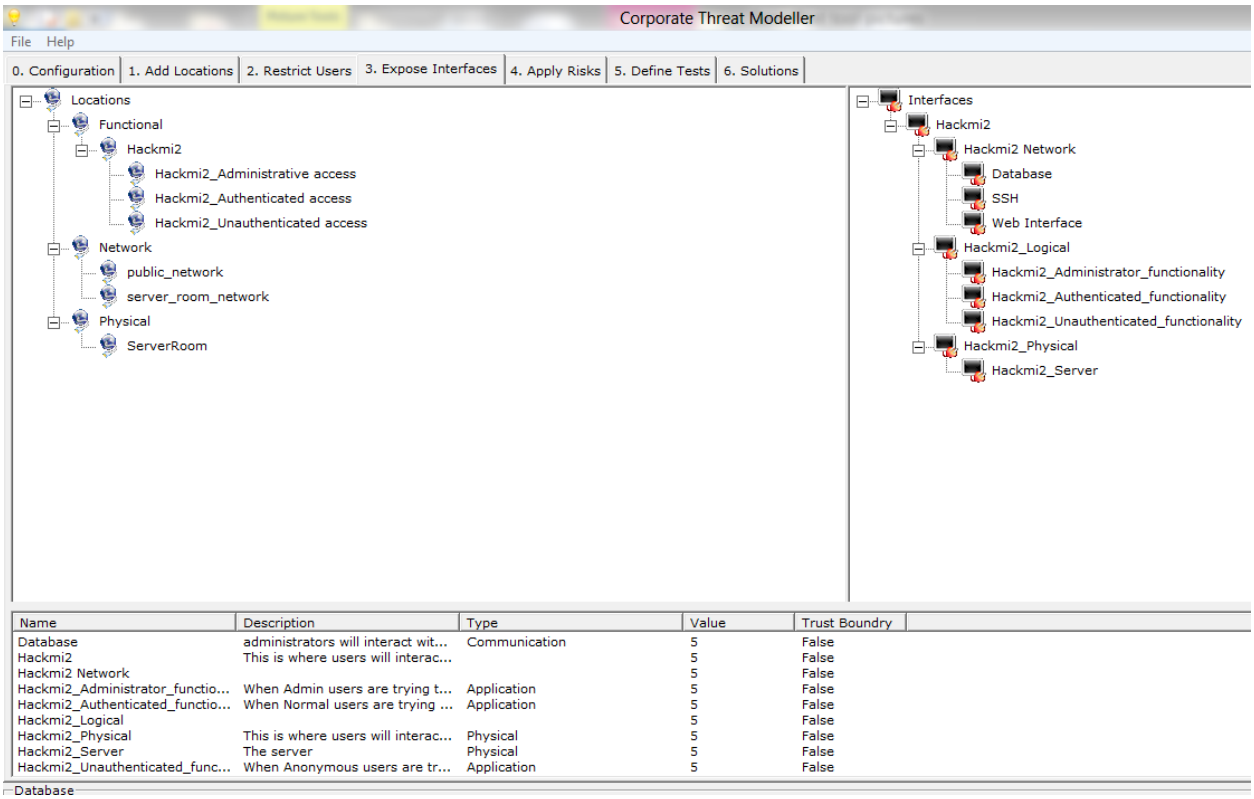

Figure 23. Interface values and trust boundry

Now that interfaces for the social network have been defined, each interface must be mapped to a location. The mappings of interfaces to locations for the HACKMI2 web application are shown below.

Functional:

Administrator functionality ➔ administrative access: Administrators must have access to the Hackim2 before they can do administrative functionality i.e. update, configure and delete features on the social network.

Authenticated functionality ➔ Authenticated access: When users are logging into the HACKMI2 web application then that is authenticated functionality, the server will grant them access to the application.

Unauthenticated functionality ➔ unauthenticated access: When users get access to HACKMI2 without authenticating themselves to the server then that is unauthenticated functionality.

Network:

Web interface ➔ public network – The web interface will be exposed in the public network. Users will use their web browser to access the social network on the public network.

Server room network ➔ SSH, Database. The administrators are expected to login into the system via the SSH and connect to the database. The HACKMI2 server will have the SSH and database interface.

Physical:

Server ➔ server room – The HACKMI2 server will be located at the server room.

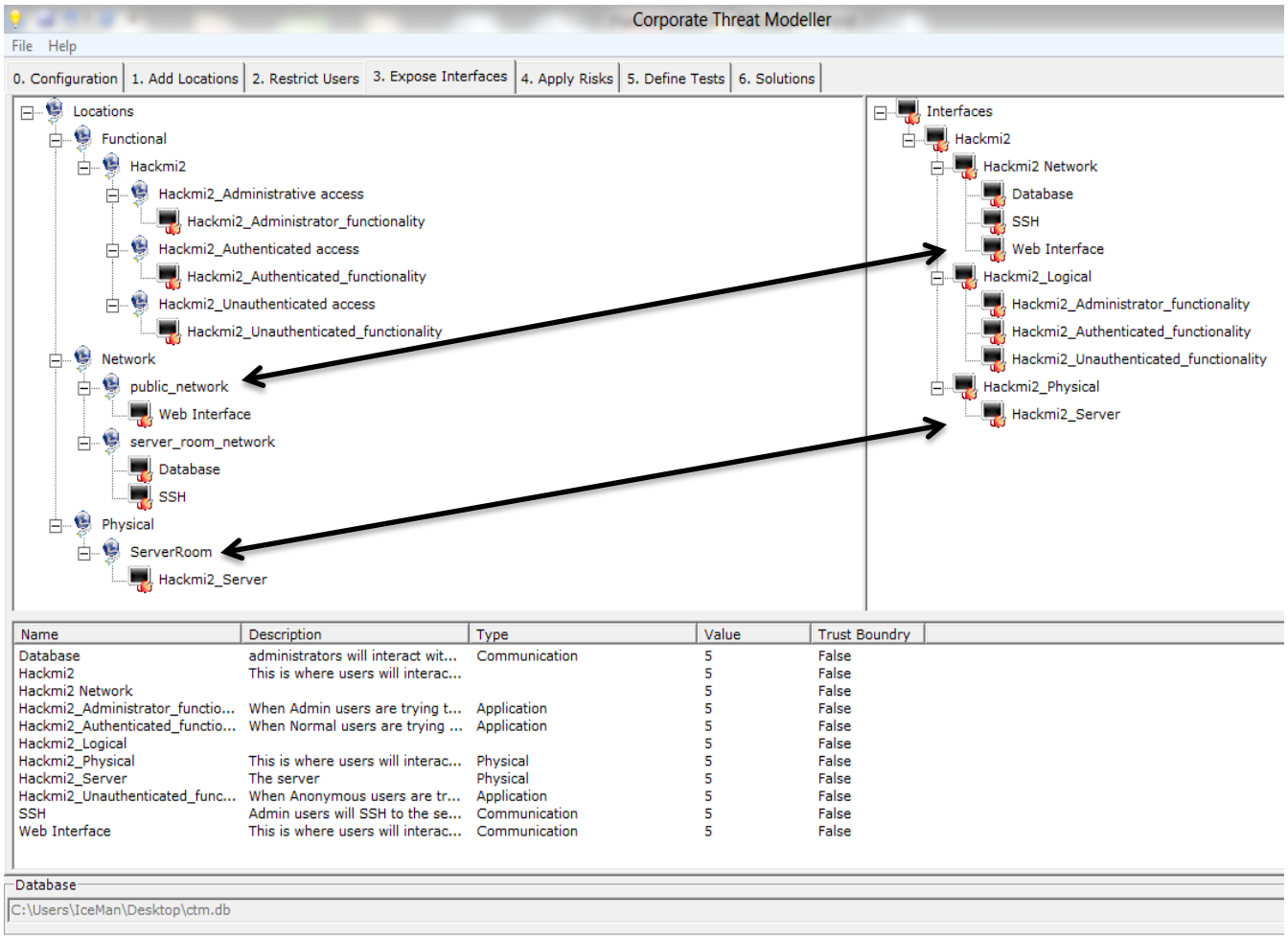

Figure 24. Mapping of interfaces to locations

Step 5: Apply Risks

According to the CTM tool threats must be defined according to its impact and likelihood. The impact and likelihood of threats has scales of 1-5 with 1 being "insignificant" and "rare" and 5 being "severe" and "almost certain" respectively. OWASP has released a list of the top ten most common vulnerabilities in web applications, with threats ranked according to its impact [12]. Each threat was placed into category; these include environmental, hacking, misuse and physical. Figure 25 shows the threats that could harm the HACKMI2 web application and their impact and likelihood. These threats were categorized and defined as follows:

Environmental:

The environmental category includes natural events such as earthquakes, flooding, etc. These also include environmental threats in which assets of the application are located i.e. severs, data centres. The focus on these threats will not be much but it is worthwhile noting these threats to the web application.

Threats:

Power Failure: It is a loss of electricity to an area; this can be a short or long term.

Impact – This threat has a "major" impact (impact value of 4) as users will not able to access the web application. The threat cannot be given an impact of "severe" (impact value of 5) as the server will not delete or lose the data stored on it.

Likelihood – The likelihood of this event occurring is "rare" (likelihood value of 1) as the server room where the server is located has never had a power failure. The University has generators to back up any power loss.

Hacking:

This category is includes attacks that that an attacker would use to gain access to HACKMI2 or harm the information assets of the web application, these threats include:

Threats:

Brute-force login attack: The goal of the attack is to gaining access to another user's account by constantly guessing the password of that user [18].

Impact - This attack could have a serious impact on the HACKMI2 application. If the attacker can get access to an administrators account, then he can completely change or delete the social network. Therefore this attack can be "severe" (impact value of 5). However if the attacker can get access to a normal user account then they can still be restricted therefore less impact.

Likelihood – The likelihood of this threat is "almost certain" (likelihood value of 5), as users will always try to login to another users account.

Cross-site scripting (XSS) attack: This attack occurs when an attacker inserts a malicious script into the web browser, this script can steal data i.e. user's session [12].

Impact - This attack could have a severe impact on the web application. and this attack is ranked 2nd in the OWASP most critical web application security risks [12]. This attack could be used as a denial-of-service and session hijacking attack; these attacks will be described later in the paper.

Likelihood – The likelihood of this attack is almost certain as uses of HACKMI2 will interact with the social network using a web browser

Denial-of-service (DoS) attack: occurs when an attacker deny service to valid users i.e. when the attacker make a web application temporarily unavailable. This attack violates the availability in security property.

Impact - This could have a severe impact as users will not be able to access the HACKMI2 web application.

Likelihood - This will have an "almost certain" likelihood, as this can be easily used with a cross-site scripting attack.

SQL-injection attack: SQL injection attacks are caused by attackers who insert a malicious SQL query into the web application to manipulate data or even gain access to the back end of the database [19].

Impact - This threat could have a severe impact on the system, an attacker could control the database and have all the username and passwords of all the users on the HACKMI2. SQL injections are ranked 1st on the list which shows they can be a serious concern in applications that uses a database

Likelihood - Databases are used in many web applications and so it makes sense to take measures to enforce protection of the data. This threat has an "almost certain" likelihood as HACKMI2 uses a database to store information of users.

Session hijacking: This attack occurs when an attacker takes control of another user's session meaning they have taken over their identity on the application. For example an attacker can hijack a user's authentication cookie and use that session to authenticate access the web application.

Impact – This attack can have a severe impact since an attacker can hijack an administrator's session and take control of the system.

Likelihood – This attack is common to web application can be triggered by a XSS attack.

Misuse:

This category includes threats that are internal to the system, as administrators that are granted access or privileges and misuse them inappropriately on HACKMI2.

Likelihood – This threat has a likelihood value of (3), threat is possible but is not almost certain, and administrators can choose not to misuse their privileges.

Impact – This could have serious impact, if an administrator could let some users the privileges of being an administrative user.

Physical:

This category is there the physical storage of HACKMI2 where threats like theft are possible.

Impact – The HACKMI2 web application will have severe impact if the server will be stolen. The data of all the users will be at the hands of the attacker.

Likelihood – This will have a likelihood value of 1 as this is very unlikely and server are usually in safe areas and often only administrators are allowed of the server room.

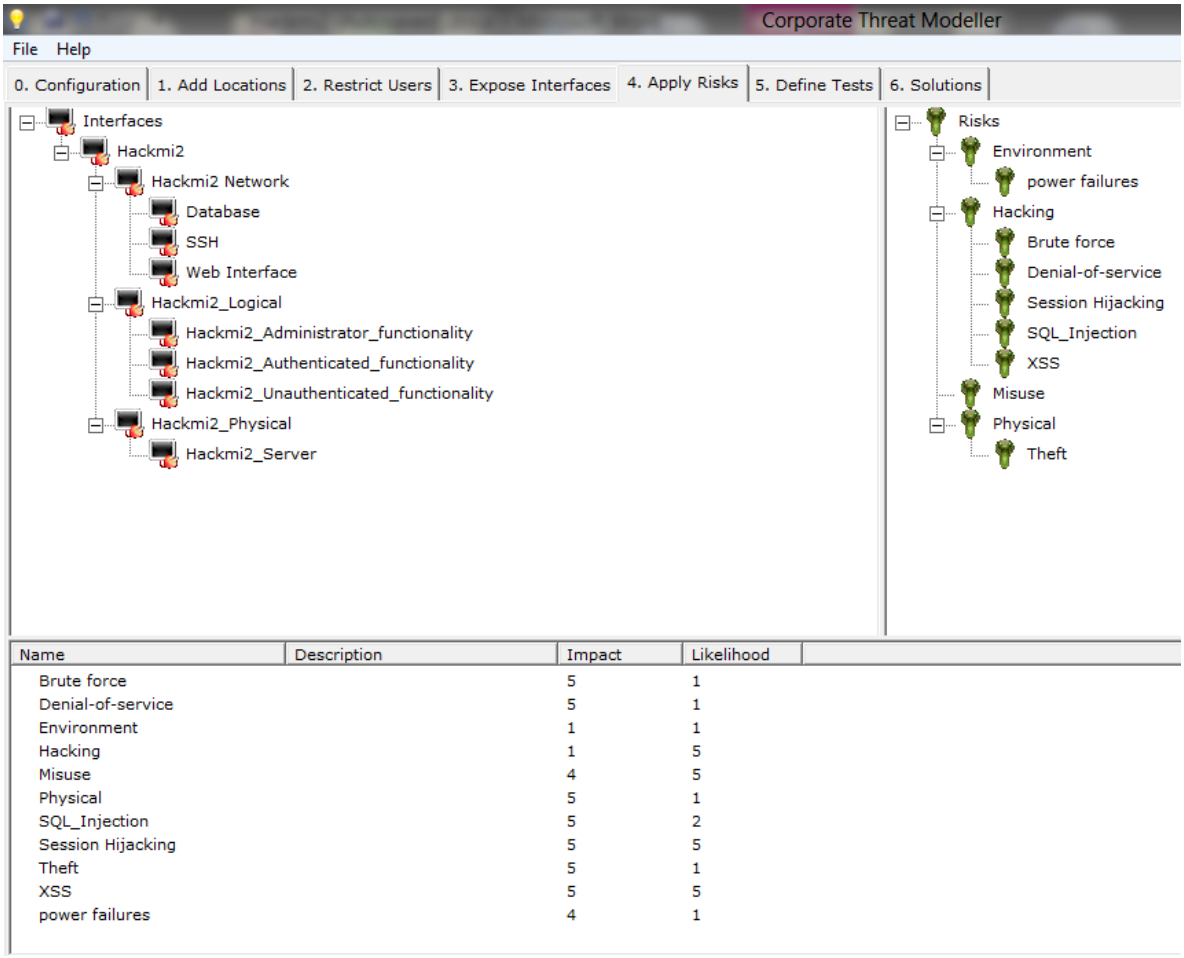

Figure 25. Defined threats with impact and likelihood

Each threat needs to be mapped with an interface in which the threat will be exposed in. The interfaces are divided into categories, these include network, logical and physical as shown in figure 26.

Network

The HACKMI2 network is vulnerable to hacking attacks, i.e. XSS, session hijacking, DoS attacks.

Logical

The logical functionality is vulnerable to misuse threats.

Server

The HACKMI2 server is vulnerable to physical and environmental threats.

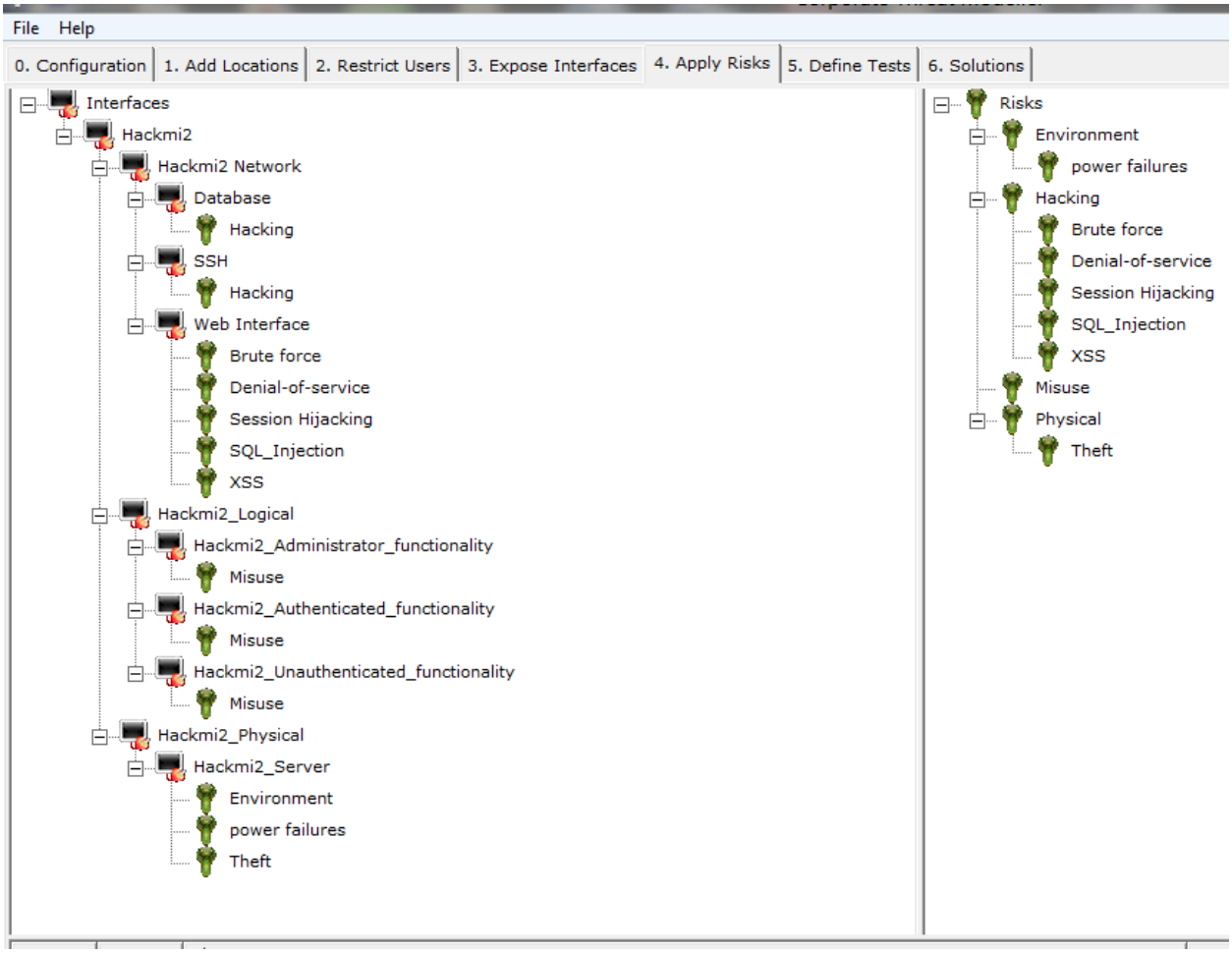

Figure 26. Threats mapped to interfaces

CTM Threat Report:

The CTM will generate the threat report after the all the steps have been applied. Figure 27 shows the threat report generated by the SensePost CTM.

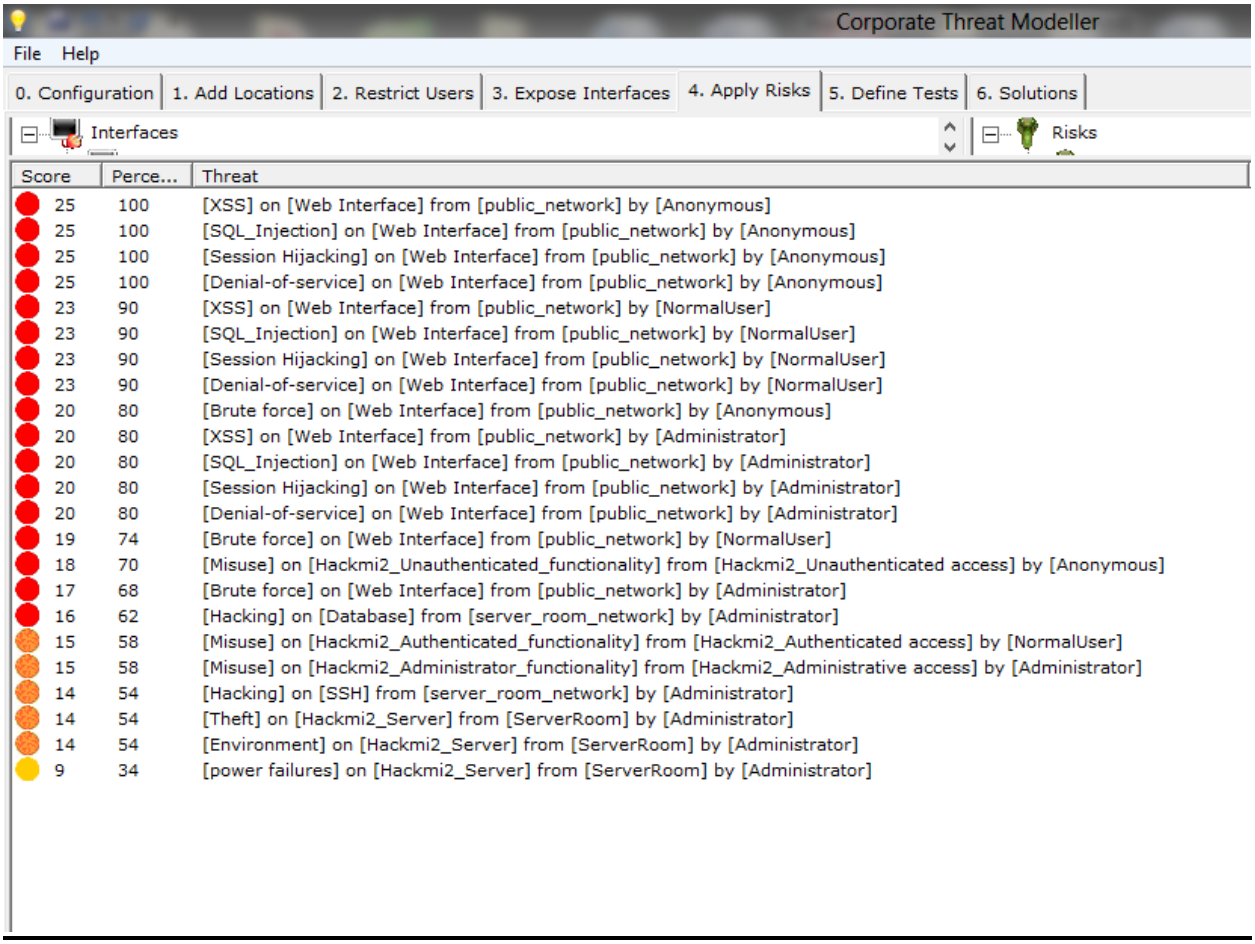

Figure 27. SensePost CTM threat report

This threat report shows the type of attack, then specifies which interface that attack will occur, the location where the threat can be found and the user most likely to user the attack. As can be seen from figure 27 the XSS attack, SQL injection, Session Hijacking and denial of service on a web interface from a public network by anonymous users are the main threats that will harm the HACKMI2 web application. It was expected these threats to be ranked high by the CTM as they had a high impact and likelihood on the HACKMI2 web application. The threat that is not likely to occur is power failure on the HACKMI2 server from the server room by administrator. This threat was expected to be rated low because of its impact and likelihood of occurring is very low compared to other threats defined.

Chapter 6. Countermeasures of threats discovered

In this chapter we discuss the techniques used to mitigate the threats found on the HACKMI2 web application. Since the software centric model was used, the focus was protecting threats from the infiltrating the HACKMI2 web application. The countermeasures for the HACKMI2 threats are described below.

Brute-force attack: This threat has much less likelihood on HACKMI2, each user is given 3 attempts to log into the application if it fails on the third attempt then the system temporarily locks the account of the user for 2 minutes. If the same user fails authentication again within 5 min period then the account is locked for longer period now and so. This will prevent automated tools from performing a brute-force attack on HACKMI2. However, it also has weakness as an attacker can lock many user accounts by 1 malicious code. This can cause a denial-of-service attack, if an attacker can keep on putting a wrong password for another user X number of times; this means that the account of the user will be lock for X minutes. If X is very big the account could be locked for weeks, months or years thus denying service to the real user.

SQL injection: This attack could not be successful on HACKMI2 web application as Elgg framework has its own security, by storing a hash[5] of a username and password. When users register on HACKMI2 they are required to enter their username and password, HACKMI2 does not store the username and password but rather store the hash of them. It is easy to compute a hash of value but it is very difficult when you have the hash and calculating the original value. When users are logging into the application, it authenticate them by comparing the two hashes, one of the username and password with the on the database. This suggests an attacker can access the database and will only see the hash and cannot use the hash to calculate the username and password.

XSS attack: This attack was applied on the HACKMI2 web application by M.Maoyi and it was successful. It was applied on the HACKMI2 blog and is triggered when the user clicks on the blog that the attack is applied to. This is an example of this attack (<script>alert('hello');</script>). The XSS attack occurs because the web browser executes the code that is in the "<script>" tags.

[5] Hash function is a function that takes an arbitrary length input and has a fix length output.

They way this attack could be prevented by ensuring that before messages, comment and blog posts are displayed to the screen it must be purified. This can be does by a function called html purifier which sanitizes the input so that the browser can displays the input. After purifying the input the browser would then display this input "<script>" instead of executing the script tags.

Chapter 7. Comparative Analysis of Threat Models

In this chapter we compare the different threat models by looking at the advantages and disadvantages of the SensePost CTM, Microsoft Security Development Lifecycle (SDL) and Microsoft Threat Analysis and Modeling tool (TAM).

SensePost CTM tool:

Advantages

- Ranks threats according to its impact on the system, thus making it easy for developers on deciding which threats to mitigate.
- The developer need to define threats, therefore it will only focus on threats applicable to the application.
- Mappings of interfaces and threats: this makes it easy to see which threat will be available on an interface.
- Maps users and interfaces to locations: Making it intuitive to see which users will be available at a location and using an interface i.e. Users will be located on a public network through a web interface.

Disadvantages

- It is very subjective when it comes to modeling the application. The trust level of users, location and interface ratings are determined by the developer. If an interface is given a low rating it would affect threats that are available at that interface i.e. if anonymous users, a public network and web interface are given high trust then the

threat report would rate the impact of threats on a public network on a web interface as insignificant.

- Must be used by a security expect, as you need to have knowledge about threats. The developer can oversee some threats as the CTM does not generate threats automatically i.e. if the developer does not have knowledge about the XSS attack, then they would not model it as one of the threats to the web application.

Microsoft SDL threat modeling Tool:

Advantages

The SDL threat modeling tool lets one create a high level overview of the system of application architecture in data flow diagrams. By just defining the data flow between the components of the application and pointing out the trust boundaries in the application landscape, this tool will help one point out the spots where security attention is needed. Reports generated by the Microsoft SDL threat modeling tool can easily be imported to Microsoft one note

Microsoft SDL threat modeling tool is designed in such a way that even people who are not security experts can use it. Microsoft SDL threat modeling tool was designed specifically for use by any software architect since this tool embeds prescriptive, at-a-glance help throughout its use.

The Microsoft SDL threat modeling tool is centered on the software, as opposed to other threat modeling tools that centre on assets or attackers. It builds on activities that all software developers and architects are familiar with; such as drawing pictures of their software architecture.

Disadvantages

Microsoft SDL threat modeling tool uses Microsoft Visio to draw DFD's. A machine without the Microsoft Visio won't be able to run this tool, which means that before one starts using SDL threat modeling tool, although it is free, one has to pay for the license to use Microsoft Visio.

Since the SDL threat model uses the high level overview, sometimes there is a need to go in to more detail. For example, defining the technology used in building a component and specific threats for the technology. The quality of the resulting report depends on the knowledge of the one who created the model.

This tool lacks the possibility of prioritizing the threats. Not every threat is likely to happen (because of other factors) and not every threat has the same impact on different business. Threats with a higher priority demand more attention, while low priority threats can be left unattended, thus saving money or leaving user-friendliness intact.

The out of the Microsoft SDL threat model tool relies on the data flow diagram. If the data flow diagram does not correctly represent the system that is being modeled, then the threat model will not show all the threats and vulnerabilities that the system is exposed to and this leaves many loopholes in the security of the system.

Sometimes one can over-represent the system using the DFDs', for example, by adding level 1 DFD while modeling using context (level 0) DFD. Figure 31 shows an example of mixed level 1 and 0 DFDs. Figure 31 has a DFD diagram with 3 external entities, logged in User, Unlogged in User and External plugin dependencies. The logged in and unlogged in users pose same threat to the system since they access the server via data flow. The server does not store the states for the user, i.e. whether a user is logged in or not. A logged in user has to send his/her logging session token along with the action or requests to the server in order for the server to recognize him/her as a logged in user. So the logged in and unlogged in user can send similar requests to the server, but the server decides which action from a user is to be allowed. Logged in and unlogged in user should be represented as one external entity in the level 0 DFD because the threat that these two elements pose to the system and the mitigation procedure are all the same. Having users represented as two separate elements brings a redundant threat and mitigation procedure to the system. The redundancy also happens when an element such as the external entities, data store and processes have more than one out-going/in-coming data flow to/from the same element. As an example, in figure 31, the server has more than one out-going and incoming data flow to and from the database. Each outgoing data and incoming data flow has the same threats, hence mitigation procedure.

Figure 31 was the first DFD used to model HACKMI2, but after the discovery of the redundancy in the diagram, it was changed to the DFD in figure 13.

Since Elgg framework is a very large, complex and yet extensible system, one cannot just wake-up and draw the DFD of the system. Drawing the DFD of the HACKMI2, as well as coming up with the correct DFD took fair amount of time. Each and every day one gets to discover something new about Elgg and had to put it into consideration for a better DFD of the system.

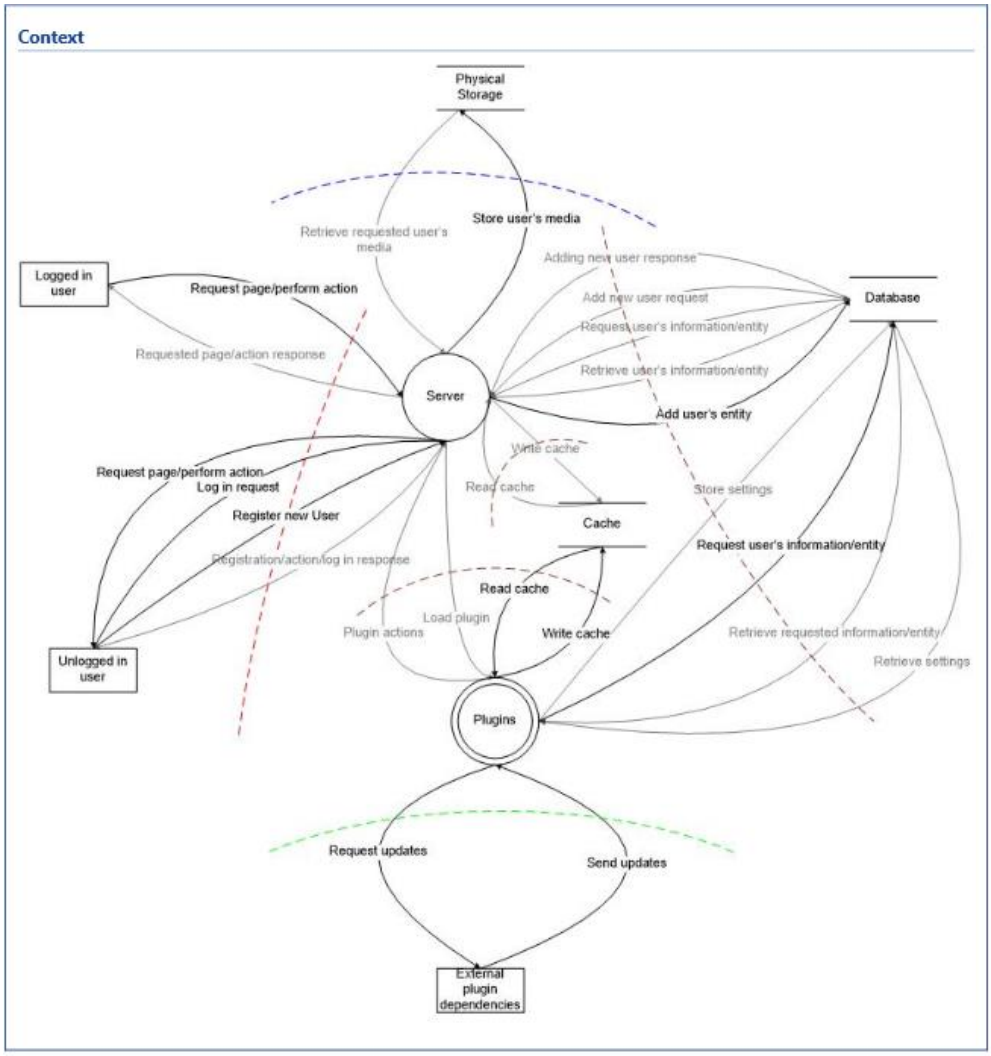

Figure 28. First DFD to model HACKMI2

Microsoft TAM Modeling Tool:

With the Microsoft TAM, to define your application context, it is necessary to first define your application requirements, and then define your application architecture. The application requirements consist of business objectives, user roles, data, and use cases, all of which are defined by business owners. The application architecture consists of components, service roles, external dependencies, and calls, and is defined by application architects.

The core function of the Threat Analysis & Modeling tool is to identify threats, while facilitating the process of defining a security strategy. Even if you are not a security subject-matter expert, you now have the ability to consistently and objectively identify threats to your software application.

Creating a threat model using the Microsoft Application Security Threat Analysis & Modeling tool is a three-phase process. First, you define your application context. Second, you model your threats on top of your application context. Third, you measure the risk that is associated with each threat. Once you have completed these phases, you can comprehend your threat models through analytics, visualizations, and reports.

The Threat Analysis & Modeling tool automatically generates potential threats to your software application, based solely on known information that you provide. The Threat Analysis & Modeling tool also has the capability to assimilate the information you provide to build security artifacts such as access control matrices, data flow and trust flow diagrams, and focused, customizable reports

An attack library is a collection of attack types along with their relevant vulnerabilities and proposed countermeasures to those vulnerabilities. Attack libraries enable software application teams to define and adopt secure engineering techniques, gain the information necessary to detect security concerns, and create relevant security test cases. Attack libraries provide a way to define, with absolutely minimal permission, the relationship between the exploit (attack), the cause (vulnerability), and the fix (countermeasure). The attack library helps ensure that various development teams understand the security assumptions and dependencies of your application

Chapter 8. Conclusion

As the number of people using social networks for communication increases, it also increases the chances of them being attacked. Methods such as threat modeling are very useful; they identify threats and provide countermeasures to these threats. This method provides a great level of security in web applications. In security we cannot measure how secure our system or application is but we know that we can never have a truly secure system. Threat modeling is a process must be repeated and must be able to adapt to new attacks discovered. We provide ways to countermeasure some of the most critical web application threats such as XSS attacks. Different threat modeling tools operate differently however they provide an efficient way of modeling a web application and discover similar threats discovered. For future work we can look at security on mobile devices such as smartphones. Using threat models to discover what sorts of security threats are found in mobile device platforms. This is an interesting environment as the number of mobile device users are increasing exponentially.

References

[1] "Internet world stats." http://www.internetworldstats.com/emarketing.htm.

[2] J. B. D. Joshi, "Web-based applications," vol. 44, no. 2, 2001.

[3] T. Modeling, P. To, and E. Application, "InfoSec Reading Room Threat Modeling : A Process To Ensure Application In tu ho rig."

[4] "What is Elgg? - Elgg Documentation." [Online]. Available: http://docs.elgg.org/wiki/What_is_Elgg? [Accessed: 15-Oct-2012].

[5] "Getting Started With Development - Elgg Documentation." http://docs.elgg.org/wiki/Getting_Started_With_Development

[6] "Introduction to ELGG, the Open Source Social Network Platform." http://www.slideshare.net/mobicules/introduction-to-elgg-the-open-source-social-network-platform-presentation

[7] "Three tier architecture." http://pic.dhe.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/covr_3-tier.html

[8] L. Desmet, B. Jacobs, F. Piessens, and W. Joosen, "THREAT MODELLING FOR WEB SERVICES BASED WEB APPLICATIONS."

[9] N. Sportsman, "Threat Modeling," 2011.

[10] A. Shrivastava, "An Approach To Web Application Threat Modeling," no. April, 2008.

[11] "Threat Modeling Process." http://msdn.microsoft.com/en-us/library/ff648644.aspx

[12] P. C. I. Dss, "Open Web Application Security Project," 2010.

[13] I. W. Applications, "LDAP Injection & Blind LDAP Injection In Web Applications."

[14] "DREAD - Risk Rating Model." http://www.cisodesk.com/web-application-security/threat-modeling-risk-analysis/dread-risk-rating-model/

[15] K. Beaver, "The essentials of Web application threat modeling." http://searchsoftwarequality.techtarget.com/tip/The-essentials-of-Web-application-threat-modeling.

[16] Dominic White (SensePost), "Threat Modeling (SensePost)."

[17] Dominic White (SensePost), "SensePost Corporate Threat Modeler (Risk Equation)." http://www.sensepost.com/blog/4798.html

[18] "Brute force login attack." http://www.computerweekly.com/answer/Techniques-for-preventing-a-brute-force-login-attack

[19] P. S. Madan, "Shielding Against SQL Injection Attacks Using ADMIRE Model Migration from Risk Analysis to Threat," 2009.