Honours Project Report

University of Cape Town


Department of Computer Science

# HACKMI2: Threat Models in Social Networks

*Author*
*Ratshidaho Rotondwa Wayne*

*Supervisor*
*Dr Anne Kayem*

*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** |

If the human brain was simple enough for us to understand, we'd be so simple we couldn't understand
*Unknown*

# Abstract

This paper focuses on threat models, and more specifically on applying the Microsoft SDL threat modelling tool on an open-source social network called HACKMI2, that we created from Elgg framework. This threat modelling tool is applied to this social network in order to identify its weaknesses and strength with respect to threat modelling in comparison to two other threat modelling tools, namely; Microsoft Threat Analysis and Modelling (TAM) and SensePost Corporate Threat Modelling (SensePost CTM). My project partners, separately applied TAM and SensePost threat modelling tools, to threat model the same social network (HACKMI2).

All these three threat modelling tools have useful upsides, but quite annoying downsides. The Microsoft SDL threat model allows to one analyse a high level of the system's architecture, but in certain cases it would be more useful to have a more detailed model of the system and the inherent threat possibilities. Furthermore, one cannot rate threats on SDL threat modelling tool as the tool only indicates areas or components where security attention is needed by simply defining the data flow between the components of the system.

On the other hand, the Microsoft Threat Analysis and Modeling (TAM) threat modelling tool allows the user to evaluate the potential threats on the system in more detailed way, and can also allow the user to rate threats. Additionally, TAM can automatically generate threats, and includes a graphical tool to view all the threats as an attack surface or threat tree. However, the images produced are rather large and so are not easily adaptable to normal screen size.

Finally, the SensePost threat model allows the user to define and rate the threats that he/she wants to focus on. In this way, the user can choose to only concentrate on threats that are applicable to the system being modelled. However, one can hardly find any information relating to threat modelling using the SensePost CTM online and/or in regular publications, which makes evaluating it rather challenging.

It was discovered that HACKMI2 was vulnerable to attacks such as the Cross-Site Scripting (XSS) attacks via blog comment, session hijacking and phishing attack carried out by taking advantage of the system's vulnerability to XSS. The implemented mitigation procedure for XSS was to filter input received from users. Solving the system vulnerability to XSS also solves the problem of Session hijacking and phishing attack.

# Acknowledgements

I would like to give thanks to my supervisor Dr Anne Kayem and Dominic White from Sense post for all the input and guidance during the course of the project. I would also like to send special thanks to my parents, brother, sister and grandmother for the support, prayers and encouragement that they have been giving me throughout this whole year. To my friends Rosa Mphasha, Sibongakonke Nkosi, Thobela Mfeti …, who played an important role during this tough year of my Honours degree, I will forever be grateful. And to the awesome guys, Maoyi Molulaqhooa and Macanda Sanele, not only did I get great group partners, but also great friends that I will always cherish. Last but not the least, what can a man do without you Jesus? I am grateful for your love and grace. Thy rod and staff really comforted me.

# Table of Contents

# Table of figures

# Table of tables

# I. Introduction

## 1.1 Motivation

Web applications and sensitive corporate information are increasingly vulnerable to security violation. Malicious users seeking to provoke security violations recognise that network-layer attacks are yesterday's news, and they have moved to new level of attacks that target web application vulnerabilities.

In general, when web services are built, the security is usually incorporated at the end of the design cycle. This is the main reason why most web applications contain a considerable number of types of vulnerabilities during deployment. If security modelling is incorporated at the design cycle, vulnerabilities are more likely to be identified and countermeasures to circumvent these identified vulnerabilities can be developed. The National Institute of Standards and Technology (NIST) estimates that code fixes performed after release can result in 30 times the cost of fixes performed during the design phase.

As shown in figure 1, the cost of fixing vulnerabilities is highest after an application has been deployed. In addition to the costs involved with engineering a fix for a given vulnerability, it is usually accompanied by a significant loss of user productivity.

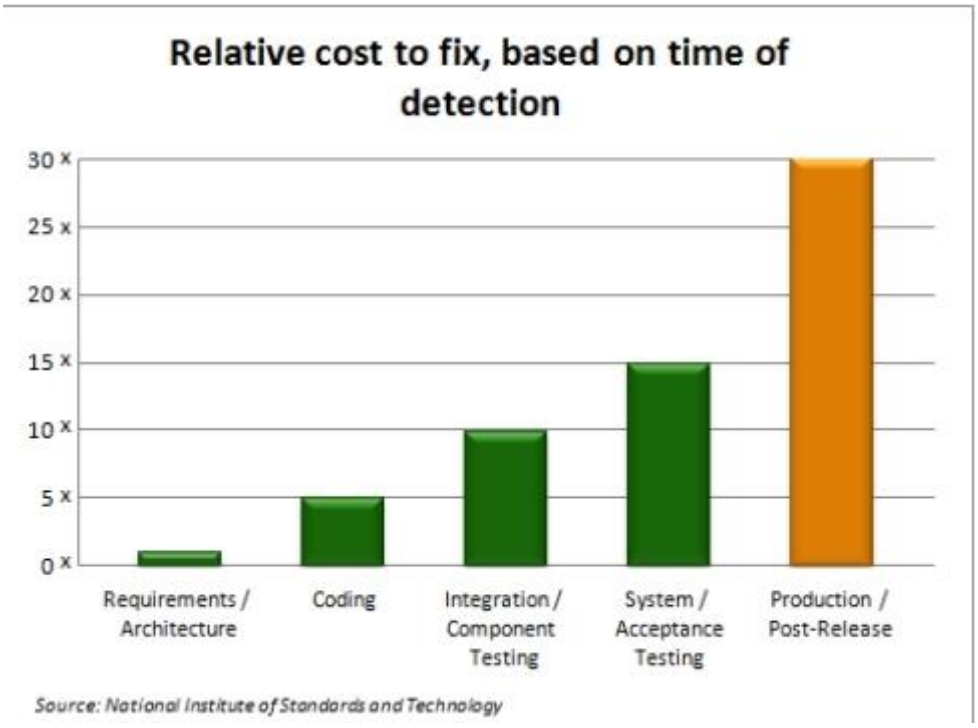



**Figure 1: Cost to fix at different phases of development**

This project focuses mainly on the threat analysis and modelling tools that can be used to incorporate security in the design phase. Threat modelling allows security engineers to identify and mitigate potential security issues early, when they are relatively easy and cost-effective to resolve, thereby helping in reducing the total cost of development.

Existing threat modelling tools have both advantages and disadvantages in the identification of potential vulnerabilities. In this project, we explore three different threat modelling tools, namely; the Microsoft **S**ecurity **D**evelopment **L**ifecycle (SDL) threat modelling tool, the SensePost **C**orporate **T**hreat **M**odelling (CTM) tool, and the Microsoft **T**hreat **A**nalysis and **M**odelling (TAM) tool. The goal is to compare their effectives in identifying of vulnerabilities.

The focus of this report is mainly on Microsoft SDL threat modelling tool. The SDL Threat Modelling Tool can be used in any issue-tracking system, making the threat modelling process a part of the standard development process. The innovative features that the Microsoft SDL threat modelling tool has are;

- Integration: Issue-tracking system
- Automation: Guidance and feedback in drawing a model.
- STRIDE per element framework: Guided analysis of threats and mitigations
- Reporting capabilities: Security activities and testing in the verification phase.

The threat modelling process will be applied to HACKMI2, an open source social network created from Elgg framework for the purpose of this project. Existing social network like Facebook and Twitter do not allow anyone to evaluate the security of their social network without prior permission. Evaluating the security of the social network belonging to these companies without prior permission is considered to be a criminal offence. HACKMI2 gives us the rights to evaluate, add, edit or remove any feature that is present in the social network. So we can evaluate the security of HACKMI2 without fear of legal repercussions.

The three different threat modelling tools will be compared based on the results we get from applying them to HACKMI2 social network.

Sanele Macanda deals with the SensePost threat model (CTM) and Molulaqhooa Maoyi will focus on the Microsoft Threat Analysis and Modelling tool (TAM).

## 1.2 Research Question

This project has based its main research questions under the following categories.

1. What are the social network security risks?
2. What sort of attacks on the social networks can be perpetrated by malicious code and how?
3. What counter measures could be employed by software centric model to identify threats and mitigate them?
4. How does the attack centric and software centric models differ with respect to identifying threats and vulnerabilities?

In general, attackers can use many paths in a social network to exploit vulnerabilities that can then be used to harm businesses, organisations or users. Each of these paths represents a risk that may, or may not be serious enough to warrant attention and so we need to identify and categorise the risks in terms of their impact intensity. Therefore, we aim to answer the research questions raised above by applying a threat modelling tool to identify security vulnerabilities in an experimental social network (HACKMI2). The threat modelling tool highlights the sorts of attacks that can be executed, based on this we build security patches to mitigate the vulnerabilities identified. As a second step, we emulated an attacker to determine flaws in the social network that an attacker might exploit and proposed some additional code patches as mitigation strategies to address these flaws. Finally, in order to address the fourth question, we compared the results to those obtained from applying TAM and SensePost CTM

tools to the HACKMI2 social network from both an attack and software centric perspective respectively.

In the next section, the report is outlined in order to describe the layout of the report.

## 1.3 Structure of the report

This section describes the parts that the report has been broken down into.

The Background chapter (Chapter 2) provides more insight about the focus of the project. The first section introduces online social networks and the risks that social networks have on assets that we entrust them with. Section two introduces Elgg framework, a social network engine that was used to create an experimental social network platform for the entire HACKMI2 project. The next section then introduces the experimental platform called "HACKMI2". The last section discusses the Ethical property with regard to the social network.

Chapter 3 introduces the threat modelling process in detail. It starts off by defining what a threat model is and why it is important. It also describes the features of a threat model, like how they represent threats and the benefits of such a presentation. Then it describes the 5 aspects or steps of threat modelling and how they classify or categorise threats. The last section introduces the Microsoft SDL threat modelling tool and discusses how one can start threat modelling one's system using this tool.

Applying Microsoft SDL Threat Modelling tool on HACKMI2 (chapter 4) follow with a step-by-step process of threat modelling HACKMI2 using the Microsoft SDL threat modelling tool. The process starts by drawing a data flow diagram representing HACKMI2 social network. From the data flow diagram, the tool then points out the spots where security attention is needed. The next step is for the user to describe the impact of the threat followed by possible mitigation procedure.

After describing the impact of the threat and the possible mitigation procedure, the tool then generate reports (chapter 5) that one can use to check the weak spots (spots where security attention is needed). The Microsoft SDL threat modelling tool generates 5 reports, the Bug report, Recommending Fuzzing, Diagrams only, Analysis report, Threat Model report and Threats list report. These reports are discussed here.

Chapter 6 discusses attacks applied on HACKMI2.

Chapter 7 describes how attacks that were successfully carried-out on the social network HACKMI 2 were carried out, as well as the mitigation procedure implemented to make the system invulnerable to the attack.

Chapter 8 discusses attacks that were not successful when applied to the social network HACKMI2, as well as the methods the social network implements to be invulnerable to these attacks.

Chapter 9 discusses the advantages and disadvantages of the three threat models used.

The Conclusion and future work chapter (chapter 10) concludes the entire report by first revisiting some important aspects learnt and then in comparison with the three threat

models, it tries to give a judgement on which one is better than the rest. The final section explains how this project can be taken further for future research.

# II. Background

## 2.1 On-line Social Networks

On-line social networks (OSN) have become the most visited sites surpassing information gatherers like Google, MSN or Yahoo!, consuming most of the time that users spend connected to the Internet, both via desktop and mobile devices[1]. There is no universally accepted definition for OSN, this report will use the working definition provided by INTECO and the Agencia Espahola de Proteccion de Datos[2];

> '*Online social networks are services that let their users to create a public profile where they can introduce personal data and information. The users have different tools to interact with each other.*'

The most well-known social network includes Facebook, twitter, LinkedIn, MySpace, etc.
For the purpose of this report, the main features of OSN and their tools are:

- Communication – Allow sharing of knowledge.
- Community – Help finding and integrating communities.
- Cooperation – Provide tools to develop activities together.

Well known as the popular three C's.

Many enterprises are embracing OSN and integrating them within their strategic plans: viral marketing campaigns; collaborative working environments within the enterprise to allow a free knowledge flow in the new paradigm known as Enterprise Social Network (ESN)[3]; image and reputation promotion of enterprises and people within the enterprises; collaborative content creation via wikis, blogging or microblogging; information exchange with faithful and potential clients, partners or competitors, search for candidates; etc.
Unfortunately, along with the aforementioned personal and corporative benefits come several web-platform-dependant threats. OSN are becoming the favourite target for cybercriminals ever since its expansion, both in and out the enterprise. In 2009, OSN were one of the main significant channels to identity theft and information leaking[1][4]. Furthermore, spam sending and malware distribution through OSN are increasing at an incredible pace[1].

The next section describes the assets in social network that are targeted by cybercriminals.

### 2.1.1 Assets at Risk in Online Social Network

Every person or organisation has at disposal several assets that must be protected to guarantee the proper course of its business. The loss, theft, destruction, reduction, or damage of any of these assets could prevent the organisation from achieving its objectives. The list below is of people's or organisations' assets that are specially threatened by OSN as described by B Sanz, C Laorden, G Alvarez and P Brinas[1]

1. **Private information**
   Private information can be stolen or used against its legitimate owner in order to harass, extort, or send hyper-contextual advertising.

2. **Financial assets**
   Financial assets can be stolen through online banking fraud, telephone fraud, or lost by decreased productivity. The most recent example of financial assets lost is a story about a guy called Bongani, as UCT ICTS named him, who connected to a free public open wireless network named "*Mug_Bean_free_wireless*" when he was sitting in a restaurant. Bongani decided to check his account balance via online banking, and as he was doing so, hackers managed to get access to his banking account. Few minutes later, he got an sms from his bank telling him that R5000 was transferred to Sudan Offshore Holdings via EFT Mug_Bean_free_wireless[1].
3. **Intellectual property**
   Intellectual properties can be stolen, plagiarised, or illegally distributed free of charge, causing economic losses.
4. **Corporate secrets**
   Leakage or theft of corporate secrets can cause economic losses, reputation damage or decreased competitiveness.
5. **Physical security**
   Stalkers, harassers, criminals or thieves can compromise the physical security of an individual or organisation.
6. **Computing and network resources**
   Computer and network resources can be massively consumed leading to denial of service or decreased Quality of Service (QoS).
7. **Corporate and personal reputation**
   The reputation of an individual or corporation can be irreversibly damaged.
8. **Digital identity**
   An attacker can steal or spoof the digital identity of an individual.

How do social networks ensure that all these assets that people and organisation have entrusted them with are safe or protected? It is the social network security engineer's responsibility to ensures and helps secure these assets. The next section discusses different ways that these assets can be compromised by an attacker[2].

### 2.1.2 Attacks in Online Social Networks

OSN have concocted a dangerous mixture of user-supplied content, open APIs, and web pages heavily loaded with JavaScript and embedded media of all types. Since attacks are aimed at the aforementioned assets, next we introduce the potential attacks that affect OSN categorised by the objectives they are oriented to as described by[1]

#### *a) Private information*

- **Sensitive data retrieval**
  Attackers are able to collect users' personal data due to their negligence when publishing private information[5].
- **Sensitive attribute inference models**
  According to Zheleva et al[4], attributes of users who are connected in social networks are often correlated, and from this, different attacks to infer the hidden sensitive values were introduced:

[1] http://www.icts.uct.ac.za/modules.php?name=News&file=article&sid=6329
[2] Anyone who impose danger to assets stored in a system, example hackers.

- **Friend-Aggregate model (AGG):** AGG looks at the sensitive attribute distribution amongst the friends of the person under question.
- **Collective classification model (CC):** Unlike more traditional methods, in which each instance is classified independently of the rest, collective classification aims to learning and inferring class labels of linked objects together.
- **Groupmate-link model(CLIQUE):** One can think of groupmates a friends to whom users are implicitly linked. In this model, they assume that each group is a clique of friends, thus creating a friendship link between users who belong to at least one group together.
- **Group-based classification model (GROUP):** Another approach of dealing with groups is to consider each group as a feature in a classifier, inferring sensitive information according to the groups a user belongs to.
- **BASIC:** In the absence of relationship and group information, the only available information is the overall marginal distribution for the sensitive attribute in the public profiles. So, the simplest model is to use this as the basis for predicting the sensitive attributes of private profiles.

- **Data Mining for demographic information**
  Using data mining techniques to retrieve public demographic data[6], one could infer unpublished personal data about other users.
- **Automated User Profiling**
  Retrieval of users sensitive data by querying social networks for registered email addresses and crawling every profile found to collect personal information[7].
- **De-anonymise OSN users**
  It exploits group membership information that is available on social networking sites, which is often sufficient to uniquely identify users, or at least to significantly reduce the set of possible candidates.
- **OSN Aggregators**
  Services that integrate several OSN which multiply vulnerabilities by giving read/write access to several social network accounts using a single weak authentication[8]

### *b) Financial Assets*

- **Cross-Site Scripting (XSS)**[3]
  A type of computer security vulnerability typically found in web applications that enables malicious attackers to inject client-side script into web pages viewed by other users.
- **Cross-Site Request Forgery(CSRF)**
  Unlike XSS which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser.
- **Bank-customer oriented Malware**
  In order to maximise their monetary benefits, malware creators target bank customers' credentials. The appearance of these attacks has increased, due to the use of social networks as a distribution channel. Example include *koobface*[4],which upon successful injection, gathers sensitive information from the victims such as credit card numbers.

### *c) Intellectual Property*

- **Contents publication property of third parties**

[3] More information given in section 6.1
[4] http://news.cnet.com/koobface-virus-hits-facebook/

This occurs when a user publishes the contents not being the legitimate holder of the intellectual property rights of the material.

- **Search engines indexation of protected contents**
  This happens when search engines indexes web pages that contain confidential information.
- **Loss of control over contents when users unsubscribe from the on-line service**
  OSN based on profiles eliminate or at least blocks all the contents associated to the profile of the user leaving the service. In platforms based on contents, the members can get to publish works without being associated directly to their profile, material remain publicly accessible.

### *d) Corporate Secrets*

- **Social Engineering**
  Manipulating people into performing actions or divulging confidential information using information found on OSN profiles.
- **Spear Phishing**
  Spear phishing is an e-mail spoofing fraud attempt that targets a specific organisation seeking unauthorised access to confidential data[9]. The email appears genuine to all the employees or members within a certain company, government agency, organisation, or group, using information found in OSN profiles.

### *e) Physical Security*

- **Location Inferring**
  Using recognisable place or feature in a picture uploaded to get the location of the user.
- **Location Inferring**
  A user's location can be extracted from the IP (Internet Protocol) connection.
- **Facial Recognition**
  Using sophisticated facial recognition algorithms to identify unknown user.
- **Cyber-bullying**
  Harassment via electronic communication tools from child to child.
- **Cyber-grooming**
  Harassment via electronic communication tools from adult to child.

### *f) Computing and Network Resources*

- **Spam and Hyper-contextualised Advertising**
  Spam is becoming a major issue for OSN, and the use of hyper-contextualised advertising (i.e. adapt advertising to users preferences) increases the possibility of the junk message being read.
- **Botnets**
  Attacks designed solely disable infrastructure to those that also target people and organisations.

### *g) Corporate and Personal Reputation*

- **Sybil Attacks**
  Given a reputation system, a peer might attempt to falsely raise its reputation by creating fake identities or sybils and using them for its benefits.
- **Classes of attacks against reputation systems**
  Hoffman et al. [10] classifies attacks against reputation systems based on the goals of the reputation system.
    - **Self-promoting:** Attackers manipulate their own reputation by falsely increasing it.

- **Self-Serving/Whitewashing:** Attackers escape the consequence of abusing the system by using some system vulnerability to repair their reputation. Once they restore their reputation, the attackers can continue the malicious behaviour.
- **Slandering:** Attackers manipulate the reputation of other nodes by reporting false data to lower their reputation.
- **Orchestrated:** Attackers orchestrate their efforts and employ several of the above strategies.
- **Denial of Service (DoS):** Attackers may cause denial of service by either lowering the reputation of victim nodes so they cannot use the system or by preventing the calculation and dissemination of reputation values.

### *h) Digital Identity*

- **Credential Theft**
  The attacker can use technical hacking techniques like session stealing[5].
- **Profile Cloning**
  This consists of identifying a victim and creating a new account with his real name and photograph inside the same social network.
- **Cross-site Profile Cloning**
  This way, the attacker identifies victims who are registered in one social network, but not in another and steals their identities creating accounts for them in the network where they are not registered.

Online social networks that already exist have Intellectual Property[6] (IP) rights that anyone who uses them must obey. This gives limits to an individual from evaluating the security of these social networks.

To avoid any legal repercussions, an open-source social network named 'HACKMI2' (HACKMI2.cs.uct.ac.za) was created. HACKMI2 was created so that we can freely add, remove or change features in the social network, thereby giving us all the power to control or evaluate the security of entire social network.

HACKMI2 is created from Elgg Framework, which is an open source engine for social network. More insight on Elgg is given in section 2.2. HACKMI2 is an experimental platform, its security will be evaluated, and if any threats are discovered, possible mitigation will be provided and implemented to make HACKMI2 invulnerable to these threats.

## 2.2 Elgg

Elgg is a social network framework that provides the basic functionality to run or create a social network. Elgg is extensible and so makes adding new functionalities easy via plugins that are described in section 2.2.6. In the following section, we provide an overview of the Elgg framework[11]. We chose to work with Elgg because it is open source and the award-winning social network engine, and is used by most businesses, schools, universities and associations to create their own fully-featured social networks and applications. Some of the well-known organisations that are powered by Elgg engine include Australian government, British government, The World Bank, Aerospace, NASA, Harvard University Extension School, Stanford University[12], just to mention a few. This makes Elgg a much more interesting tool to work with.

[5] Described in section 6.2
[6] Refers to creation of mind for which exclusive rights are recognised in law.

### 2.2.1 Data Model

As with most dynamic web applications, Elgg use a database for the back-end for storing some of the important data. Elgg has been designed to ensure that the author of a plugin should not need to create his/her own database to support customised functionalities. This idea is facilitated by the fact that Elgg is underpinned by a flexible, generic data model. Having a flexible generic data model is advantageous in sense that if database functionalities are introduced, or new data is inserted into the back-end database to support multiple servers or other infrastructural requirements, there is no necessity of updating the plugins or customising the code.

### 2.2.2 Elgg Entities

Elgg runs all its services and processes on a unified data model, based on atomic units called entities. Examples of entities include; a user, a blog post, a group and so on. Figure 2 show Elgg data model. All entity classes extend a base class called *ElggEntity* in the Elgg framework. The ElggEntity class control access permissions and ownership.

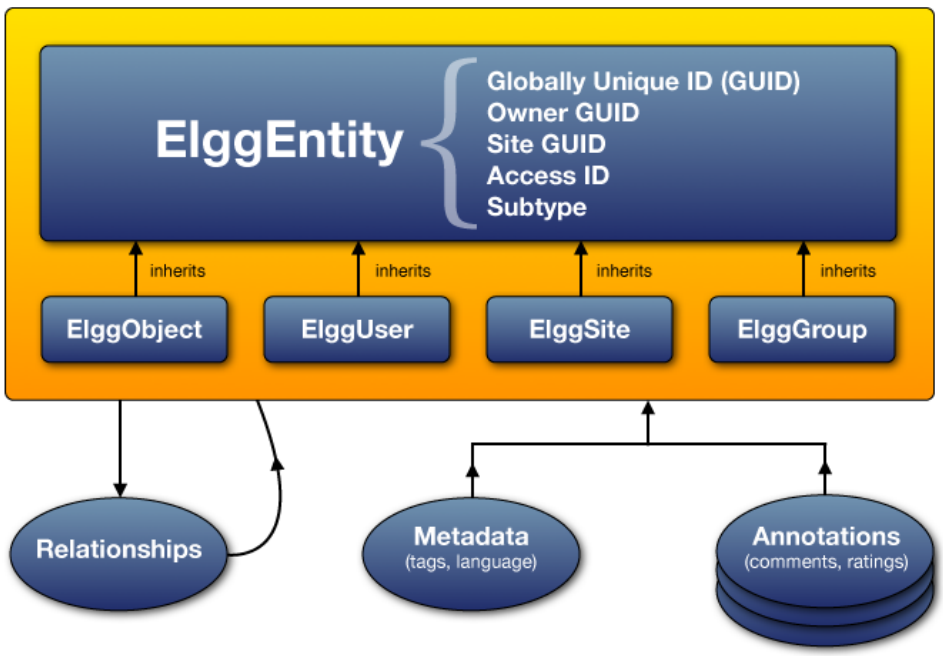



**Figure 2: Elgg data model**

ElggEntity has four main specialisations which provide extra properties and methods to more easily handle different kinds of data.

i) **ElggObject**
The ElggObject entity type represents an arbitrary objects within an Elgg install. Examples include blog posts, uploaded files, bookmarks, etc.

ii) **ElggUser**
The ElggUser entity type represents users within an Elgg install, i.e. each user in the system.

iii) **ElggSite**
The ElggSite entity type represents sites within an Elgg install.

iv) **ElggGroup**

The ElggGroup entity type represents groups within an Elgg install.

The main advantage of unified data model approach is that, apart from modelling data with greater ease, a common set of functions is available to handle objects, regardless of their type.

### 2.2.3 Actions

An action in Elgg is the code that runs when a user does something. Actions are Elgg's way of providing interactivity: every user's active participation is performed via an action. Logging in, creating, updating, or deleting content are all generic categories of actions. Actions are called through a central action handler and are registered with the framework during the framework boot process. The security handler performs security check on the action before calling the registered action file.

### 2.2.4 Elgg Event System

Elgg events are triggered when an action is performed or when Elgg framework is loading. Plugins can also trigger arbitrary events.

### 2.2.5 Plugins

The Elgg framework is very extensible. One can write a plugin to extend or to replace just about any part of the core Elgg functionality. Plugins interact with the system through events, actions and notification amongst others. Plugins can also override these functionalities such as the default theme by overriding the cascading styling sheets and /or changing other views. Plugins can add new significant functionalities to the system, such as the chat system or an event calendar system.

### 2.2.6 Metadata

Elgg uses metadata to allow users to store extra data on an entity beyond the built-in fields that entity supports. Metadata contains the owners and access ID, both of which may be different to the owner of the entity it's attached to.

### 2.2.7 Access Control

Granular access controls are one of the fundamental design principles in Elgg and was the centre of the system throughout the development. The idea is that a user should have full control over who sees an item of data he/she creates.

**Access controls in the data model**

A granular access control was achieved by having each entity, annotation and piece of metadata having an access_id property which in turn corresponds to one of the pre-defined access controls or an entry in the access_collection database table.

**Pre-defined access controls**

- **0** – Private
- **1 –** Logged in users.
- **2 –** Public data.

## 2.3 HACKMI2

HACKMI2 is a social network created for the purpose of this project. It was created from the Elgg v1.8.8 framework, so it has the same data model as the one described in section 2.2. The difference between HACKMI2 and an ordinary Elgg social network is that HACKMI2 has been customised with a better theme and interactive features such as the chatting system. This was essential for the social network so that it can attract users and have a social network that is more realistic.

Before a user can use the social network, he/she has to register. The registration process was made to be simple and straight forward since this is for academic purpose. Figure 3 shows the HACKMI2 registration page. The only required information for the registration is the users' display name, E-mail address, username and password. After registering, the user has to validate his email address. When a user registers, an email is sent to a given registering email with a link that a user has to click to validate his/her mail. If the user registering has no access to the email he/she is using for the registering process, then the account won't be validated.

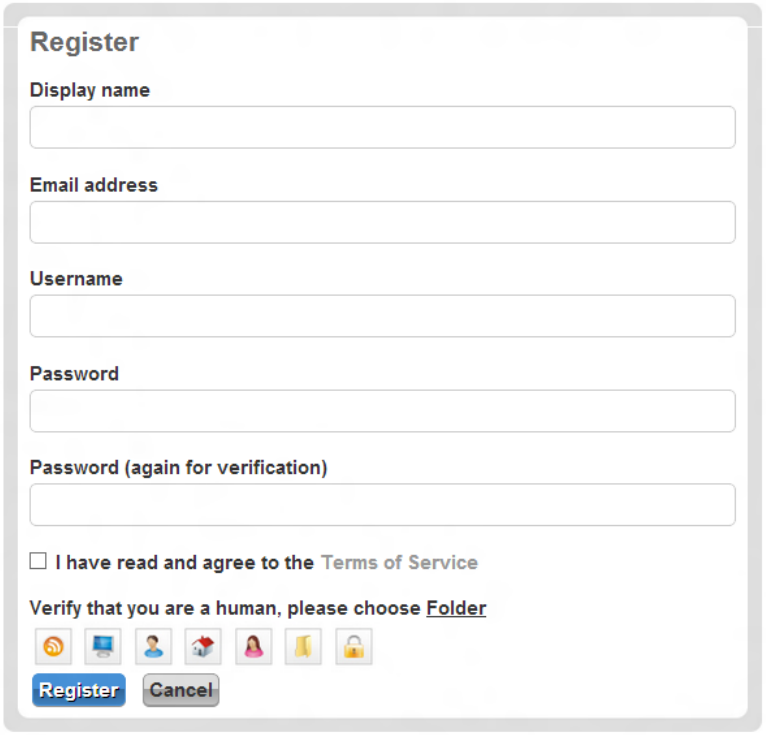



**Figure 3: HACKMI2 registration screen**

After registering, the user can login into HACKMI2 via the login page. Figure 4 below shows the login page of HACKMI2. The user will have to provide his registered username and password to be allowed access to the social network. A user can successfully login if he/she has registered and validated his/her email address.

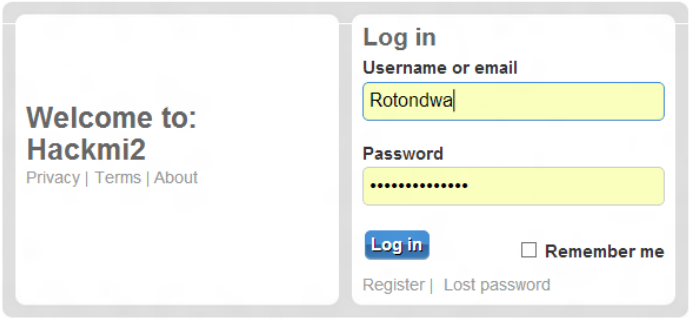



**Figure 4: HACKMI2 login page**

For in case the user has lost his/her password, HACKMI2 has a feature for resetting a user's password. The user will have to provide his/her registered email address and an email with a link to reset a user's password will be sent to the provided email. Figure 5 show the lost password page.

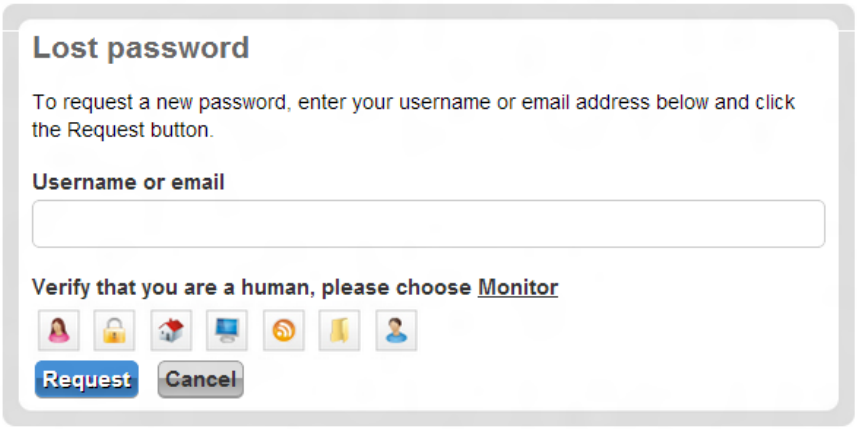



**Figure 5: HACKMI2 lost password page.**

When a user successfully logs in, he/she will be redirected to HACKMI2 activity page shown in figure 6. Figure 6 has interesting features number 1-8 which are explained below.

1. **User's profile picture or avatar**
   As with other social networks, a user is allowed to use a profile picture of his choice. HACKMI2 allows a user to upload pictures and use any of them as his/her profile picture.
2. **Share your thoughts with everyone**
   Facebook refers to this as a status. HACKMI2 termed it a thought since a user would be letting what is in his/her mind to other users. HACKMI2 implements "logged in users" access control, so only users who are logged in will be able to see other user's thoughts.
3. **Latest uploaded files**
   This window shows files that have recently uploaded. HACKMI2 has a size limit of 5 MB for all file being uploaded.
4. **Daily horoscope**
   Among other features like a clock, calendar just to mention a few, HACKMI2 choose the daily horoscopy activity to be added on the activity page. A user has to click on a horoscope to get more information related to that horoscope of interest. Above the horoscope block is a

block that tell the user the last time he/she was logged in and the day of registration to the social network.

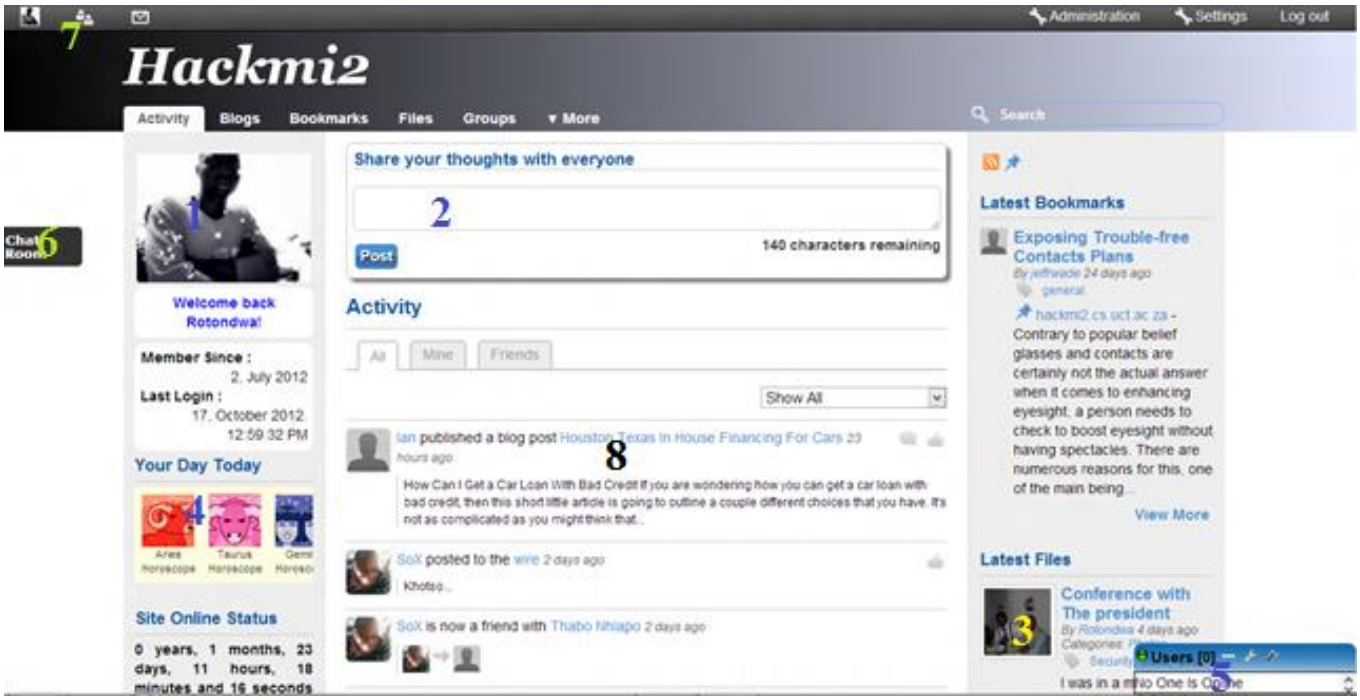



**Figure 6: HACKMI2 activity page**

5. **Private chat window**
   This window enables logged in users to instantly chat with each other privately. This window displays a list of users online, and the user has to click the username of a user he/she wants to privately chat to.
6. **Global chat**
   When a user clicks this button, a global chat window pops up with different groups that a user can interact with, as shown in figure 7. Unlike the private instant chat, this one is visible to everyone has logged in.

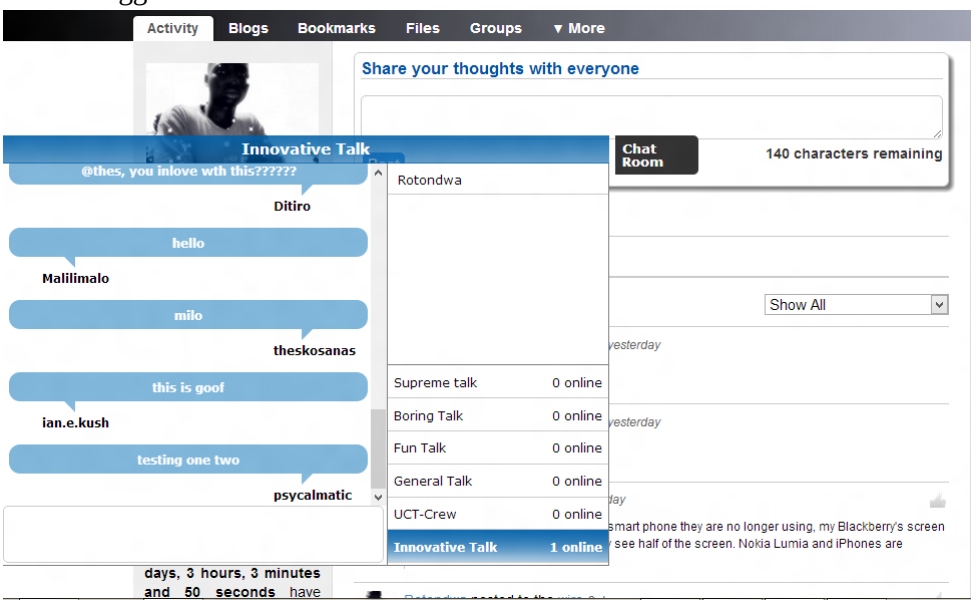



**Figure 7: HACKMI2 global chat window.**

7. **Profile, Friends and Messages**
   Number 7 shows Profile, Friends and Messages icons respectively.
   **The profile** link directs the user to the profile page where a user can change his/her profile picture; edit profile settings just to mention a few.
   **The Friends** link takes the user to the friends page where there will be a list of all users a user has become friends with in the social network through accepted friend requests.
   **The Messages** link takes a user to the private messaging page. This is where a user can send private messages to other users.
8. **Recent activities**
   This section of the activity page displays the most recent activities that users have done in the social network, such as blogging, commenting, changing profile picture or uploading files.

Users can also blog, upload files, create groups, add bookmarks, just to mention a few, by just clicking on the respective link that are shown in every page of the social network.

HACKMI2 is running on Apache webserver situated in the Computer Science server room. HACKMI2 has managed to get more than 180 users to register and was able to get them to interact with the system through its nice features like the Blogs and the instant chat.

Elgg runs on a combination of Apache web server, MySQL database system and the PHP interpreted scripting language. Due to Elgg's advanced functionality, there were some configuration requirements that had to be met:

The Apache web server needed to be installed with the following modules:

- Mod_rewrite
- PHP 5
- MySQL 5+ was needed for data storage.
- PHP 5.2+ had to be installed as an Apache module (not CGI mode or safe mode) with the following libraries:
  - GD and Freetype (for graphics processing, e.g user icon rescaling, Captcha).
  - JSON (for API functionality).
  - XML (not installed/compiled by default on all systems).
  - Multibyte String support (for internationalisation)

The server machine is situated in the Computer Science server room. The social network will have to handle multiple requests simultaneously because many users can be logged in at the same time. So the machine running the server had to be fast enough to be able to handle multiple requests simultaneously efficiently. A slow machine would repel users away because the social network would run slow.

Since there is no limit to a number of profiles that can be created, as well as the number of files that a user can upload, the space available to store this data becomes a factor. A machine with a small amount of space will run out of space early, and users won't have full access to all the features of the social network, hence will be repelled away.

Taking these two features of a machine that should be running the server, a server machine with the following specification was used to run HACKMI2.

- Vendor ID : GenuineIntel
- CPU family : 6
- Model name : Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
- Number of Cores : 4
- Number of Threads : 8
- Clock Speed : 3.4GHz
- DMI :5GT/s
- Cache : 8 MB
- Hard drive space : 3TB
- RAM : 16GB

To secure the social network, a specialised tool called threat model will be used. There are a variety of threat modelling tools that are available, but in this report we have chosen to focus on the Microsoft Security Development Lifecycle (SDL). Microsoft Windows Vista was the first major software done entirely within the SDL[13]. 'The Windows Vista Year One Vulnerability report'[7] demonstrates that significantly fewer patches had to be released for Windows Vista in its first full year release as compared to Windows XP. This puts more interest on the methodology implemented by Microsoft SDL threat model, hence the reason for the focus on this threat modelling tool.

## 2.4 Ethical Property

HACKMI2 has people who have registered from different University and companies. This was because HACKMI2 had to simulate a real life social network. Since the social network had to have real people interacting with the system, ethical clearance was obtained from the Science Faculty.

With regard to legal issues, all people had to agree to HACKMI2 terms and condition (Appendix 1) before registering. Users where allowed to register using fake names, but the email had to be their own. The privacy policy (Appendix 2) and the aim of the social l(Appendix 3) was also available for users to read before registering. People were also made aware of the main purpose of the social network.

[7] <http://www.microsoft.com/mscorp/execmail/2002/07-18twc.mspx>

# III. Thread model

## 3.1 What is a threat model?

The term threat model has the term "threat", used to mean both 'threat-agent', i.e. the person attacking a system, and as a risk, that is, what might go wrong.

The threat modelling process assesses and documents a system's security risks[14]. Threat modelling involves the identification of formal and informal entry points to a system, privilege boundary mapping and threat visualisation[15]. The entry point identification is a process of determining all possible access points to the system, whether authorised or unauthorised[15]. Privilege boundary mapping is the assignment of access rights to system objects. Threat visualisation is a formal representation of threats using techniques like attack trees, security pattern description, fault trees or attack nets which are explained in the next sections. Although threat modelling tools can be applied at any stage of the development cycle of a system, it is advised that it is implemented during the design phase as it helps software architects to identify and mitigate potential security issues, when they are relatively easy and cost-effective to resolve[8]. Therefore this helps reduce the total cost of development.

The next three sections discuss the techniques used for formal representation of trees.

## 3.2 Attack Trees

An attack tree is a term that describes a directed graph which presents the why and how the security of a system can be compromised. Every node in the attack tree represents and adversary goal and the root node represent the overall goal. Intermediate nodes in the graph represent the sub-goals the adversary has to accomplish in order to achieve the main objective. Leaf nodes represent the atom of an attack, that is the sub-goals or goals that cannot be refined any further. Attack trees have simple semantics to allow the propagation of costs that an adversary must incur to achieve a given task. However, the semantics for attack trees have limited internal structure and cannot facilitate sufficient logical reasoning about the threats they represent. For example, when are two attack paths equal/ achieve the same goal? Or what is the internal structure of an attack? Which event can have more negative impact? Figure 8 below shows an example attack tree for a computer virus. We can assume that the system being attacked in is Windows NT. The attack tree shows possible execution path of a virus that has infected a file in the operating system.

[8] <http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx>

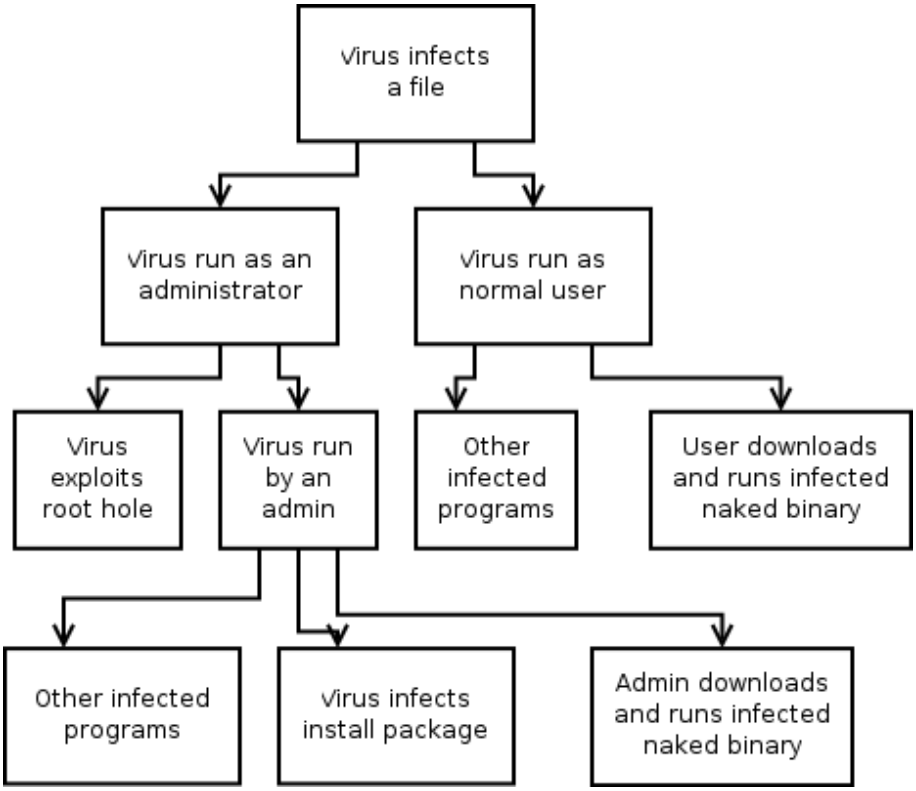



**Figure 8: An attack tree for a computer virus**

## 3.3 Security pattern description

Security Pattern Descriptions (SPD's) are documents which describe the threat of a system in natural language[15]. Security pattern descriptions are more expressive than graphical representations because they are not bound by formalism constrains. SPD's enable the capturing of atomic attributes of threats and background knowledge of the system. SPD represents threat models in a simple way, hence making the model easy to interpret.

## 3.4 Fault Trees

Fault trees are a graphical representation of interaction of system failures[16]. The failures represent system vulnerabilities which present threats to the system. A node in the fault tree represents an event and the edges represent a causal-effect relationship between events.

## 3.4 Attack nets

McDermott[17] defined an attack net as a Petri Net with a set $P=\{p_0p_1p_2\ldots p_n\}$ of places representing interesting states or modes and a set $T=\{t_0,t_1,\ldots,t_n\}$ of transitions that represent input events, commands, or data that cause one or more security relevant entities to change state. Places represent state or known knowledge that might cause a change of state in one or more linked places. The places are linked to transitions by unidirectional arcs, which represent the casual effect relationship. An attack net has a set of tokens S held in places and the movement of tokens between places along a given direction represent the progress of an attack. Attack nets present a departure from fault based analysis and attack tree threat representation by separating events from goals. The separation of events from goals enhanced the descriptive power of the representation, hence allowing security analyst to investigate atomic components of attacks.

## 3.5 Why build a threat model?

Threat models are built to identify potential threats so that one has a solid security strategy to guard against them. We cannot feasibly build a secure system until we understand the potential threats against the system. It is important to realize that threats are the by-product of an application[14]. This is why it is crucial to first understand the application context, before one begins trying to defend it.

There are two general approaches to threat modelling, namely *Attacker-centric* (AC) and *System-centric* or *threat-centric*. The following section gives a brief description to each of these approaches.

## 3.6 Attacker-Centric Approach

The Attacker-Centric(AC) threat modelling process focuses on identifying all the possible access points to the system and the possible adversary goals[15]. In general, the attacker's aim can be categorised into one of the following groups of threats: **S**poofing, **T**ampering, **R**epudiation, **I**nformation-disclosure, **D**enial of services and **E**levation of privileges: Which can be summarised with the ***STRIDE*** acronym as is the popular consensus.

## 3.7 STRIDE Threats

STRIDE is a method of categorising a wide variety of threats into six easy to remember threat types[18], namely ***S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of services or **E**levation of privileges* . Most AC-based threat models are mainly visualised as attack trees, which makes them simple to interpret and understand. Not all threats fit easily into a STRIDE category and some fit into more than one. With its matching of threats to mitigating features, STRIDE is also a convenient way of shifting the focus from threat identification to threat mitigation. STRIDE captures only the intention of the adversary, but not his/her capabilities and system defence attributes. Capabilities in this context refer to the potential tool, knowledge and techniques the attacker might use to compromise the system.

i) **Spoofing**

Spoofing threats involve an adversary creating and exploiting confusion about who is talking to whom. Spoofing threats apply to the entity being fooled, not the entity being impersonated. Thus, external entities are subject to a spoofing threat when they are confused about what or whom they are talking to; for example, a web surfer might spoofed and thinks that accountonline.com is a real bank site or Accounts.contoso.com is spoofed when it faulty verifies that a user is giving it authorised credentials.

ii) **Tampering**

Tampering threats involve an adversary modifying data, usually as it flows across a network, or resides in memory, on disk or in database. Examples include the following:

- An adversary tampers with network packets, and change commands after the user has logged in.
- An adversary tampers with registry key, making us run any program he wants.
- An adversary tampers with a DLL file, inserting his code into his program.

iii) **Repudiation**

Repudiation threats involve an adversary denying that something happened, that is an attacker performs an action that cannot be traced back to him. Such attacks mostly happens to applications or systems that did not adopt controls to properly track and log

users' actions, thus permitting malicious manipulation or forging the identification of new actions[9].

. For example, person X might access person Y email server and inflammatory information to others under the guise of one of person Y's top managers. This information might be embarrassing to person Y's company and possibly do irreplaceable harm. This type of attack has been proven to be fairly easy to accomplish because most email systems do not check outbound email validity[19].

iv) **Information Disclosure**

Information disclosure threats involve an adversary having access to data that should not be available to him. This data includes passwords for users, copies of emails, names or social security numbers in a database. Possible examples of information disclosure attacks as described by [20]:

i. **Message security and HTTP**
   The HTTP transport layer does not encrypt or protect HTTP headers, so if someone is using a message-level security over an HTTP transport layer, their headers are in plaintext and the attacker read it off from there.

ii. **Memory Dumps**
   When an application fails, log files such as those produced by Dr. Watson in Windows operating systems, can contain the claim information. The claim information also contains private data. If this information is exported to untrusted entities, then the private information is also compromised.

v) **Denial of Service**

Denial of service threats involves an adversary who can prevent things from happening. Examples include the following:

- An adversary prevents customers from connecting to a website.
- An adversary prevents the client from getting a DNS[10] response.
- An adversary prevents the client from speaking SSL[11], and forces a downgrade to an insecure connection.

vi) **Elevation of Privilege**

Elevation of privilege threats involve an adversary being able to do something, or obtain the rights to do things, which he has not been authorised to do. Examples include the following:

- An adversary who starts as an anonymous internet user can feed commands into an application that execute as the web server.
- An adversary with a web server can make code run as the local user.
- An adversary who has the ability to logon to the machine can become an administrator.

[9] <https://www.owasp.org/index.php/Repudiation_Attack>
[10] Domain Name Server
[11] Secure Sockets Layer

The table 1 below summarises the property that each of these threat types compromise.

| Threat | Property compromised |
|---|---|
| Spoofing | Authentication |
| Tampering | Integrity |
| Repudiation | Non-repudiation |
| Information Disclosure | Confidentiality |
| Denial of Service | Availability |
| Elevation of Privilege | Authorisation |

**Table 1: STRIDE and property compromised.**

## 3.8 Threat-Centric Approach

A threat analyst employing Threat-Centric(TC) approach focuses on capturing system design and deployment flaws which can translate into vulnerabilities[15]. This approach provides mechanism of examining system design principles and deployment looking for vulnerabilities against each component of the design. TC threat modelling approach is the oldest technique of identifying vulnerabilities of a system[15]. Most TC-based threat models are visualised as fault trees of the system.

This report takes on the Attack-Centric approach. The Attack-Centric approach views the system in a hacker's point of view, and since hackers are on today's headlines, this make the Attack-Centric approach more interesting.

## 3.9 Aspect of Threat Modelling

There are five aspects/steps to the threat modelling process. The list below defines these aspects and elaborates on each of them.

1. **Identify threats**

   The first thing to do is to identify assets of interest by first modelling the system either with *data flow diagrams (DFDs)* or *UML deployment diagrams*. From these diagrams, the entry points to the system such as data sources, application programming interfaces (APIs), Web services and the user interface can be identified. Since an adversary gains access to a system via the entry points, they need to be the starting point in understanding potential threats and the dangers these threats present. To help identify security threats, "privilege boundaries" are added with dotted lines onto the diagrams. Figure 9 depicts an example deployment diagram used to explain the boundaries applicable to testing a relational database. A privilege boundary separates processes, entities, nodes and other elements that have different trust levels. Wherever aspects of the system cross a privilege boundary, there is a potential for security problems to arise. For example, in the system's ordering, module interacts with the payment processing module, anyone can place an order, but only manager-level employees should be able to credit a customer's account when her/she returns a product. At the boundary between the two modules, a malicious user could use functionality within the order module to obtain an illicit credit. As an example, someone can claim that he/she have paid for an order that has not yet been delivered and needs to cancel the order.

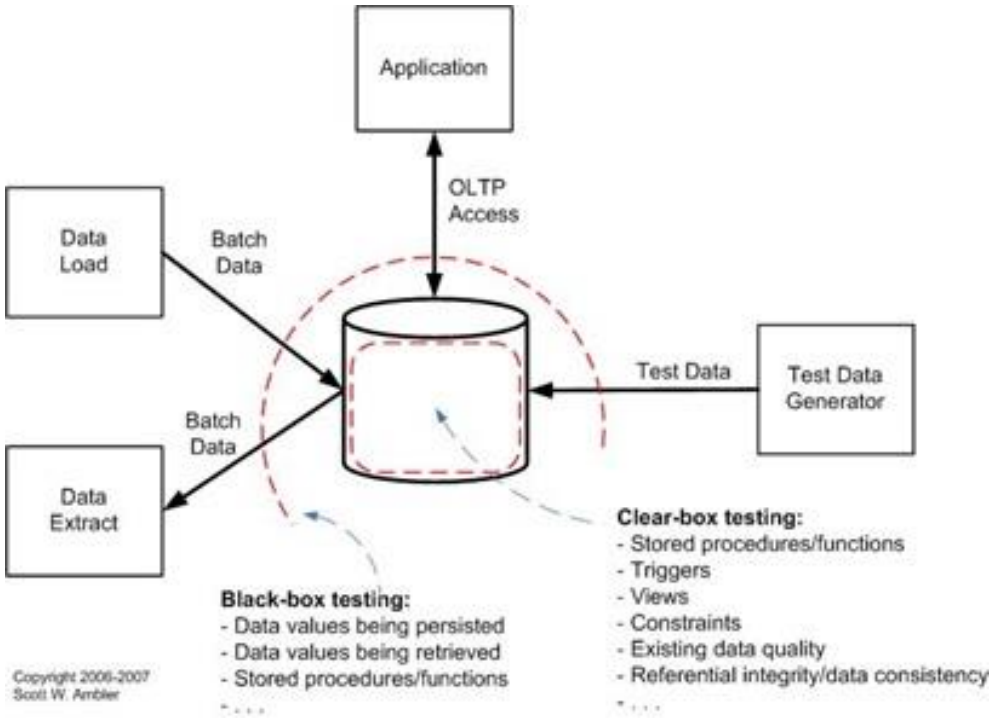



**Figure 9: A UML deployment diagram with threat/privilege boundaries**

## 2. Understand the threat(s)

In order to understand the potential threats at an entry point, all security-critical activities that occur must be identified and imagine what an adversary might do to attack or misuse your system. Questions to ask are centered "How could the adversary use an asset to modify control of the system, retrieve restricted information, manipulate information within the system, cause the system to fail or be unusable, or gain additional rights". In this way, the team can determine the chances of the adversary accessing the asset without being audited, skipping any access control checks, or appearing to be another user. In order to understand the threat posed by the interface between the order and payment processing modules, the team would identify and then work through potential security scenarios. For example, an adversary who makes a purchase using a stolen credit card and then tries to get either a cash refund or a refund to another card when he returns the purchase.

## 3. Categorize the threats

In order to categorize security threats, the STRIDE (**S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of Service, and **E**levation of privilege) approach is used. Classifying a threat is the first step toward effective mitigation of security vulnerability. For example, if one knows that there is a risk that someone could order products from one's company but then repudiate receiving the shipment, one should ensure that one accurately identifies the purchaser and then logs all critical events during the delivery process.

## 4. Identify mitigation strategies.

In order to determine how to mitigate a threat, one/security team can create a diagram called a threat tree. At the root of the tree is the threat itself, and its children (or leaves) are the conditions that must be true for the adversary to realize that threat. Conditions may in turn have sub-conditions, for example, under the condition that an adversary makes an illicit payment. The fact that the person uses a stolen credit card or a stolen debit/check card is a sub-condition. For each of the leaf conditions, one must identify potential mitigation

strategies; in this case, to verify the credit card using the XYZ verification package and the debit card with the issuing financial institution itself. Every path through the threat tree that does not end in a mitigation strategy is a system vulnerability.

5. **Testing**

   The threat model then becomes a plan for penetration testing. Penetration testing investigates threats by directly attacking a system, in an informed or uninformed manner. Informed penetration tests are effectively white-box tests that reflect knowledge of the system's internal design, whereas uninformed tests are black box in nature. For informed penetration testing, someone who knows the structure of the system well is the one who tests the system, whereas in uninformed testing, someone without any knowledge of the internal structure of the system performs the penetration testing.

This report emphasises on the **Microsoft Security Development Lifecycle (SDL)** threat model because of its significant improvement to the security of Microsoft Windows from Windows XP to Windows Vista, as described at the end of section 2.3. In the next section, Microsoft SDL threat model is introduced and described.

## 3.10 Microsoft Security Development Lifecycle Threat Model

Microsoft Security Development Lifecycle (SDL) threat modelling tool is an easy to use graphic tool for creating and analysing threat models. It captures impact assessments for proposed mitigation, and produce actionable reports as well as bugs possibilities to ensure that vulnerabilities are mitigated and/or eliminated. A threat model is a view of the software that emphasises the threats it faces based on the software components and the trust boundaries. When the a system is viewed in this way, security vulnerabilities stand out much more sharply than when the focus is solely on functionality and performance[18].

Threat modelling can be performed before a product has been implemented in code, whereas penetration testing or fuzzing can only be done after the system has been implemented. This code independence is one of threat modelling chief advantages, allowing it to be performed from the earliest design phases of a project forward.

Microsoft Security Development Lifecycle (SDL) threat modelling methodology is a 4 step process to enable security engineers to have some metric of assurance that they have found important threats[21]. The goal of the process is to improve the security of design, to document the security design activity and to learn about the security of the system as each component is evaluated. This methodology involves four major steps:

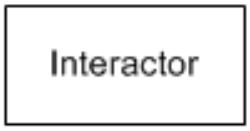
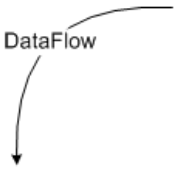
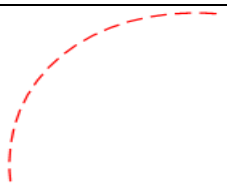
i) Diagramming
ii) Threat enumeration
iii) Mitigation and
iv) Verification.

The SDL threat modelling tool approach starts with a data flow diagram. From the diagram, potential threats are then identified. For each threat, mitigations are proposed. When the design with its mitigation has been implemented in code, the code is validated against the threat model to ensure that the mitigations work and that design functionality and performance are intact.

Next we shall discuss all this for steps in a more detailed description because the whole threat modelling processing using Microsoft SDL threat modelling tool is based on these four steps.

### *i) Diagramming*

Diagramming is done using a diagramming system derived from standard Data Flow Diagrams (DFD), with additional "*trust boundaries*". The DFD uses elements *Process*, *Data store*, *Data flow*, and *External entity*. The table 2 below defines and show how these elements are represented in the DFD diagram.

| Element | Shape | Description |
|---|---|---|
| Process |  | Any running code. For example, a .exe or a .NET Assembly. |
| Multiple Process |  | A Multiple Process represents two or more processes. |
| External Interactor |  | A user or a machine that interacts with the software and which is wholly external to it. In other words, we do not have control over it or its process. |
| Data store |  | Any 'data at rest,' such as a file, registry key or database. Shared memory segments can also be thought of as data stores. |
| Data flow |  | Data flow is any transfer of data from one element to another. Data flows are always unidirectional, as the threat impact may be very different in each direction. |
| Trust Boundary |  (red colour) | Trust Boundary. An entry point where untrusted data may be presented, or where many principals have shared access. |
| Process Boundary |  (Orange colour) | A trust boundary between processes. Functionally identical to the Trust Boundary element. |

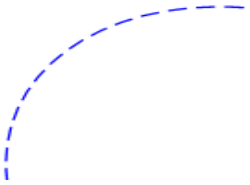
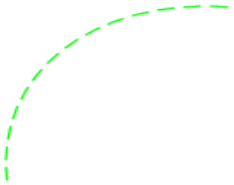
| | | |
|---|---|---|
| Machine Boundary |  (blue colour) | A trust boundary between physically separate machines. Functionally identical to the Trust Boundary element. |
| Other Boundary |  (green colour) | A trust boundary of a type distinct from the other boundary types. Functionally identical to the Trust Boundary element. |

**Table 2: DFD Elements**

Two main reasons why DFDs are used is that they are easy to understand and are very data-centric[21]. This is good because many great software attacks involve the flow of data through the system in some way, and so DFDs are focuses on mitigating vulnerabilities that emerge in DFDs. An example of a DFD is shown in figure 10. The SDL threat modelling tool use Microsoft Visio to draw the DFDs.

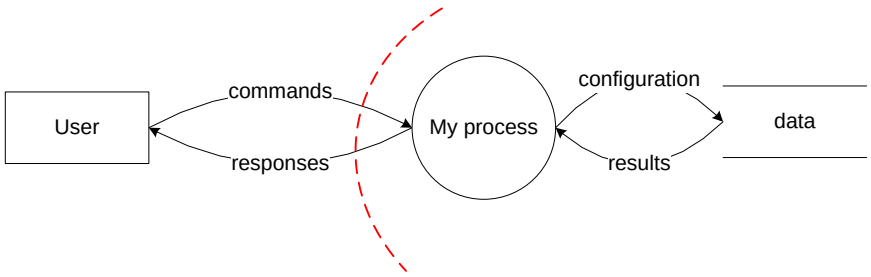



**Figure 10: Example of Data flow diagram**

## *ii) Threat enumeration*

Microsoft SDL threat model uses the diagrams in a technique called "STRIDE per element" to provide guidance for non-experts, as well as repeatability. In the STRIDE methodology, threats are grouped into one of the following six categories described in section 2.3.7.1. So, "STRIDE per element" each element is associated with one or more of the threats in STRIDE. Table 3 below summarises "STRIDE per element", i.e. to which category in the STRIDE do threats in a particular element belong. An "x" marks the category that a threat belongs to in the STRIDE model.

| | S | T | R | I | D | E |
|---|---|---|---|---|---|---|
| Process | x | x | x | x | x | x |
| Data Store | | x | x | x | x | |
| Data Flow | | x | | x | x | |
| External | x | | x | | | |

**Table 3: STRIDE per Element**

## *iii) Mitigation*

The SDL threat modelling training and documentation discusses four approaches to mitigation, in order of preference: redesign, use 'standard' mitigations, such as Access control list, use unique mitigations with caution or accept risk in accordance with the system's policies.

In is important to connect a model to practical resolution on a number of levels. First, improving system security is the goal of threat modelling. Providing a connection between an identified problem and a way to address the problem makes it easier. Secondly, there is a psychological component. Telling an engineer that there is a problem without providing fix information will frustrate many engineers[21].

### *iv) Verification*

In this stage, a number of heuristics for validating threat models is provided, including graph analysis of diagrams, checking that the final diagrams reflect the final code, that STRIDE per element have been enumerated, that the whole threat model has reviewed, and that each threat is mitigated.

The diagram (figure 11) below summarises the entire SDL process.

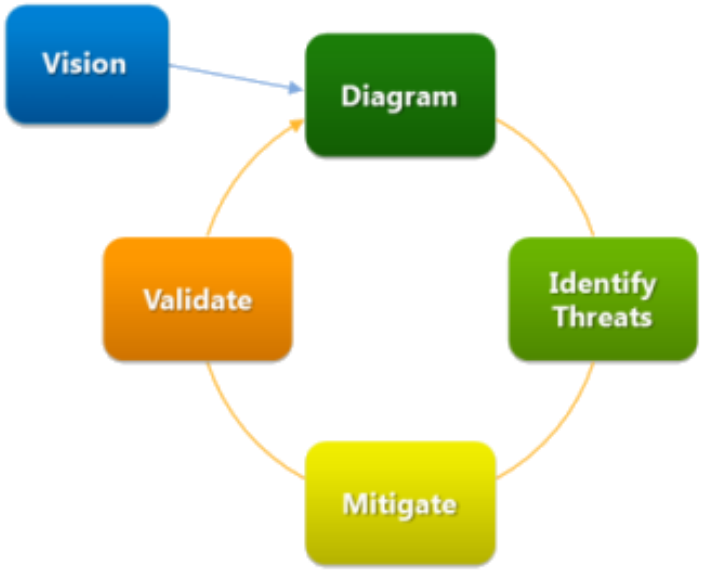



**Figure 11: SDL threat modelling process**

# IV. Applying Microsoft SDL Threat Model on HACKMI2

In this chapter, we apply the Microsoft SDL threat model tool to Elgg framework. This process shall be done step by step. The version of the threat model being applied is V3.1.8.

## 4.1 Data Flow Diagrams

The first step when applying the SDL threat model tool is to create the data flow diagram using elements described in section 2.1.1. The IDE used embeds Microsoft Visio, so elements can be added through drag and drop.

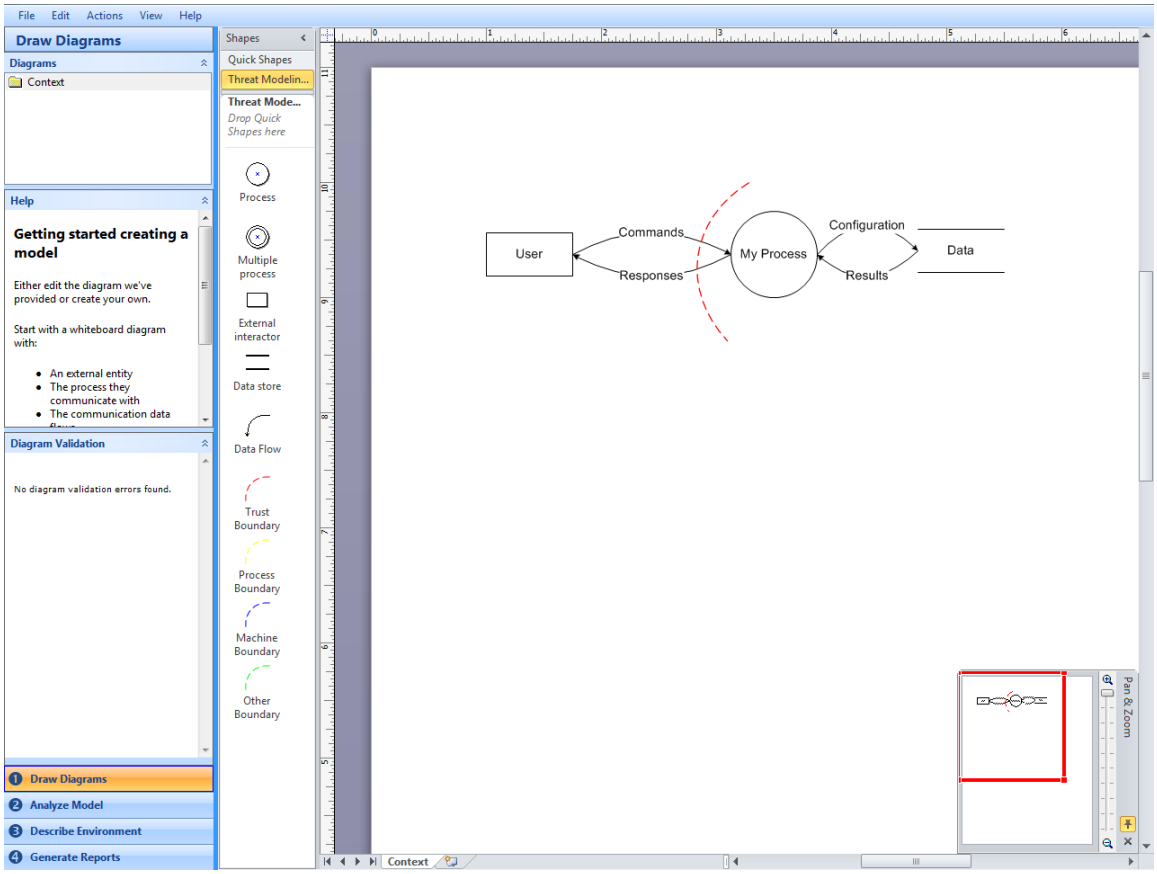



**Figure 12: SDL DFD screen**

The diagram used here is a context(level 0) DFD, i.e. the diagram only documents the system's boundaries by highlighting its sources and destinations, as opposed to level 1 DFD that shows the system's primary processes, data stores, sources and destinations linked by data flows. The diagram below (figure 13) shows a DFD for HACKMI2 social network followed by the explanation of the diagram components or elements.

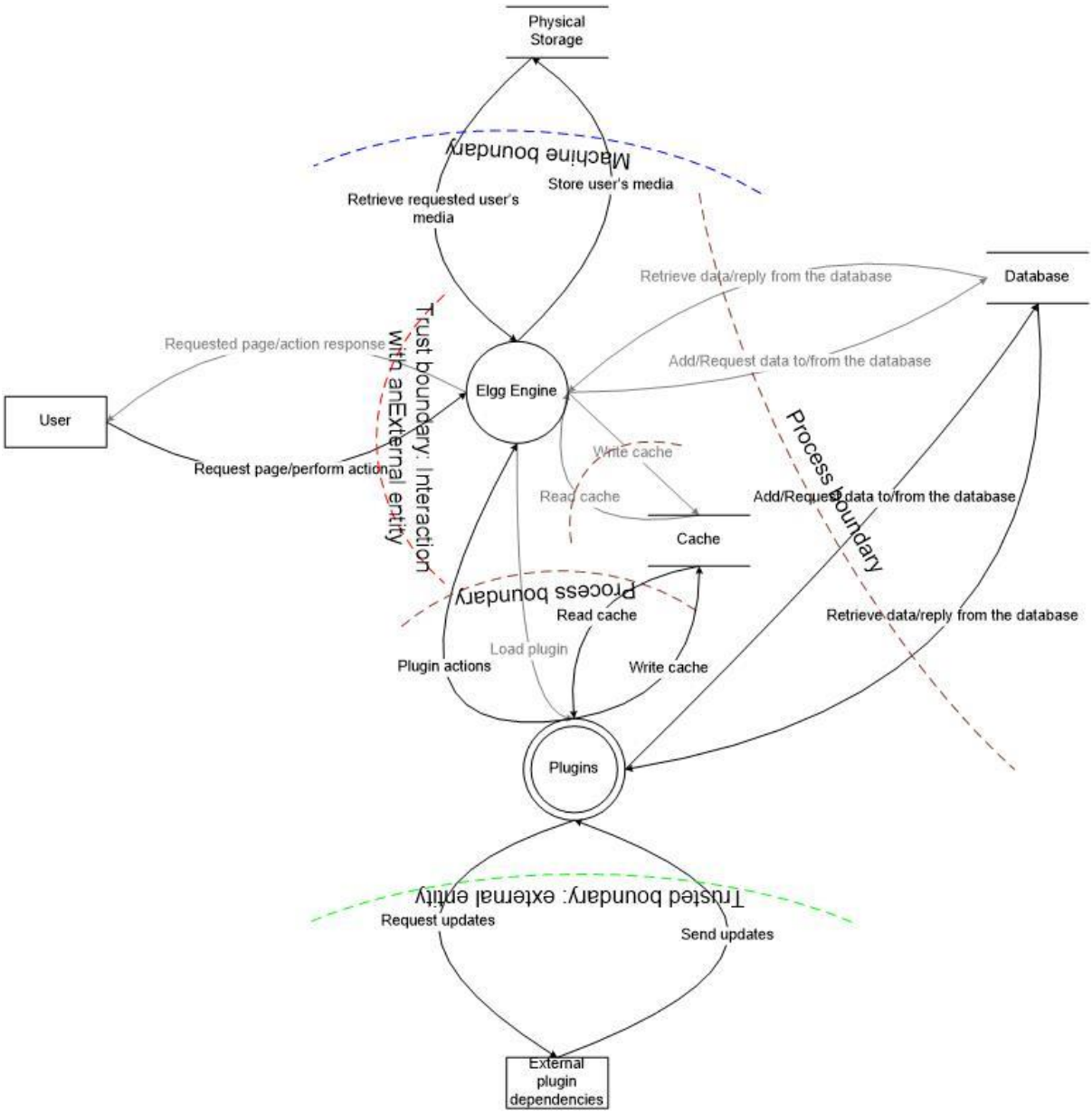



**Figure 13: Data flow diagram for HACKMI2 framework**

The DFD for HACKMI2 has two external interactors, namely a user and external plugin dependencies. These two interactors are the main source of threats to the system. Any attack to the system is initiated from one of these interactors. The user can send data through the data flow from the user element to the Elgg Engine.

There are limitless actions or request that a user can perform or request to/from the database. However the Elgg decides whether to allow or not to allow certain actions performed by a user. Hence having some rules imparted to the Elgg engine is of great importance to the security of the system. The Elgg engine has no control over some of the harmful actions, for instance those initiated from the plugins. This is because Elgg has put trust on the administrator of the system, that they will install trusted plugins. As explained in section 2.2, Elgg plugins are used to extend Elgg functionalities. Some of these plugins have external sources of updates, as for example, "*Message Board 1.8*" plugin accesses the site "*http://www.eastrolog.com/free-daily-horoscopes/taurus-horoscope-today.php*" to get updates for its daily horoscopes displays. The problem arises when the plugin external dependency sends back

malicious codes to be executed by the plugin. Since the Elgg engine trusts the plugin due to the trust granted to the administrator, all actions performed by the plugin will be accepted. The only interactor the Elgg engine has not given full trust to, are the users. The database uses MySQL, and it stores user's personal information and user's entities. Physical storage is where the user's uploaded media is stored. The size of the physical storage used is in hosting our social network 3 terabytes. The Cache refers is the normal cache used on web browsers and all the data flows represent actions.

## 4.2 Analysing the DFD Diagram

The SDL threat modelling tool will analyse the DFD created in step 1 and produce a list of threats. This is where we analyse the threat, and entering threat analysis, impact estimation and how we plan on addressing these threats. The figure 14 below is the analysis model main screen with automatically generated threat instances and selected progress information.

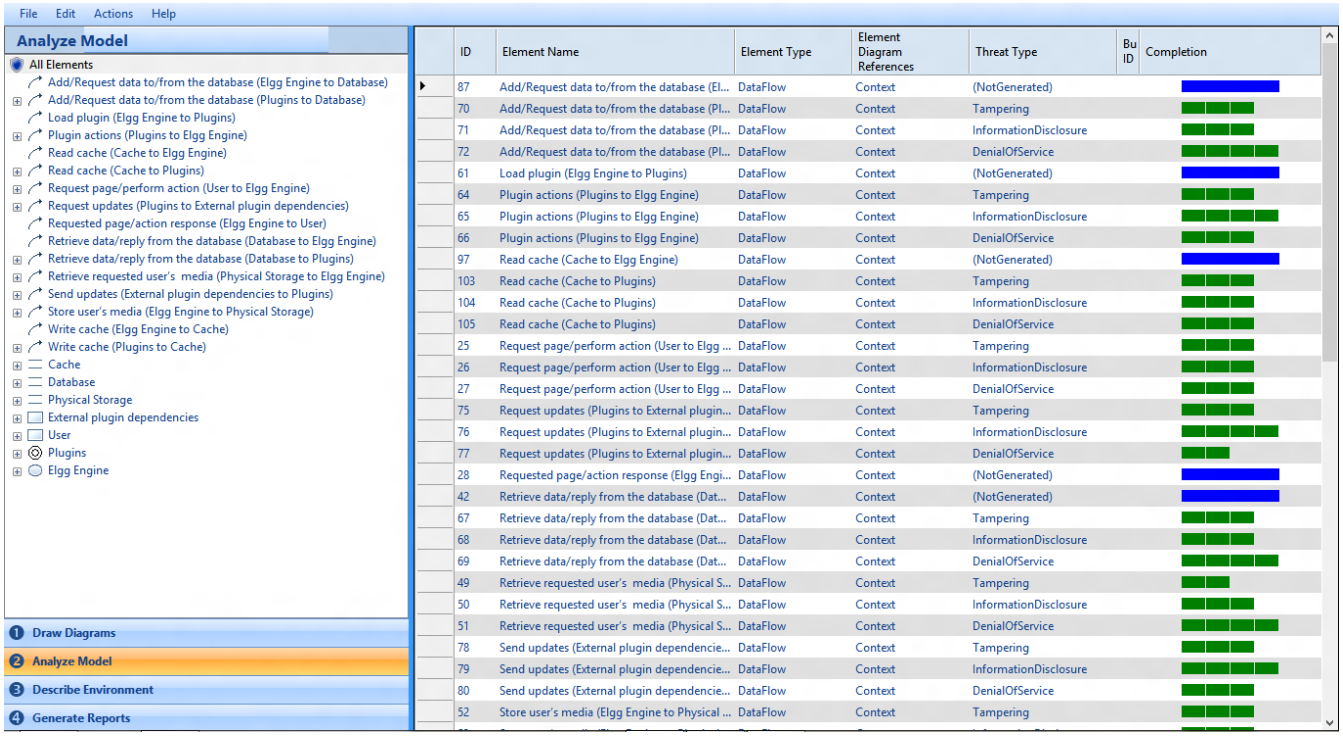



**Figure 14: SDL threat model analysis main screen**

The main display area contains a list of all the threats found in the DFD diagram, each with an ID number. The HACKMI2 DFD diagram generated more 100 threats. Each threat is grouped according to the elements they are associated with. For each type, threats are identified according to STRIDE type.

By clicking on a threat, we get a display analysis screen specific to that threat instance. The figure 15 below shows an analysis screen for threat with ID 81.

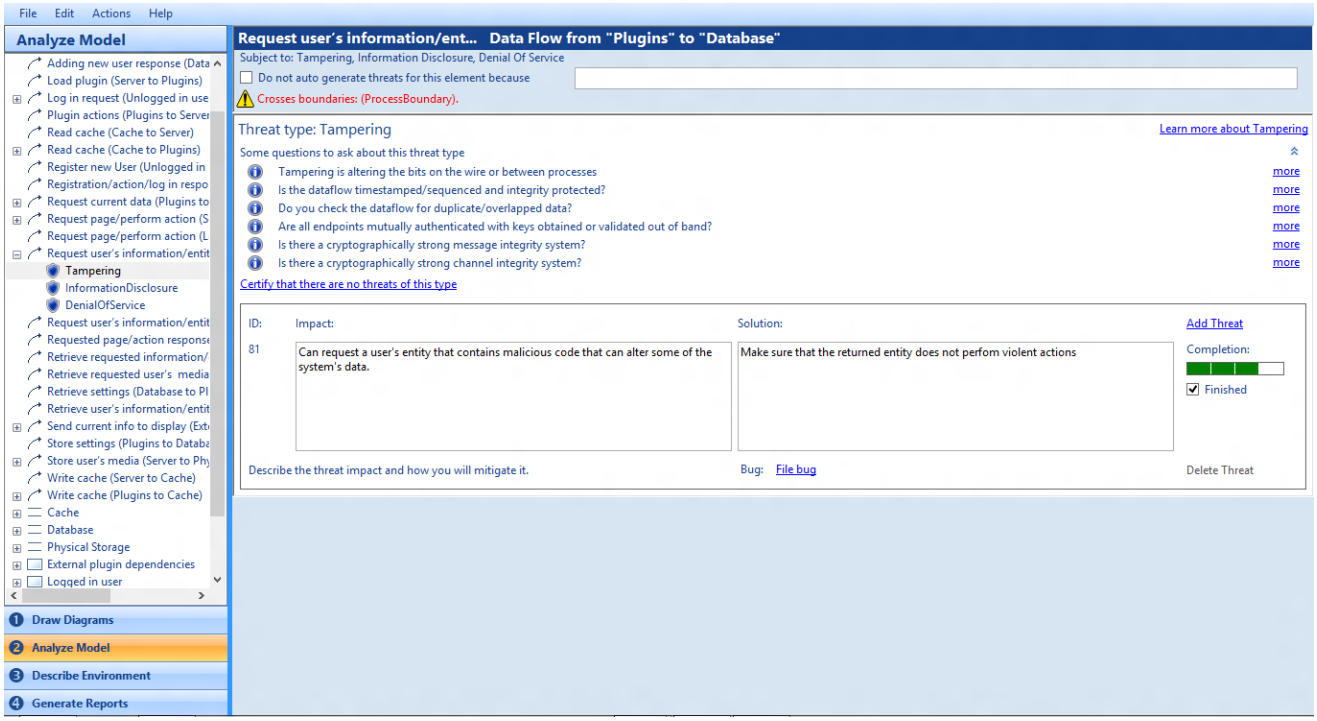



**Figure 15: Analysis screen for threat with ID 81**

The analysis screen has the following functions:

- Threat type. Example- Tampering.
- List of important questions to ask and answer to help in evaluating the impact of the threat and determine how to mitigate it.
- An entry field in which to describe the threat and impact of the threat.
- An entry field in which to describe the solution to the threat.
    - The SDL has four approaches to mitigation discussed in section 2.4.10 which are (in order of preference)
        - Redesign
        - 'Standard' mitigations such Access control lists.
        - Unique mitigation or
        - Accept risk in accordance with policies.

      HACKMI2 is an already finished product, and redesigning it won't be the mitigation approach that will be applied to HACKMI2 because that would require more time than the given duration. Two mitigation approaches that will be used are the 'Standard' mitigation and Accepting risk in accordance with policies. The 'standard' mitigation procedures used were found from[18].

- Provides a file bug feature to enter the threat as a bug in the back tracking system. Microsoft does not allow non-Microsoft users to submit bugs. This feature does not affect the threat modelling process, but for Microsoft to try and find bugs on their product.
- A checkbox that allows one to note when done with the descriptions.
- A completion bar, tracking progress through the four stages: threat, mitigation, bug filed and finished.

For each threat in the analysis, we respond in the following manner

a) For each threat we consider to be genuine, we enter an impact estimate and our proposed mitigation in the following text entry field provided, with heading solution.
b) For each threat, we submit a bug as appropriate. This won't be applied to our DFD.
c) Where appropriate, we certify that the threat does not apply using the button so labelled. Reasons why one certify a threat includes:
    - One is confident that the threat is not applicable.
    - Threat is within a process boundary: Data flows within a process boundary are rarely vulnerable.
    - The threat is mitigated elsewhere.
    - The threat is an acceptable risk as defined by the applicable standards.
d) For elements that we do not want threats to be generated and marked them in our diagram; they will be grayed in the DFD diagram, there will only be one marked "informational". Clicking on the entry will display the screen displayed in the figure 16 below;

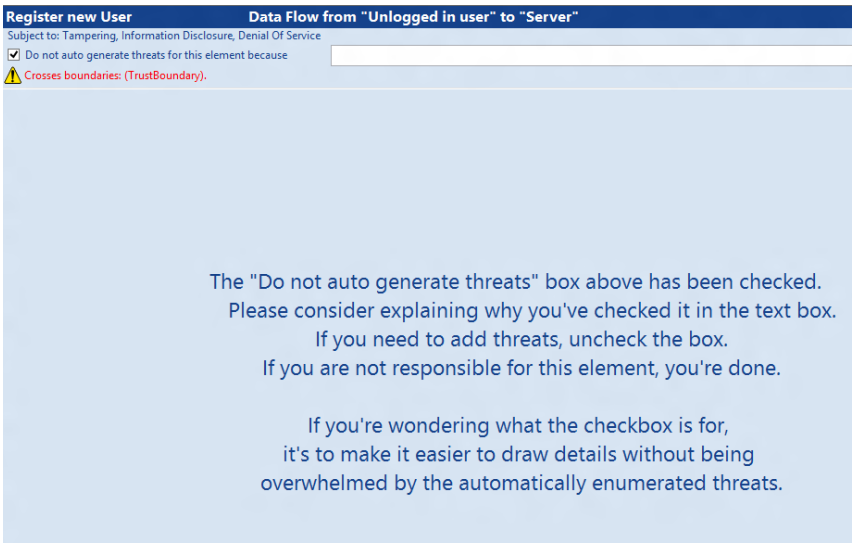



**Figure 16: Informational elements**

## 4.3 Describe Environment

This step consists of describing the environment in which our software will be deployed. This information has no effect on the tool's analysis of our design. This is just the additional information that will be captured in the report that we shall generate very soon. The Describe Environment screen is shown in the figure 17.

The environment description contains the following fields:

- **Dependencies**
  Libraries that will be linked to as part of the build process, especially third-party code or other code that may not have been through security verification must be entered here.
- **Assumptions**
  Enter assumptions as they come up in the work or discussions. As an example, there can be an assumption that certain actions can only be performed by the administrator.
- **External Security Notes**
  Any security related information that users of the product should know.

- **Document Header Information**
  This includes useful header information that will appear at the top of all generated reports.

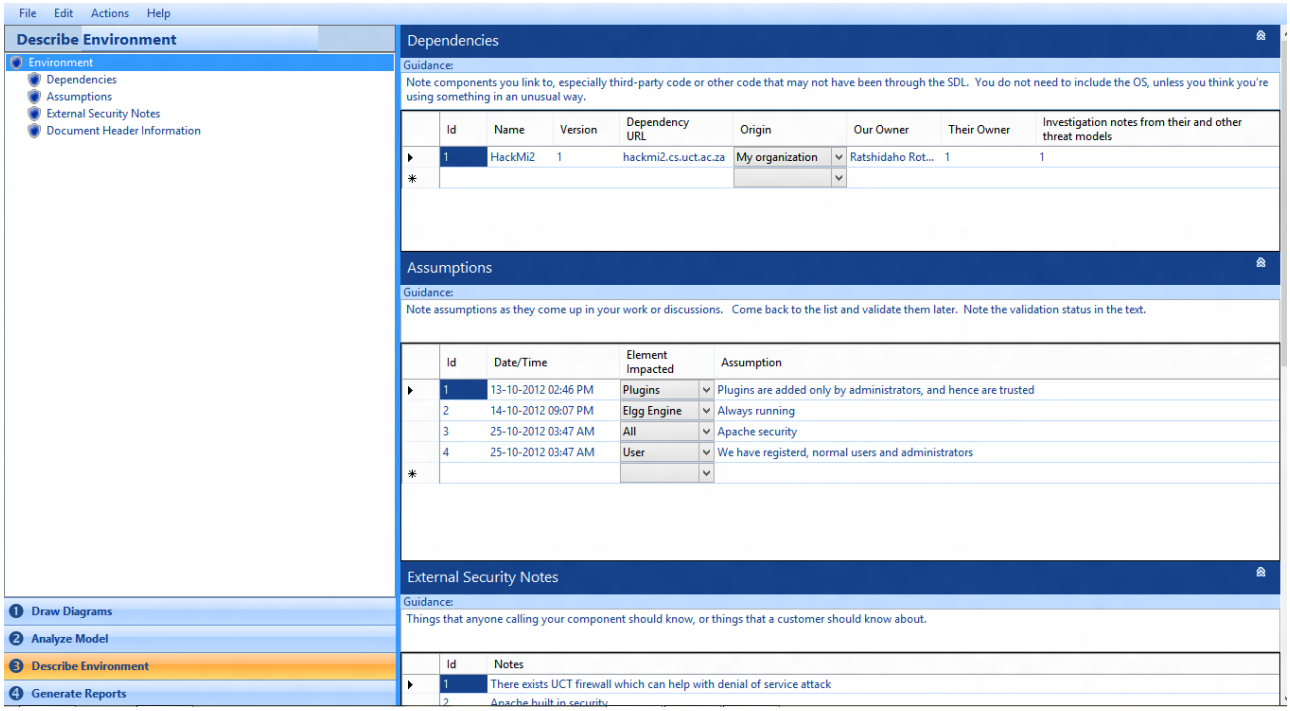



**Figure 17: Describe Environment Screen**

The next step involves generating the report of the threat model after being applied to the HACKMI2. The report generated forms an import discussion for this report hence it will be discussed as a chapter.

# V. Generating reports

The generate report steps generates and displays reports of selected types. Figure 18 shows the Generate Report Windows. The Generate report gives us five different types of reports that we can generate.

## 5.1 Bug report

A list of bugs that have been entered against threats discovered in the analysis.

But report won't be generated because the SDL threat modelling tool does not allow external (non-Microsoft) users to submit bugs.


**Figure 18: SDL Generate Reports Screen**

## 5.2 Analysis Report (Appendix 5A and 5B)

This report shows the status of various elements, threats and certifications identified in the model. These have been grouped as follows:

- **Errors**
  This contains elements in the model that have failed the validation.
  This report shows that all elements that have been used passed the validation test.

- **Empty Threats**

These are Threats which have had no information entered during the Analyse Model stage.
For the threat model created, all threats have been considered.

- **Unmitigated Threats**
  Threats which have had no mitigations entered during the Analyse Model stage.
  For each threat, there was a proposed solution. This category is empty.

- **Certifications**
  The Certification such as "not a threat" that have been made, identified by the alias of the user who attested user alias.
  This category contains a list of all threats that were regarded as not a threat for reasons provided in the description field. Some of these threats are just not feasible given the location of the element that is exposed to the vulnerability, and other threats will have little or no impact at all.
- **Informational elements/ Non-Autogenerated Elements**
  Elements that are entered for context only and for which you do not want threats generated automatically.
  Some of the reason these elements have no threats generated might be because the element only interact with trusted source and destination, making it trusted to, or there exists another elements with similar threats that already has the threats mitigated.

## 5.3 Threat Model Report (Appendixes 6A-6I)

The Threat modelling report is a comprehensive report containing all of the information captured during the threat modelling.

The first information that the threat model provides is the "Threat Model Information" which returns information such as a summary of the threat model, the owner of the threat model, participants and other generic information.

The next section on the Threat model report is the data flow diagram. This is the same data flow diagram that was drawn in the data flow diagramming of the SDL threat model.

Threats and mitigations follow after the data flow diagram section. This section contains a table that have a list all elements used in the data flow diagram, and the description for each element, if an element was marked to have no threats generated, the reason for not having threats generated is provided in the description field as well.

All elements in the data flow diagram are then group according to their element type, i.e. External entities, Processes, Multi-Processes, Data flows and Data Stores. Each of these elements has the threats it is exposed and each threat is followed by its Id. Each threat is followed by a mitigation procedure.

Certification section has the same description as the one described in section 4.2.

The last 3 sections are the external dependencies, implementations and Assumptions and external security notes are defined in section 3.3.

## 5.4 Threats List Report (Appendixes 7A-7G)

The Threat list report shows elements per threat. I.e. Each element with all the threats associated with it. See Appendix 7A-7G.

# 5.5 Recommended Fuzzing (Appendix 8)

This report provides a prioritised list of recommended targets for fuzz testing. Fuzzy testing is the feeding of arbitrary data into the program inputs[18]. The recommended fuzzing report is shown in Appendix 8.

The SDL threat model tool puts the data flow from external an entity that crosses data boundaries as priority 1. This is because external entity is the main source of threats, or the initiators of actions that might impose threats to the social network. For the social network HACKMI2, 4 elements crossing boundaries from the external entities were place in the priority 1 list, which are:

- Actions/requests from users
- Updates received from the external Elgg entities

Priority 2 consists of the data flows not from external entities that cross a boundary or boundaries. If a priority 1 threat has successfully bypassed all the mitigations, it will also have to bypass priority 2 mitigation procedures. This is because a threat needs a data path for transversal, and the data flow is the only element that provides means of locomotion from one element to the other. Priority 2 of HACKMI2 social network consists of the following data flow elements or actions:

- Retrieving user's media from the physical storage
- When storing user's media to the physical storage.
- When a plugin write a cache.
- When plugins perform any actions
- When adding to requesting data from/to the database.
- Data retrieved from the database.
- When the plugin reads a cache.

# VI. Attacks

This chapter gives an insight on some attacks or threats before the following chapters describe how the attacks were carried.

## 6.1 Cross-site Scripting (XSS)

Cross-Site Scripting attacks are a type of injection problem, in which malicious scripts are injected into the otherwise benign and trusted web sites[22]. These attacks occur when an attacker uses a web application to send malicious code, generally in a form of a client/browser side scripts, to a different end user. Cross-site scripting attacks occur anywhere in a web application uses input from a user in the output it generates without validating or encoding it.

An attacker can use XSS to send malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because the browsers thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by your browser and used with that site. This can do almost anything, including rewriting contents of the HTML page.

XSS attacks can generally be categorised into two categories: stored and reflected.

### 6.1.1 Stored XSS attacks

Stored attacks are those attacks where the injected code is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information.

### 6.1.2 Reflected XSS attacks

Reflected attacks are those attacks where the injected code is reflected off the web server, such as an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as an e-mail message, or on some other web server. When a user is tricked into clicking on a malicious link or submitting a specially crafted form, the injected code travels to the vulnerable web server, which reflects the attack back to the user's browser. The browser then executes the code because it came from a trusted server.

## 6.2 Session hijacking

The session hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token.

Because http communication uses many different TCP connections, the web server needs a method to recognise every user's connections. The most useful method depends on a token that the Web Server sends to the client browser after a successful client authentication. A session token is normally composed of a string of variable width and it could be used in different ways, like the URL, in the header of the http requisition as a cookie, in other parts of the header of the http request, or yet in the body of the http requisition.

The session hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorised access to the web server.

There are many ways to compromise a session cookie, and the most common ones are:

- Predictable session token.
- Session Sniffing
- Client-side attacks (XSS, malicious Java Script codes, etc.)
- Man-in-the-middle attack- Interception of communication between two systems.
  The attacker splits the original TCP connection into two new connections, one between the client and the attacker, and the other between the attacker and the server as shown the figure below (figure 19).

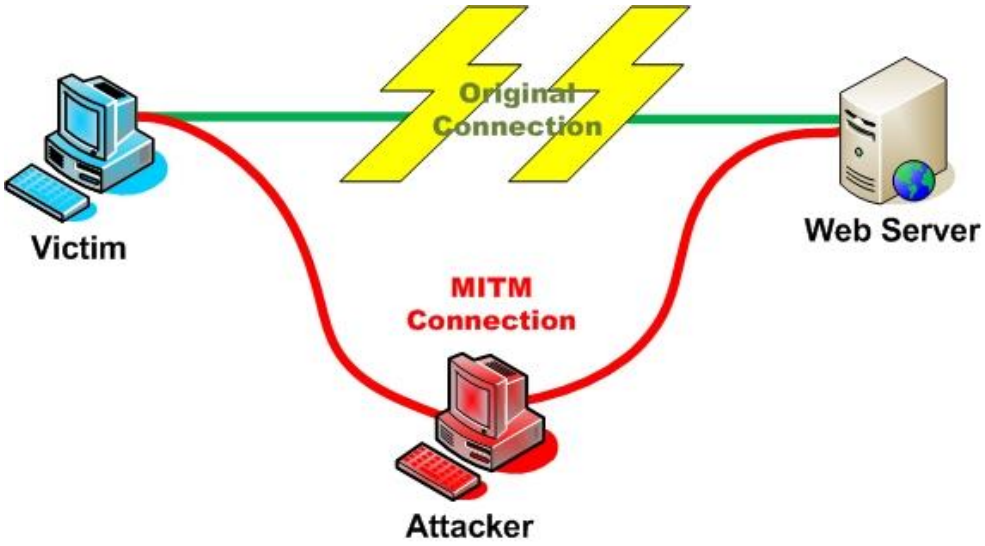



**Figure 19: Illustration of the man in the middle attack**

- Man-in-the-browser attack
  This attack has the same approach as the Man-in-the-middle attack, but in this case, a Trojan horse is used to intercept and manipulate calls between the main application's executable and its security mechanisms or libraries on-the-fly. A Trojan horse is simply a program that uses malicious code masqueraded as a trusted application[22].

## 6.3 Session fixation

Session fixation is an attack that permits an attacker to hijack a valid user session. The attack explores a limitation in the way the web application manages the session ID, more specifically the vulnerable web application. When authenticating a user, it doesn't assign a new session ID, making it possible to use an existent session ID. The attack consists of inducing a user to authenticate himself with a known session, and then hijacking the user-validated session by the knowledge of the used session ID. The attacker has to provide a legitimate web application session ID and try to make the victim's browser use it.

The session fixation is a class of Session Hijacking, which steals the established session between the client and the web server after the user logs in. Instead, the session fixation attack fixes an established session on the victim's browser, so the attack starts before the user logs in.

There are several techniques to execute this attack, depending on how the Web application deals with session tokens. Some of the most common techniques are described below:

- **Session token in the URL argument:** The session ID is sent to the victim in a hyperlink and the victim accesses the site through the malicious URL.
- **Session token in a hidden form field:** The victim must be tricked to authenticate in the target Web Server, using a login form developed for the attacker. The form could be hosted in the evil web server or directly in html formatted e-mail.
- **Session ID in a cookie**
    - **Client-side script**
      Most browsers support the execution of client-side scripting. This is the case where the aggressor could use attacks of code injection as the XSS (cross-site scripting) attack to insert a malicious code in the hyperlink sent to the victim and fix a Session ID in its cookie. Using the function document.cookie, the browser which executes the command becomes capable of fixing values inside of the cookie that it will use to keep a session between the client and the Web Application.
    - **<META> tag**
      <META> tag is also considered a code injection attack. However, different from XSS attack where undesirable scripts can be disabled, or the execution can be denied. The attack using this method becomes much more efficient because it is impossible to disable the processing of these tags in the browsers.
    - **HTTP header response**
      This method explores the server's response to fix Session ID in the victim's browser. Include the parameter Set-Cookie in the HTTP header response, the attacker is able to insert the value of Session ID in the cookie and sends it to the victim's browser.

## 6.4 Account lockout attack

Account lockout attack is when the attacker attempts to lock out all users account, typically by failing login more times than the threshold defined by the authentication system. For example, if the users are locked out of their accounts after three failed login attempts, an attacker can lock out their account for them simply by failing login three times. This attack can result in a large scale denial of service attack if all user accounts are locked out, especially if the amount of work or time required to rest the accounts is significant.

## 6.5 Phishing

Phishing is misrepresentation where criminal uses social engineering to appear as a trusted identity. The attacker leverages the trust to gain valuable information such as bank account details or logging details.

There are many ways that phishing can be carried out. Some of them are discussed below:

- Delivery via website, e-mail or instant message, the attack asks a user to click on a link to re-validate or re-activate their account. The link displays a believable facsimile of the site and brand to con users to provide details.
- Sends a threating e-mail to users telling them that the user has attacked the sender. There's a link in the e-mail which asks users to provide personal details.
- Urgent messages that the user's account has been compromised and they need to take some sort of action to "clear it up".
- Messages from the "Security" section asking the victim to check their account as someone illegally accessed it on this date, just click this trusty link…

## 6.6 Brute force attack

A brute force attack is a particular strategy used to break a user's password. This is the most widely used method for cracking passwords and it involves running through all possible permutations of keys until the correct key if found[23]. If a users' password is 2 characters long and consists of letters and numbers, and is case sensitive, then a brute force attack would see a potential of 3844 different guesses for the given password. The longer the password, the more guesses, hence more time needed for the brute force to run through all the possible permutations. Another similar attack that uses the words in the dictionary instead of using every possible combination of words is called the dictionary attack.

# VII. Attacks and Vulnerabilities on HACKMI2

## 7.1 Session Hijacking

A session hijacking attack was successful when applied to HACKMI2. This was accomplished through a both stored and reflected XSS attack. Although HACKMI2 have ways of checking if the session token has been compromised as described in section 8.1.1.2, this attack was able to be carried out successfully by sending the user's agent to the attacker as well.

### 7.1.1 Stored XSS Attack

This attack was accomplished through a blog comment. The attacker just opens a blog that someone has created, or he can create his own blog, and then injects a malicious code as a blog comment which will be reflected in the activity page. A user doesn't have to open the blog for the effects of the malicious code to take place, but as long as he/she views the activity page the code will begin being executing.

Elgg stores its session tokens in a cookie on a client-side browser. Attack through a malicious code takes advantage of this feature; it gets the cookie session and sends it to the attacker's server. The user won't notice anything because after the session token has been stored on the attacker's side, he/she is redirected back to HACKMI2 website where the session resumes normally.

### 7.1.2 Reflected XSS Attack

Same malicious code used in the stored XSS attack is used, but in this case it is used on the chat. The attacker sends a message to another user, and when the user clicks on the notification to read the message, the malicious code is executed, and the session token is sent to the attacker. The effect of this attack is less severe compared to the stored XSS attack since this one only affects one person at a time, and can only affect the people the attacker is a friend of, unless the attacker uses the global chat.

After the user gets hold of the other users' session token, he/she can go to the login page of HACKMI2, add the session token to the cookie and reload the page, after which the attacker now has hold of the users account.

The data flow of the attack as viewed from the data flow diagram is: A logged in user (Attacker) performs the action to the server (adding a comment), the server loads the blog plugin, and the blog plugin responds with an action to the server to save the comment (Entity) to the database.

When another user logs in, the server loads entities (blogs and the related comments), and the malicious code will be part of these entities being loaded and will be executed on the client-side web browser.

### 7.1.3 Impact of the attack

The attacker gets hold of the user's account; hence he can read all of the user's private information, including the private messages, change user's name and other personal details.

The impact gets more severe if the hijacked session token is for the administrator. In such a case, the attacker can change the administrators' password, and delete other administrators' accounts living him in control of the entire social network. If the system does not have any back-up, then it is the end of it and it has to be recreated.

### 7.1.4 Solution

Since this attack relies on the applications vulnerability to XSS, having the application immune to XSS would solve the problem. The solution implemented on HACKMI2 was to sanitise user's inputs before it was stored in the database. Take for instance "*<script>alert('hello');</script>*" as the code that the user wants to store in the database. The server will sanitise this input and store it as *<script>alert('hello');</script>*

What the sanitisation does is to change HTML-escapes special characters to other form of representation that browsers can interpret. For example "<" was represented as "*<"* in this case, and this is what will be stored in the database. When a browsers comes across this token, it will display it as a "<".

## 7.2 Account lockout attack

Sanele views the system as if he is the administrator, as opposed to me who views the system as a hacker. To increase the security strength of the system, Sanele decided to define the following rules when login:

*A user is given three login tries, after which the account will be locked for 5 minutes. After 5 minutes the account will be unblocked and the user will be given three more tries again. If the user continues to fail login, the account is blocked two times longer than the previous minutes.*

```
If login fails
      NumberOftries++
      If(NumOfTries%3==0)
            Block the user for NumberOfTries/3 * (NumberOfTries%3)
Else
      numberOfTries=0;
      login suceeful
```

**Figure 20: Pseudocode for the fail login try.**

The following section describes how this rule can be used to provoke a denial of service attack on the system. The diagram below (figure 21) shows the relationship between the number of time a user is blocked, without any successful login in between, and the time to wait before the account is unblocked.

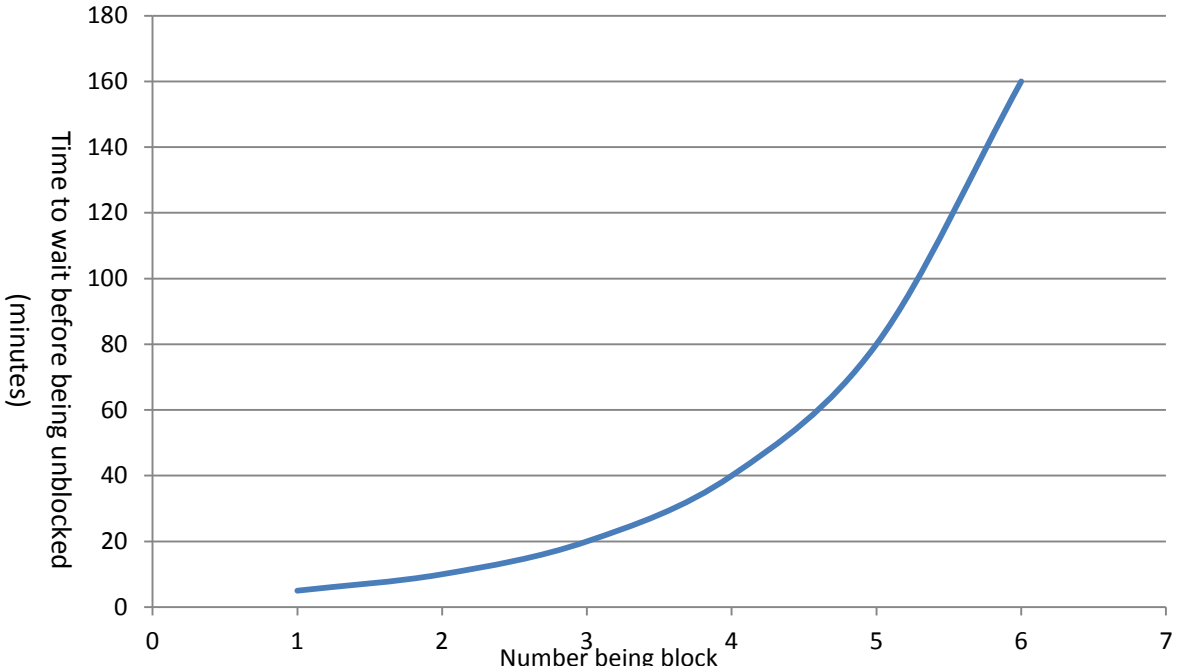



**Figure 21: Relationship between failed login attempts and the time to wait to be unblocked.**

An attacker can write a script that continually feeds-in a wrong password for a username, until that user's account is blocked. As the graph in figure 21 shows, there is no limit in the amount of time that a user can be locked out. Worst case scenario would happen if the user being locked out is the administrator. In that case, the system would not have an administrator. This is one of the attacks that the Microsoft SDL threat model classifies as priority 1 threats in the recommended fuzzing report (section 4.5).

### 7.2.1 Solution

The advantage of the rule given above is that it protects the system against brute force attack, but put the system vulnerable to account lockout attack as described above.

A better solution against both Account lockout and the brute force attack is to use *"**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part"* (CAPTCHA).

A CAPTCHA is a type of challenge-response test used in computing as an attempt to ensure that the response is generated by a person[24]. Figure 22 shows an example of a CAPTCHA. A user is asked to enter the letters or digits from distorted image. CAPTCHA's are not only limited to alphanumeric characters, they can also use pictures or quizzes instead of alphanumeric.

**Figure 22: Example of a letter CAPTCHA (source: wikipedia)**

Every time a username fails to login three times, instead of being blocked, the system can activate a Captcha for that username before login attempts.

## 7.3 Phishing users

A phishing attack was successfully carried out by taking advantage of the XSS vulnerability to the system. A hacker takes the HACKMI2 logging page source code and change the login 'form action' to a script in his server where he will be able to extract users' login details. The hacker then injects a java script as a comment to the blog that redirects people to his fake HACKMI2 login page. When a normal user sees the comment that the malicious user has made, the script is automatically executed and the user will be redirected to the fake login page. Most users' won't really know what happened, but will think that they have been logged out and try to log in again by entering their login details. When the user clicks login, the users' login details is sent via POST or GET to the hackers server where he/she will save this details and redirects the user back to the real HACKMI2 site. Since the user was already logged in, he/she will be directed to the activity page his/her session will continue normally.

### 7.3.1 Impact

The impact is almost the same as that of the session hijack attack, but the attacker has the users' login details and can log in to the social network any time he/she wants. If the details compromised are for an administrator, then the entire system is on the attackers hands.

### 7.3.2 Solution

A patch to get the social network invulnerable to XSS attack solved this problem, i.e. filtering every input received from the user.

Educating people about such attacks can reduce the number of people being victims of phishing. Users can be educated to first check the page URL before entering their login details.

HACKMI2 has a report button that gives users a way to report any abuse, be it malicious comments or over-flooding the activity page to the administrators. User can be encouraged to use the report button if there is a need to.

Blocking URL's can help prevent the phishing attacks, this has the drawback of reducing the flexibility of the social network.

## 7.4 Spammers

Within a week of HACKMI2 operating, the social network already had fake accounts created that just spammed the activity page by blogging random things. Such accounts are called spammers. This was a problem because a user could miss out when other normal users blog, because their notification on the activity blog will go to the bottom of the page.

Spammers were able to register to the social network because there was no mechanism of checking whether a given machine is being operated by a human being or not.

### 7.4.1 Solution of Spammer

The solution to these was to make use of the CAPTCHA feature described in section 7.2.1. When a user registers, he/she is provided with a number of pictures and is asked to choose a specific item from the pictures provided as shown in figure 23 below, the user is asked to choose a folder. If a user chooses the wrong item or nothing, then the registration fails.

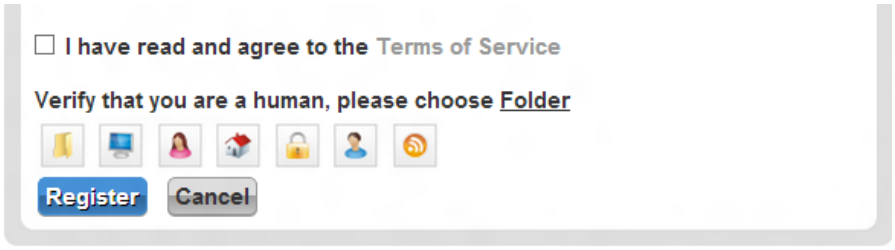



**Figure 23: HACKMI2 Captcha when registering**

# VIII. Unsuccessful Attacks on HACKMI2

The initial plan for the project was concentrate on top four of the ten most critical web application security risks described in the OWASP Top 10-2010 document. This document was meant to focus on 'Broken Authentication and Session management' and 'Insecure Direct Object References'.

The Elgg framework v1.0 was released in August 2008. Ever since then, it has continually been updated. Some of this updates were security patches against the vulnerabilities that this paper was supposed to focus on.

## 8.1 Broken Authentication and Session management

### 8.1.1 Sessions

Elgg uses PHP's session handling with custom handlers. Session data is stored in the database. The session cookie contains the id that links the user to the browser. The user's metadata is stored in the session including GUID[12], username, and email address. The session's lifetime is controlled through the server's PHP configuration. Sessions end automatically when the browser is closed.

#### 8.1.1.1 Session fixation

Elgg protects the system against session fixation by regenerating the session id when a user logs in. Thereby making sure that even if the attacker gets hold of a user's old session ID, he/she wouldn't do anything with it.

#### 8.1.1.2 Session hijacking

Besides protecting against session fixation attacks, Elgg also has a further check to try to defeat session hijacking if the session identifier is compromised. Elgg stores a hash of the browser's user agent and a site secret as a session fingerprint. The use of the site secret is rather superfluous but checking the user agent might prevent some session hijacking attempts. This however does not completely prevent session hijacking since the attacker can get the user's agent by using JQuery commands.

## 8.2 Insecure Direct Object References

This occurs when a developer exposes a reference to an internal implementation object such as file, directory or database key. Without access control check or other protection, attackers can manipulate these references to access unauthorised data.

Elgg restrict users from accessing certain path or directory. This was accomplished by having a method called gatekeeper(), which whenever called, it checks if the user has the rights to have access to the directory or file.

[12] Globally unique identifier

# IX. Comparison of the three threat modelling tools

This chapter focuses on the comparison between the three threat modelling tools that have been used in this project to evaluate the security of the HACKMI2 social network; The Microsoft threat and analysis, SDL and SensePost CTM threat model. The first section discussion the advantages and disadvantages of the Microsoft SDL threat model.

## 9.1 Microsoft SDL threat modelling tool

### 9.1.1 Advantages

The SDL threat modelling 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.

Microsoft SDL threat modelling tool is designed in such a way that even people who are not security experts can use it. Microsoft SDL threat modelling 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 modelling tool is centred on the software, as opposed to other threat modelling 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.

Reports generated by the Microsoft SDL threat modelling tool can easily be imported to Microsoft one note; further more documentation on the usage of Microsoft SDL tool is available online for free.

## 9.2 Disadvantages

Microsoft SDL threat modelling 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 modelling 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 prioritising 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.

It was not an easy task to get started with the Microsoft SDL threat modelling tool without having received any formal training. Describing use case scenarios and determining threat types requires engineers or security experts. Coming up with use case scenarios and determining threat types took a great amount of time of reading different security articles.

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 modelled, 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 modelling using context (level 0) DFD. Appendix 9 shows an example of mixed level 1 and 0 DFDs. It 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, hence threat. 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 recognise 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 Appendix 9, 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.

Appendix 9 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.

## 9.2 Microsoft Threat Analysis and Modelling 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 & Modelling 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 & Modelling 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 & Modelling tool automatically generates potential threats to your software application, based solely on known information that you provide. The Threat Analysis &

Modelling tool also has the capability to assimilate the information you provide to build security artefacts 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

The following section is a summarised advantages and disadvantage of TAM.

### 9.2.1 Advantage

The Microsoft Threat Analysis and Modelling (TAM) tool starts with the use cases, defining roles and types of data and it gives great details (to the point of CRUD[13] operations) to the application architecture.

The Threat Analysis and Modelling Tool allow one to rate threats and define the technology used in the application architecture. The images generated by the tool are a bit fancier through the use of colouring.

Auto-generation of threats is one of the major benefits for using the Microsoft Threat Analysis and Modelling tool. This threat modelling tool also has a nice view of representing threats using threat trees, attack surface or system call flaws. The documentation on the usage of TAM is available online for free.

### 9.2.2 Disadvantage

The downside of this tool is that the generated images (threat tree, attack surface and system call flaws) are not easily adaptable and, with a bit more complexity in the application, tend to get confusing. Besides this, the automatic generation of threats highlights a lot of threats, but most are not useful in the context of a social network. The detail level is sometimes too complex the simple system.

## 9.3 SensePost (CTM) Threat modelling tool

### 9.3.1 Advantages

The SensePost threat modelling tool allows the user to rank threats according to the impact they will have on the system and the likely hood of the vulnerability to be taken advantage. This enables the CTM tool to easily and effectively prioritise threats.

The user has to define threats in CTM. This is advantageous because the threat modelling process will only focus on threats that are applicable to the system being modelled.

The SensePost CTM allows the user to define the technology used in the system being modelled, which plays an important role in rating threats. For example, suppose there are computer machines say X and Y, operating independently having 10TB and 50 GB of hard drive space respectively. The

[13] CRUD – **C**reate **R**ead **U**pdate and **D**elete

DoS through memory consumption would have less impact on machine X than in machine Y. The CTM tool would rate this threat differently, higher priority for machine Y than in X. But with Microsoft SDL tool, this threat would have the same level of impact on both machines.

### 9.3.2 Disadvantages

Sometimes the security expert can forget to consider some of the threats that are applicable to the system, living the system vulnerable to threats that have been left out.

There limited or little publication on the process of threat modelling using the SensePost CTM. This gives the user hard time when using SensePost CTM because there is no reference available.

The prioritising of threats depends on the ratings given by the user which are subject to errors. Sometimes a user can mistakenly give a high/low rate to a threat that is supposed to be low rated, leading to the results with faulty priorities of threats.

This report focuses on applying the threat model using the attack centric approach. As discussed in section 3.6, the Attack-Centric approach focuses on identifying all the possible access points to the system and possible adversary goals. The TAM and SensePost CTM models do not clearly points out the entry points or how data flows from one element to the other, which is the basis of the Attack-Centric approach. Hence the TAM and SensePost CTM are not suitable for an Attack-Centric approach.

However, the Microsoft SDL threat model clearly shows the flow of data from one element to another, hence giving the attacker entry points of the system. This makes the SDL threat modelling tool suitable for an Attack-Centric approach to threat modelling.

# X. Conclusion and Future work

## 10.1 Conclusion

HACKMI2 is the experimental social network platform that was built from an open-source social network engine called Elgg. HACKMI2 was used to evaluate the effectiveness of three different threat modelling tools, namely Microsoft SDL threat modelling tool, Microsoft Threat and Analysis modelling tool (TAM) and SesnsePost Corporate Threat Modelling (CTM) tool. This was achieved by using these three threat modelling tools to evaluate the security of HACKMI2, and based on the results from each tool, a conclusion was made. Currently, HACKMI2 close to 200 registered members and has only been made available to the public 1 month ago.

The main aim of the project was to compare the three different threat modelling tools mentioned above. All threat modelling tools were used to evaluate the security of HACKMI2.

After getting the results from all these threat models, one can conclude that all three threat modelling tools have useful upsides, but quite annoying downsides. The conclusion one can make is that all these tools will be a good guidance and help with the steps one needs to take. All these tools can be used to have a more effective threat model, SDL threat model for the higher level overview, TAM to obtain countermeasures and generate wide variety of threats, and the CTM to obtain more detailed of the architecture and give more focus to only important threats.

The project proposal plan was to focus on the top four web application securities threats listed in the 'OWASP , The Ten Most Critical Web Application Security Risk'[25] document for evaluating the security of HACKMI2. Elgg frame has had recent updates which are patches against most of the threats listed the OWASP document, including the top four. This signifies that HACKMI2 was also invulnerable to those threats that Elgg has had security patches for. Then the focus of the threats were changed to be on any of those listed in the OWASP document, rather than just the top four. HACKMI2 was found to be vulnerable to "Cross Site Scripting (XSS)" and "Broken Authentication and Session Management", which are number 1 and 3 respectively on the list. Solution to XSS was to filter inputs given by the user. Patching XSS vulnerability also solves the problem of session management attack that was carried out.

## 10.2 Future work

We concluded that all the threat models that were used have their own ups and downs and that each of these threat models would suit be able for a different system. A good place to start with future research is to try and determine the type of system in which each of these threat models will be more effective. And also find a way to integrate the reports generated from all these threat models into one more effective report.

More threats on HACKMI2 can be studied, even those not listed on the OWASP document, thereby getting the social network more secured. Also, one can extend the research into other web applications, except social network since it was the basis for this project.

# References


[1] B. Sanz, C. Laorden, G. Alvarez, and P. G. Bringas, “A Threat Model Approach to Attacks and Countermeasures in On-line Social Networks.”

[2] INTECO, “Study on the Privacy of Personal Data and on the Security of Information in Social Networks,” pp. 1–143, 2009.

[3] Y. Wang and A. Kobsa, “Privacy in Online Social Networking at Workplace,” *2009 International Conference on Computational Science and Engineering*, pp. 975–978, 2009.

[4] E. Zheleva and L. Getoor, “To Join or Not to Join : The Illusion of Privacy in Social Networks with Mixed Public and Private User Profiles,” pp. 531–540, 2009.

[5] Y. X. B. Ng, A. KanKanhalli, “tudying users’ computer security behavior: A health belief perspective, Decision Support Systems,” vol. 46, no. 4, pp. 815–825, 2009.

[6] D. Jensen and J. Neville, “Data Mining in Social Networks,” 2002.

[7] M. Balduzzi, C. Platzer, T. Holz, E. Kirda, D. Balzarotti, and C. Kruegel, “Abusing Social Networks for Automated User Profiling.”

[8] E. Position and P. No, “Security Issues and Recommendations for Online Social Networks,” no. 1, 2007.

[9] Margaret Rouse, “spear phishing,” 2011. [Online]. Available: http://searchsecurity.techtarget.com/definition/spear-phishing.

[10] K. Hoffman, D. Zage, and C. Nita-Rotaru, “A survey of attack and defense techniques for reputation systems,” *ACM Computing Surveys*, vol. 42, no. 1, pp. 1–31, Dec. 2009.

[11] E. Org, “Elgg Data Model,” 2008.

[12] Elgg, “A powerful open source social networking engine,” 2010. [Online]. Available: http://elgg.org/powering.php.

[13] R. Harvey and S. S. Consultant, “Thoughts on the Microsoft SDL,” pp. 1–2.

[14] S. W. Ambler, “Introduction to Security Threat Modeling.” [Online]. Available: http://www.agilemodeling.com/artifacts/securityThreatModel.htm.

[15] D. P. Mirembe and M. Muyeba, “Threat Modeling Revisited: Improving Expressiveness of Attack,” *2008 Second UKSIM European Symposium on Computer Modeling and Simulation*, pp. 93–98, Sep. 2008.

[16] FaultTreeOrg, “Fault tree analysis tool.”

[17] J. P. McDermott, “Attack net penetration testing,” *Proceedings of the 2000 workshop on New security paradigms - NSPW ’00*, pp. 15–21, 2000.

[18] S. D. L. Threat and M. Tool, “SDL Threat Modeling Tool,” pp. 1–33, 2009.

[19] SpOonWiZaRd, “Types of Attacks,” 2007. [Online]. Available: http://www.go4expert.com/forums/showthread.php?t=7685.

[20] M. A. Security, “Information Disclosure,” 2012.

[21] A. Shostack, “Experiences Threat Modeling at Microsoft,” pp. 1–11.

[22] T. O. W. A. S. Project(OWASP), “OWASP attacks,” 2009. [Online]. Available: https://www.owasp.org/index.php/Trojan_Horse.

[23] P. Park, “Brute Force Attack,” 2009. [Online]. Available: http://www.hackosis.com/brute-force-attack/.

[24] W. Org, “CAPTCHA,” 2012. [Online]. Available: http://en.wikipedia.org/wiki/CAPTCHA.

[25] T. O. W. A. S. Project(OWASP), “The Ten Most Critocal Web Application Security Risk.”

## Appendix 1: HACKMI2 TERMS AND CONDITIONS

### Terms

# Hackmi2: Terms and Conditions

Hackmi2 gives no warranty and makes no representation as to the content of this site or its accuracy and accepts no liability or any errors or omissions in it. Hackmi2 does not warrant that your use of the site will not infringe third party rights. Hackmi2 does not warrant that use of this site or materials downloaded from it will not cause computer virus infection or other damage to property. It is a condition of use of the site and the materials in it that use is at the user's own risk. Neither Hackmi2 nor any of the site's editors or contributors shall be liable for any loss or damages suffered as a result of any use of the site, including but not limited to direct loss, consequential loss and loss of profits.

This site includes links to other sites on the World Wide Web. We are not responsible for such sites and cannot vouch for the suitability or accuracy of their content. You link to them at your own risk.

Hackmi2 may monitor communications on the site but is under no obligation to do so. You must not post any material onto the site which is personal, defamatory, obscene or blasphemous, which infringes third party rights or which could in any other way give rise to criminal or civil liability in any jurisdiction. We shall have no liability for any such material. We may in our absolute discretion remove any material if in our view it falls or might fall within the foregoing categories or is otherwise inappropriate. We shall own any material you send to the site or to us shall not be obliged to treat any such communication as confidential and may exploit any such communication and its contents in such ways as we see fit.

We may change these conditions of use from time to time. You will be bound by changes even if you do not re-visit this page to re-read this notice.

## Appendix 2: HACKMI2 PRIVACY POLICY

# Privacy

**Privacy Policy**

This privacy statement discloses the privacy practices for HACKMI2.

***Information Collection and Use***

HACKMI2 is the sole owner of the information collected on this site. We will not sell, share, or rent this information to others in ways different from what is disclosed in this statement. HACKMI2 collects information from our users at several different points on our Web site.

***Web Site Registration***

In order to use some features of this Web site, users must first complete the registration form. During registration, users are required to give their contact information (name and e-mail address). This information is used to contact users about the topics on our site for which they have expressed interest and to enable users to retrieve lost passwords.

***Cookies***

HACKMI2 uses cookies to remember if users have logged in while on our site. This allows web site users to avoid logging in more than once, thereby saving time. Users have the option of disabling or not accepting cookies by changing the preferences on their browsers. If users opt to disable cookies, they will still be able to use our Web site. However, they will not be able to use some functionality or post to the message boards. No personally identifiable information (e-mail address, name, etc.) is collected with the cookies that we set.

***Web Statistics***

We use IP addresses to analyze trends, administer the site, track user movement, and gather broad demographic information for aggregate use for reporting and sponsorship purposes. IP addresses are not linked to personally identifiable information.

***Links***

This Web site contains links to other sites. Please be aware that HACKMI2 does not claim any responsibility for the privacy practices of such other sites. We encourage our users to be aware when they leave our site and to read the privacy statements of each and every Web site that collects personally identifiable information. This privacy statement applies solely to information collected by this Web site.

***Security***

This Website is not fully protected and we encourage users not to post personal information or information that might lead to financial loss, this a research based social network and and the creators will not take any responsibility for security breaches.

***Updates***

We may also send the user site and service announcement updates. Members are not able to unsubscribe from service announcements that contain important information about the service. We communicate with users to provide requested services and to discuss issues relating to their accounts via e-mail or phone.

***Notification of Changes***

If we decide to change our privacy policy, we will post those changes on our home page so our users are always aware of what information we collect, how we use it, and under what circumstances, if any, we disclose it. If at any point we decide to use personally identifiable information in a manner different from that stated at the time it was collected, we will notify users by e-mail. Users will have a choice as to whether or not we use their information in this different manner. We will use information in accordance with the privacy policy under which the information was collected.

## Appendix 3: ABOUT PAGE

**About**

# HACKMI2: About

HACKMI2 is an experimental social network created by Computer Science Students at the University of Cape Town, South Africa for studying computer security in social networks.

This site is for research purposes only and not intended to be commercialized.

Creators : Molulaqhooa Maoyi, Sanele Macanda and Rotondwa Ratshidaho

For more information please email your query to hacki2@hotmail.com

# Appendix 4: Data Flow Diagram for HACKMI2

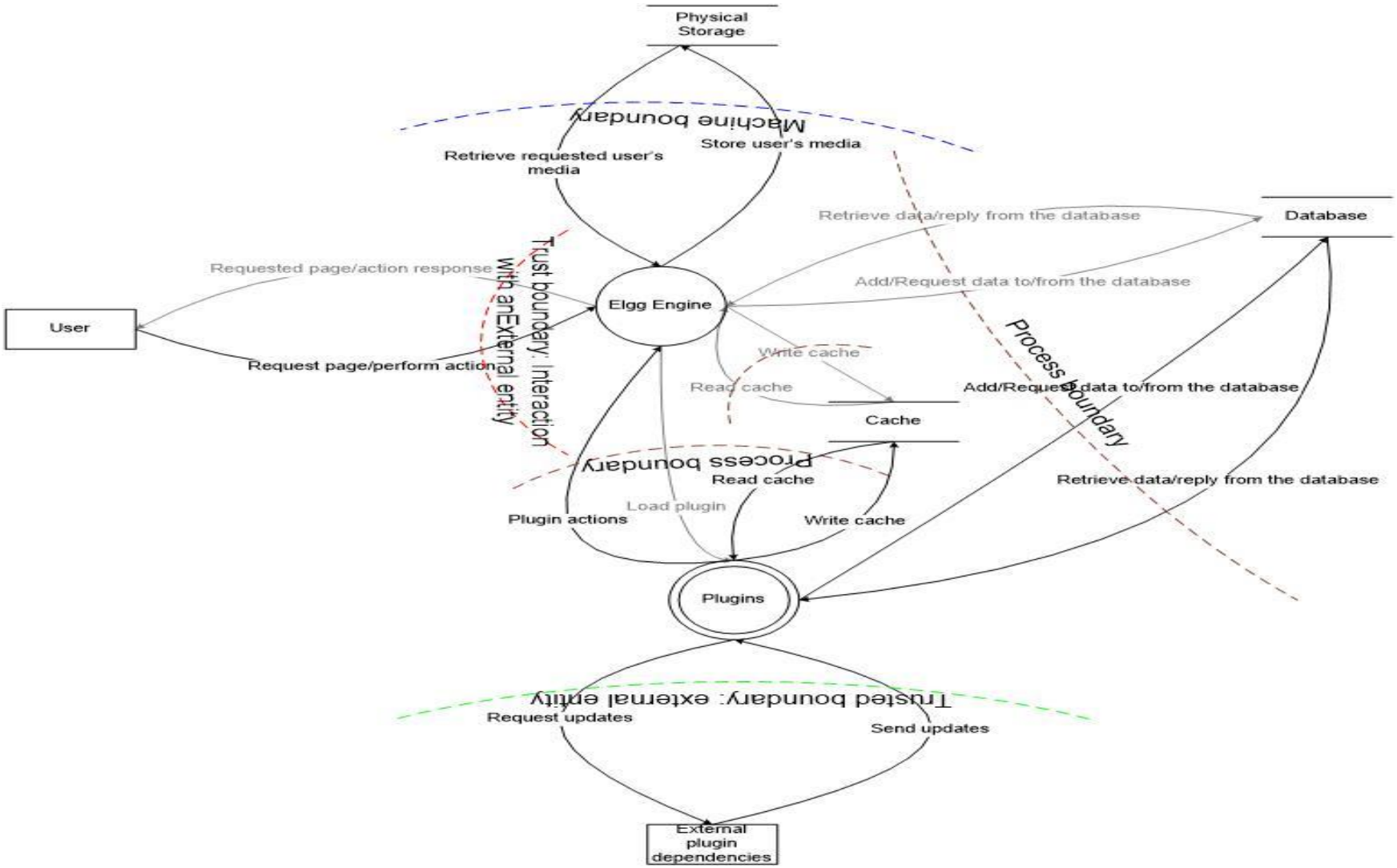



**Figure 24: DFD to threat model HACKMI2**

# Appendix 5: SDL Threat Model Analysis Report

## Appendix 5A: SDL Threat Model Analysis Report; page 1



### Threat Model Analysis Report: CSC project 2012: Hackmi2

#### Diagram Validation Messages

There are no diagram validation messages.

#### Empty Threats

All threats have descriptions.

#### Unmitigated Threats

All threats have mitigations.

#### Certifications

**The following certifications have been made for this threat model:**

| User Alias | Element Name | Threat Type | Reason for certification | Description |
|---|---|---|---|---|
| Rotondwa | Add/Request data to/from the database | DenialOfService | within a trust boundary | Trusted for dos |
| Rotondwa | Add/Request data to/from the database | DenialOfService | accepted risk (per bug bar) | Hardly cause a denial of service. |
| Rotondwa | Plugin actions | InformationDisclosure | within a trust boundary | No classified information transferred. |
| Rotondwa | Request updates | InformationDisclosure | within a trust boundary | Getting information from outside. |
| Rotondwa | Retrieve data/reply from the database | DenialOfService | within a trust boundary | The database is trusted. It is not a threat to the system. |
| Rotondwa | Retrieve requested user's media | DenialOfService | within a trust boundary | Any action from the physical storage imposes no threat to the system. |
| Rotondwa | Send updates | InformationDisclosure | accepted risk (per bug bar) | Information from external entities is not private to the social network. |
| Rotondwa | Store user's media | DenialOfService | within a trust boundary | Elgg engine is trusted. |
| Rotondwa | Write cache | DenialOfService | within a trust boundary | No dos for cache writting |
| Rotondwa | Cache | Repudiation | data store not a log, so no repudiation threat | No log is kept of actions being perfomed. |
| Rotondwa | Database | Repudiation | data store not a log, so no repudiation threat | Not data of action perfomed is stored. |
| Rotondwa | Plugins | Tampering | within a trust boundary | No data can be chaged in an installed plugin |
| Rotondwa | Plugins | Repudiation | mitigated elsewhere (provide a pointer) | Mitigated in users |

**Figure 25: SDL Threat Model Analysis report**

## Appendix 5B: SDL Threat Model Analysis Report; page 2



| | | | | |
|---|---|---|---|---|
| Rotondwa | Elgg Engine | Spoofing | within a trust boundary | Trusted process |
| Rotondwa | Elgg Engine | Repudiation | within a trust boundary | Elgg engine is a trusted process |
| Rotondwa | Elgg Engine | ElevationOfPrivilege | within a trust boundary | Elgg engine is a trusted process |

### Non-Autogenerated Elements

**Automatic threat generation has been disabled for the following elements:**

| Element Name | Element Type | Reason Provided |
|---|---|---|
| Add/Request data to/from the database | DataFlow | |
| Load plugin | DataFlow | Action initaited by Elgg engine is trusted. |
| Read cache | DataFlow | Action initiated by the Elgg engine is trusted. |
| Requested page/action response | DataFlow | From a trusted source |
| Retrieve data/reply from the database | DataFlow | Both source and destination are trusted. |
| Write cache | DataFlow | From the trusted source |

**Figure 26: SDL Threat Model Analysis report page 2**

# Appendix 6: Threat Model report

## Appendix 6A: Threat Model report; page 1



### Threat Model: CSC project 2012: Hackmi2



#### Threat Model Information

| | |
|---|---|
| **Component** | CSC project 2012: Hackmi2 |
| **Product** | Social Network: Hackmi2 |
| **SourceUrl** | hackmi2.cs.uct.ac.za |
| **Owner** | Ratshidaho Rotondwa Wayne |
| **Participants** | Many people |
| **Reviewers** | Dr Anne Kayem Prof James Gain |
| **Url** | people.cs.uct.ac.za/~rratshidaho |
| **Summary** | This is for a Computer Science honours project for the year 2012. This threat model was applied to the social network created from the elgg framework. |
| **History** | |

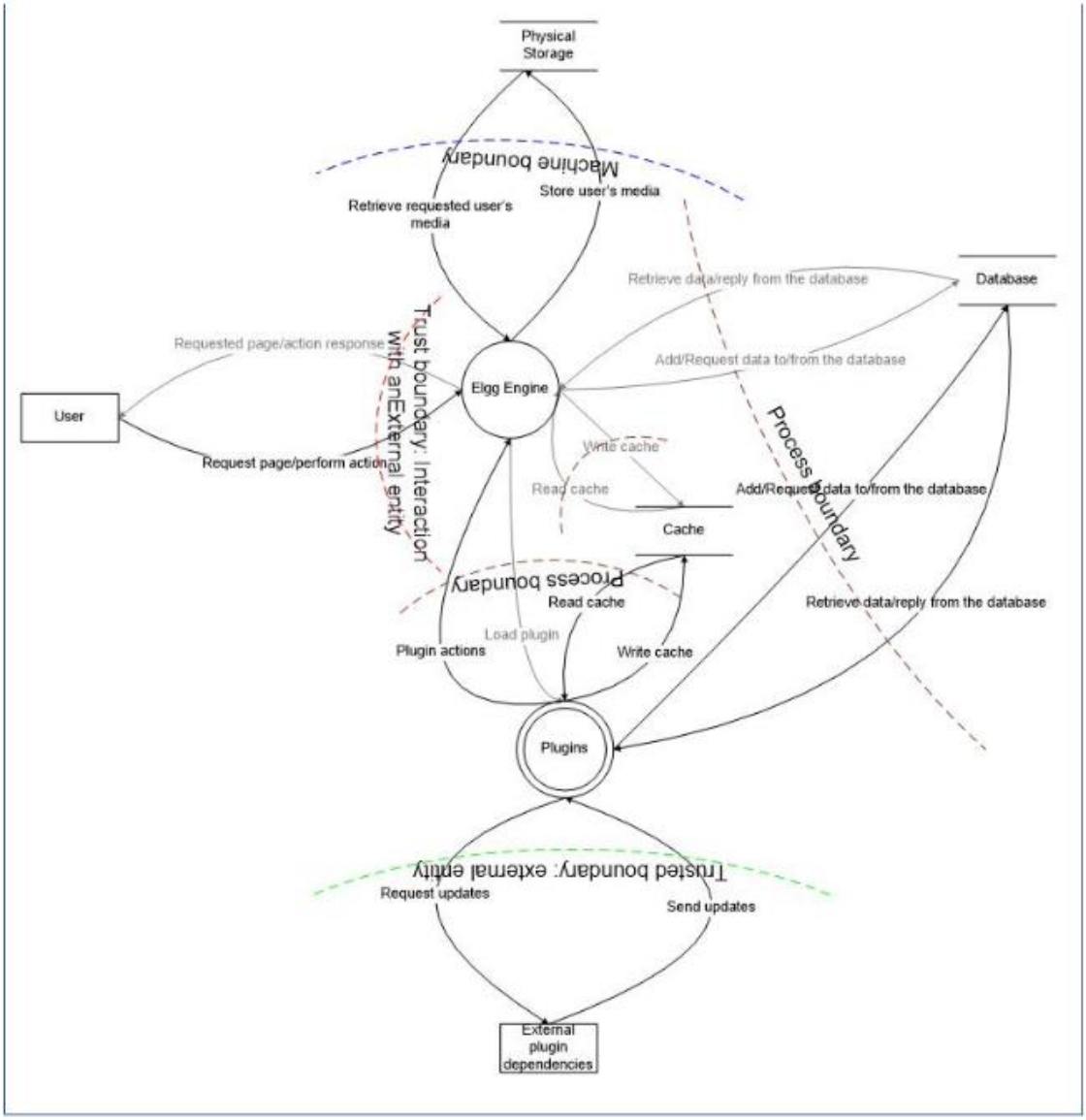
#### Data Flow Diagrams

**Context**

**Figure 27: SDL Threat Model report after modelling HACKMI2; page 1**

## Appendix 6B: Threat Model report; page 2






### Threats and Mitigations

#### Elements

| Element Type | Description |
|---|---|
| Data Flow (NotGenerated) | Add/Request data to/from the database () |
| Data Flow | Add/Request data to/from the database |
| Data Flow (NotGenerated) | Load plugin (Action initaited by Elgg engine is trusted.) |
| Data Flow | Plugin actions |
| Data Flow (NotGenerated) | Read cache (Action initiated by the Elgg engine is trusted.) |
| Data Flow | Read cache |
| Data Flow | Request page/perform action |
| Data Flow | Request updates |
| Data Flow (NotGenerated) | Requested page/action response (From a trusted source) |
| Data Flow (NotGenerated) | Retrieve data/reply from the database (Both source and destination are trusted.) |

**Figure 28: SDL Threat Model report after modelling HACKMI2; page 1**

# Appendix 6C: Threat Model report; page 3



| | |
|---|---|
| Data Flow | Retrieve data/reply from the database |
| Data Flow | Retrieve requested user's media |
| Data Flow | Send updates |
| Data Flow | Store user's media |
| Data Flow (NotGenerated) | Write cache (From the trusted source) |
| Data Flow | Write cache |
| Data Store | Cache |
| Data Store | Database |
| Data Store | Physical Storage |
| Interactor | External plugin dependencies |
| Interactor | User |
| MachineBoundary | Machine boundary |
| Multi-Process | Plugins |
| OtherBoundary | Trusted boundary: external entity |
| Process | Elgg Engine |
| ProcessBoundary | |
| ProcessBoundary | Process boundary |
| ProcessBoundary | Process boundary |
| TrustBoundary | Trust boundary: Interaction with anExternal entity |

## External Interactors

### Threats against External plugin dependencies

#### Spoofing (Threat #73)

**Threat:** Can pretended to be the original plugin external dependency.

**Mitigation:** To authenticate principals:
Basic authentication
Digest authentication
Cookie authentication
Windows authentication (NTLM)
Kerberos authentication
PKI systems such as SSL/TLS and certificates
IPSec
Digitally signed packets

To authenticate code or data:
Digital signatures
Message authentication codes
Hashes

#### Repudiation (Threat #74)

**Threat:** send malicious codes, and act as if it onl sent the updates

**Mitigation:** Trusted third parties

### Threats against User

#### Spoofing (Threat #1)

**Threat:** A unkown user steals someone's session tokken or logging credentials or hack into the system and logging as someone whose registered and starts injecting malicous code to the system

**Mitigation:** To authenticate principals:
Basic authentication
Digest authentication
Cookie authentication
Windows authentication (NTLM)
Kerberos authentication
PKI systems such as SSL/TLS and certificates
IPSec
Digitally signed packets

To authenticate code or data:

**Figure 29: SDL Threat Model report after modelling HACKMI2; page 3**

# Appendix 6D: Threat Model report; page 4



Digital signatures
Message authentication codes
Hashes

#### Repudiation (Threat #2)

**Threat:** A user pretend not to have perfomed an action like injecting a malicious code to the system

**Mitigation:** Trusted third parties
Secure logging and auditing
Digital Signatures
Secure time stamps

## Processes

### Threats against Elgg Engine

#### Tampering (Threat #16)

**Threat:** Changing some default elgg settings

**Mitigation:** Access control list
Digital signatures

#### Information Disclosure (Threat #18)

**Threat:** Most information goes through the Elgg engine and user's can try to get access to them

**Mitigation:** Encryption
Access control list

#### Denial of Service (Threat #19)

**Threat:** Account lockout
User requesting multiple page from the server

**Mitigation:** Access control list
Filtering
Quotas
High availability designs

## Multi-Processes

### Threats against Plugins

#### Spoofing (Threat #55)

**Threat:** Harmful plugin using a name of another useful plugin

**Mitigation:** To authenticate principals:
Basic authentication
Digest authentication
Cookie authentication
Windows authentication (NTLM)
Kerberos authentication
PKI systems such as SSL/TLS and certificates
IPSec
Digitally signed packets

To authenticate code or data:
Digital signatures
Message authentication codes
Hashes

#### Information Disclosure (Threat #58)

**Threat:** Setting and some systems setting exposed

**Mitigation:** Encryption
Access control list

#### Denial of Service (Threat #59)

**Threat:** An external entity sends in a lot of data just to overload the plugin process or send harmful code to crash the plugin process

**Mitigation:** Access control list

**Figure 30: SDL Threat Model report after modelling HACKMI2; page 4**

# Appendix 6E: Threat Model report; page 5



Filtering
Quotas
High availability designs

**Elevation of Privilege (Threat #60)**

**Threat:** Plugins are trusted, hence can perfom actions that an adminstrator can do and more. So plugins can be manipulated to perfom administrative actions by an authorised user

**Mitigation:** Access control list
Group or role membership
Privilege ownership
Permissions
Input validation

## Data Flows

### Threats against Add/Request data to/from the database

**Tampering (Threat #70)**

**Threat:** Can change information in the database.

**Mitigation:** Digital signatures
Message Authentication Codes

**Information Disclosure (Threat #71)**

**Threat:** Can access confidential data stored in the database.

**Mitigation:** Encryption

### Threats against Plugin actions

**Tampering (Threat #64)**

**Threat:** Change default Elgg settings making it vulnerable to other attacks.

**Mitigation:** Windows Vista Mandatory Integrity Controls

**Denial of Service (Threat #66)**

**Threat:** Can take up all resources by sending multiple requests to the Elgg engine keeping it busy all the time. Other processes won't be able to get their request processed due to lack of resources.

**Mitigation:** Quotas
Filtering
High availability designs

### Threats against Read cache

**Tampering (Threat #103)**

**Threat:** Change Elgg session tokens locking the user out.

**Mitigation:** Digital signatures
Message Authentication Codes

**Information Disclosure (Threat #104)**

**Threat:** Reveal a users' session token

**Mitigation:** Restrict plugins from getting access to cookies written by the server i.e. Access control list

**Denial of Service (Threat #105)**

**Threat:** Continuosly changing the the session token, thereby locking the user out of the system.

**Mitigation:** Access Control List

### Threats against Request page/perform action

**Tampering (Threat #25)**

**Threat:** Unauthorised users can try to mdify some data on the social network

**Mitigation:** Windows Vista Mandatory Integrity Controls
Access control list

**Figure 31: SDL Threat Model report after modelling HACKMI2; page 5**

Digital signatures
Message Authentication Codes

#### Information Disclosure (Threat #26)

**Threat:** Can request and entity that contains classified information.
SQL injections

**Mitigation:** Restrict entity access.
Hashing
Encryption

#### Denial of Service (Threat #27)

**Threat:** Can send multiple request just to take down the server.
Input harmful command that shuts down the server.

**Mitigation:** Access control list
Filtering
Quotas
Authorization
High availability designs

### Threats against Request updates

#### Tampering (Threat #75)

**Threat:** Plugin can request an update from an outside source, that can change the current state of the system

**Mitigation:** Make sure that the reponse for every request is scrutanised.

#### Denial of Service (Threat #77)

**Threat:** The plugin can continueosly send request to an outside source and thereby taking up most of the resources that is needed by other processes, and thereby halting those other processes needing the resource.

**Mitigation:** Quotas

### Threats against Retrieve data/reply from the database

#### Tampering (Threat #67)

**Threat:** Can send false retreived data

**Mitigation:** Digital signatures
Message Authentication Codes
Windows Vista Mandatory Integrity Controls

#### Information Disclosure (Threat #68)

**Threat:** Can retrieve confidential information from the database and send it to the external entity.

**Mitigation:** Encryption
Access control list

### Threats against Retrieve requested user's media

#### Tampering (Threat #49)

**Threat:** Can alter the media being returned by some other imoral medias.

**Mitigation:** Digital signatures
Message Authentication Codes
Windows Vista Mandatory Integrity Controls

#### Information Disclosure (Threat #50)

**Threat:** Can send back personal media files.

**Mitigation:** Encryption
Access control list

### Threats against Send updates

#### Tampering (Threat #78)

**Threat:** The returned data can be malicious code that alters some information

**Figure 32: SDL Threat Model report after modelling HACKMI2; page 6**

**Mitigation:** Digital signatures
Message Authentication Codes
Windows Vista Mandatory Integrity Controls

#### Denial of Service (Threat #80)

**Threat:** Can send multple actions as updates to the plugin consuming all the computing resources.
Can have malicious code that perfom denial of service

**Mitigation:** Filtering
Quotas

### Threats against Store user's media

#### Tampering (Threat #52)

**Threat:** Can change stored user's media before it is saved.

**Mitigation:** Digital signatures
Message Authentication Codes
Windows Vista Mandatory Integrity Controls

#### Information Disclosure (Threat #53)

**Threat:** The data can being stored can be revealed before being stored.

**Mitigation:** Encryption

### Threats against Write cache

#### Tampering (Threat #100)

**Threat:** Cookie poisoning

**Mitigation:** Access control list
Digital signatures

#### Information Disclosure (Threat #101)

**Threat:** Can make it possible for external entities to get access to some private inforamation such as the session tokkens

**Mitigation:** Encryption

#### Denial of Service (Threat #102)

**Threat:** Writting too many useless cache, thereby consuming all the browsers resources to store other caches.

**Mitigation:** Quotas
High availability designs

## Data Stores

### Threats against Cache

#### Tampering (Threat #90)

**Threat:** Cookie poisoning

**Mitigation:** Access control lists

#### Information Disclosure (Threat #92)

**Threat:** Session tokken/ Session hijacking

**Mitigation:** Access control list

#### Denial of Service (Threat #93)

**Threat:** Deleting or chaning the session tokkens every time a session tokken stored generated and stored by the server.

**Mitigation:** Access Control List
Authorization

### Threats against Database

#### Tampering (Threat #21)

**Threat:** Chaning user personal information

**Figure 33: SDL Threat Model report after modelling HACKMI2; page 7**

**Mitigation:** Digital signatures
Access control List

**Information Disclosure (Threat #23)**

**Threat:** Personal information stored in the database

**Mitigation:** Encryption
Access control list

**Denial of Service (Threat #24)**

**Threat:** Changing user's stored password/username in the database leading to account lockout.

**Mitigation:** Authorization
Access control list

## Threats against Physical Storage

**Tampering (Threat #45)**

**Threat:** Changing user stored media files

**Mitigation:** Windows Vista Mandatory Integrity Controls
Access control list
Digital signatures
Message Authentication Codes

**Repudiation (Threat #46)**

**Threat:** The physical storage contains the server log of all the actions perfomed.

**Mitigation:** Secure logging and auditing
Digital Signatures
Secure time stamps
Trusted third parties

**Information Disclosure (Threat #47)**

**Threat:** Can reveal stored media files to unauthorised party

**Mitigation:** Encryption
Access control list

**Denial of Service (Threat #48)**

**Threat:** Running out of storage

**Mitigation:** High availability designs
Filtering
Quotas

## Certifications

| User Alias | Element Name | Threat Type | Reason for certification | Description |
|---|---|---|---|---|
| Rotondwa | Add/Request data to/from the database | DenialOfService | within a trust boundary | Trusted for dos |
| Rotondwa | Add/Request data to/from the database | DenialOfService | accepted risk (per bug bar) | Hardly cause a denial of service. |
| Rotondwa | Plugin actions | InformationDisclosure | within a trust boundary | No classified information transferred. |
| Rotondwa | Request updates | InformationDisclosure | within a trust boundary | Getting information from outside. |
| Rotondwa | Retrieve data/reply from the database | DenialOfService | within a trust boundary | The database is trusted. It is not a threat to the system. |
| Rotondwa | Retrieve requested user's media | DenialOfService | within a trust boundary | Any action from the physical storage imposes no threat to the system. |
| Rotondwa | Send updates | InformationDisclosure | accepted risk (per bug bar) | Information from external entities is not private to the social network. |
| Rotondwa | Store user's media | DenialOfService | within a trust boundary | Elgg engine is trusted. |
| Rotondwa | Write cache | DenialOfService | within a trust boundary | No dos for cache writting |

**Figure 34: SDL Threat Model report after modelling HACKMI2; page 8**

# Appendix 6I: Threat Model report; page 9



| | | | | | |
|---|---|---|---|---|---|
| Rotondwa | Cache | Repudiation | data store not a log, so no repudiation threat | No log is kept of actions being perfomed. |
| Rotondwa | Database | Repudiation | data store not a log, so no repudiation threat | Not data of action perfomed is stored. |
| Rotondwa | Plugins | Tampering | within a trust boundary | No data can be chaged in an installed plugin |
| Rotondwa | Plugins | Repudiation | mitigated elsewhere (provide a pointer) | Mitigated in users |
| Rotondwa | Elgg Engine | Spoofing | within a trust boundary | Trusted process |
| Rotondwa | Elgg Engine | Repudiation | within a trust boundary | Elgg engine is a trusted process |
| Rotondwa | Elgg Engine | ElevationOfPrivilege | within a trust boundary | Elgg engine is a trusted process |

## External Dependencies

| ID | Name | URL | Origin | Team Owner | External Owner | Notes |
|---|---|---|---|---|---|---|
| 1 | HackMi2 | hackmi2.cs.uct.ac.za | Microsoft | Ratshidaho Rotondwa | 1 | 1 |

## Implementation Assumptions

| ID | Date/Time | Element Impacted | Assumption |
|---|---|---|---|
| 1 | 10/13/2012 2:46:31PM | Plugins | Plugins are added only by administrators, and hence are trusted |
| 2 | 10/14/2012 9:07:54PM | Elgg Engine | Always running |
| 3 | 10/25/2012 3:47:03AM | All | Apache security |
| 4 | 10/25/2012 3:47:34AM | User | We have registerd, normal users and administrators |

## External Security Notes

| ID | Notes |
|---|---|
| 1 | There exists UCT firewall which can help with denial of service attack |
| 2 | Apache built in security |

**Figure 35: SDL Threat Model report after modelling HACKMI2; page 9**

# Appendix 7: Threats per Element report

## Appendix 7A: Threats per Element; page 1



### Threats List per Element

**Add/Request data to/from the database** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can change information in the database. | Digital signatures Message Authentication Codes |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can access confidential data stored in the database. | Encryption |

- **DenialOfService** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| No description given. | No mitigation given. |

**Plugin actions** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Change default Elgg settings making it vulnerable to other attacks. | Windows Vista Mandatory Integrity Controls |

- **InformationDisclosure** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| No description given. | No mitigation given. |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can take up all resources by sending multiple requests to the Elgg engine keeping it busy all the time. Other processes won't be able to get their request processed due to lack of resources. | Quotas Filtering High availability designs |

**Read cache** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Change Elgg session tokens locking the user out. | Digital signatures Message Authentication Codes |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Reveal a users' session token | Restrict plugins from getting access to cookies written by the ser Access control list |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|

**Figure 36: Threats per Element report generated by SDL threat model; page 1**

# Appendix 7B: Threats per Element; page 2



| | |
|---|---|
| Continuosly changing the the session token, thereby locking the user out of the system. | Access Control List |

**Request page/perform action** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Unauthorised users can try to mdify some data on the social network | Windows Vista Mandatory Integrity Controls Access control list Digital signatures Message Authentication Codes |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can request and entity that contains classified information. SQL injections | Restrict entity access. Hashing Encryption |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can send multiple request just to take down the server. Input harmful command that shuts down the server. | Access control list Filtering Quotas Authorizati High availability designs |

**Request updates** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Plugin can request an update from an outside source, that can change the current state of the system | Make sure that the reponse for every req scrutanised. |

- **InformationDisclosure** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| No description given. | No mitigation given. |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| The plugin can continueosly send request to an outside source and thereby taking up most of the resources that is needed by other processes, and thereby halting those other processes needing the resource. | Quotas |

**Retrieve data/reply from the database** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can send false retreived data | Digital signatures Message Authentication Codes Windows Vista Mandatory Integrity Controls |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can retrieve confidential information from the database and send it to the external entity. | Encryption Access control list |

**Figure 37: Threats per Element report generated by SDL threat model; page 2**

- **DenialOfService** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| No description given. | No mitigation given. |

**Retrieve requested user's media** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can alter the media being returned by some other imoral medias. | Digital signatures Message Authentication Codes Windows Vista Mandatory Integrity Controls |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can send back personal media files. | Encryption Access control list |

- **DenialOfService** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| No description given. | No mitigation given. |

**Send updates** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| The returned data can be malicious code that alters some information | Digital signatures Message Authentication Codes Windows Vista Mandatory Integrity Controls |

- **InformationDisclosure** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| No description given. | No mitigation given. |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can send multple actions as updates to the plugin consuming all the computing resources. Can have malicious code that perfom denial of service | Filtering Quotas |

**Store user's media** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can change stored user's media before it is saved. | Digital signatures Message Authentication Codes Windows Vista Mandatory Integrity Controls |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| The data can being stored can be revealed before being stored. | Encryption |

**Figure 38: Threats per Element report generated by SDL threat model; page 3**

# Appendix 7D: Threats per Element; page 4



- **DenialOfService** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| No description given. | No mitigation given. |

**Write cache** (is a DataFlow)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Cookie poisoning | Access control list Digital signatures |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can make it possible for external entities to get access to some private inforamation such as the session tokkens | Encryption |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Writting too many useless cache, thereby consuming all the browsers resources to store other caches. | Quotas High availability designs |

**Cache** (is a DataStore)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Cookie poisoning | Access control lists |

- **Repudiation** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can change his/her session id to someone's else, and perfom some action, after which the user denies having done that because he did it with someone's else session id | Restrict access to cache session id |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Session tokken/ Session hijacking | Access control list |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Deleting or chaning the session tokkens every time a session tokken stored generated and stored by the server. | Access Control List Authorization |

**Database** (is a DataStore)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Chaning user personal information | Digital signatures Access control List |

- **Repudiation** threat is certified, bug is not assigned

**Figure 39: Threats per Element report generated by SDL threat model; page 4**

| Description: | Mitigation: |
|---|---|
| Unregistered user logs in as a registered user through injection | Filter input given to the input fields |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Personal information stored in the database | Encryption Access control list |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Changing user's stored password/username in the database leading to account lockout. | Authorization Access control list |

**Physical Storage** (is a DataStore)

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Changing user stored media files | Windows Vista Mandatory Integrity Controls Access control list D signatures Message Authentication Codes |

- **Repudiation** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| The physical storage contains the server log of all the actions perfomed. | Secure logging and auditing Digital Signatures Secure time star Trusted third parties |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can reveal stored media files to unauthorised party | Encryption Access control list |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Running out of storage | High availability designs Filtering Quotas |

**External plugin dependencies** (is a Interactor)

- **Spoofing** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Can pretended to be the original plugin external dependency. | To authenticate principals: Basic authentication Digest authentication Cookie authentication Windows authentication (NTLM) Kerberos authentication PKI systems such as SSL/TLS and certificates IPSec Digitally signed packets To authenticate code or data: Digital signatures Mes authentication codes Hashes |

- **Repudiation** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| send malicious codes, and act as if it onl sent the updates | Trusted third parties |

**User** (is a Interactor)

- **Spoofing** threat is not certified, bug is not assigned

**Figure 40: Threats per Element report generated by SDL threat model; page 5**

| Description: | Mitigation: |
|---|---|
| A unkown user steals someone's session tokken or logging credentials or hack into the system and logging as someone whose registered and starts injecting malicous code to the system | To authenticate principals: Basic authentication Digest authentication C authentication Windows authentication (NTLM) Kerberos authentication systems such as SSL/TLS and certificates IPSec Digitally signed packets authenticate code or data: Digital signatures Message authentication cc Hashes |

- **Repudiation** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| A user pretend not to have perfomed an action like injecting a malicious code to the system | Trusted third parties Secure logging and auditing Digita Signatures Secure time stamps |

**Plugis** (is a MultiProcess)

- **Spoofing** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Harmful plugin using a name of another useful plugin | To authenticate principals: Basic authentication Digest authentica Cookie authentication Windows authentication (NTLM) Kerberos authentication PKI systems such as SSL/TLS and certificates IPSec Digitally signed packets To authenticate code or data: Digital signatures Message authentication codes Hashes |

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Changing user's settings or personal information | ACLs Digital signatures Message Authentication Codes |

- **Repudiation** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Gainging access to someone's account or information. | Strong Authentication Secure logging and auditing Digital Signa Secure time stamps Trusted third parties |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Setting and some systems setting exposed | Encryption Access control list |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| An external entity sends in a lot of data just to overload the plugin process or send harmful code to crash the plugin process | Access control list Filtering Quotas H availability designs |

- **ElevationOfPrivilege** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Plugins are trusted, hence can perfom actions that an adminstrator can do and more. So plugins can be manipulated to perfom administrative actions by an authorised user | Access control list Group or role membe Privilege ownership Permissions Input validation |

**Elgg Engine** (is a Process)

- **Spoofing** threat is certified, bug is not assigned

**Figure 41: Threats per Element report generated by SDL threat model; page 6**

## Appendix 7G: Threats per Element; page 7



| Description: | Mitigation: |
|---|---|
| Plugin perfoming unlawful actions. | To authenticate principals: Basic authentication Digest authentica Cookie authentication Windows authentication (NTLM) Kerberos authentication PKI systems such as SSL/TLS and certificates IPSec Digitally signed packets To authenticate code or data: Digital signatures Message authentication codes Hashes |

- **Tampering** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Changing some default elgg settings | Access control list Digital signatures |

- **Repudiation** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Gainging access to someone's account or information. | Strong Authentication Secure logging and auditing Digital Signat Secure time stamps Trusted third parties |

- **InformationDisclosure** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Most information goes through the Elgg engine and user's can try to get access to them | Encryption Access control list |

- **DenialOfService** threat is not certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Account lockout User requesting multiple page from the server | Access control list Filtering Quotas High availability des |

- **ElevationOfPrivilege** threat is certified, bug is not assigned

| Description: | Mitigation: |
|---|---|
| Getting permission to perfom administractive actions | ACLs Group or role membership Privilege ownership Permissions validation |

**Figure 42: Threats per Element report generated by SDL threat model; page 7**

# Appendix 8: Recommended Fuzzing report



## Recommended Fuzzing Targets

# Pri 1: External Data Crossing A Boundary

These data flows originate from external interactors and cross trust boundaries. They are critical targets for fuzz testing.

### Request page/perform action [DataFlow]

Connects User [Interactor] and Elgg Engine [Process]

### Send updates [DataFlow]

Connects External plugin dependencies [Interactor] and Plugins [MultiProcess]

# Pri 2: Data Crossing A Boundary

These data flows cross trust boundaries. It is very important to consider fuzzing all data as it crosses a trust boundry.

### Retrieve requested user's media [DataFlow]

Connects Physical Storage [DataStore] and Elgg Engine [Process]

### Store user's media [DataFlow]

Connects Elgg Engine [Process] and Physical Storage [DataStore]

### Write cache [DataFlow]

Connects Plugins [MultiProcess] and Cache [DataStore]

### Plugin actions [DataFlow]

Connects Plugins [MultiProcess] and Elgg Engine [Process]

### Add/Request data to/from the database [DataFlow]

Connects Plugins [MultiProcess] and Database [DataStore]

### Retrieve data/reply from the database [DataFlow]

Connects Database [DataStore] and Plugins [MultiProcess]

### Read cache [DataFlow]

Connects Cache [DataStore] and Plugins [MultiProcess]

**Figure 43: Recommended fuzzing report after applying SDL to HACKMI2**

# Appendix 9: First DFD to model HACKMI2

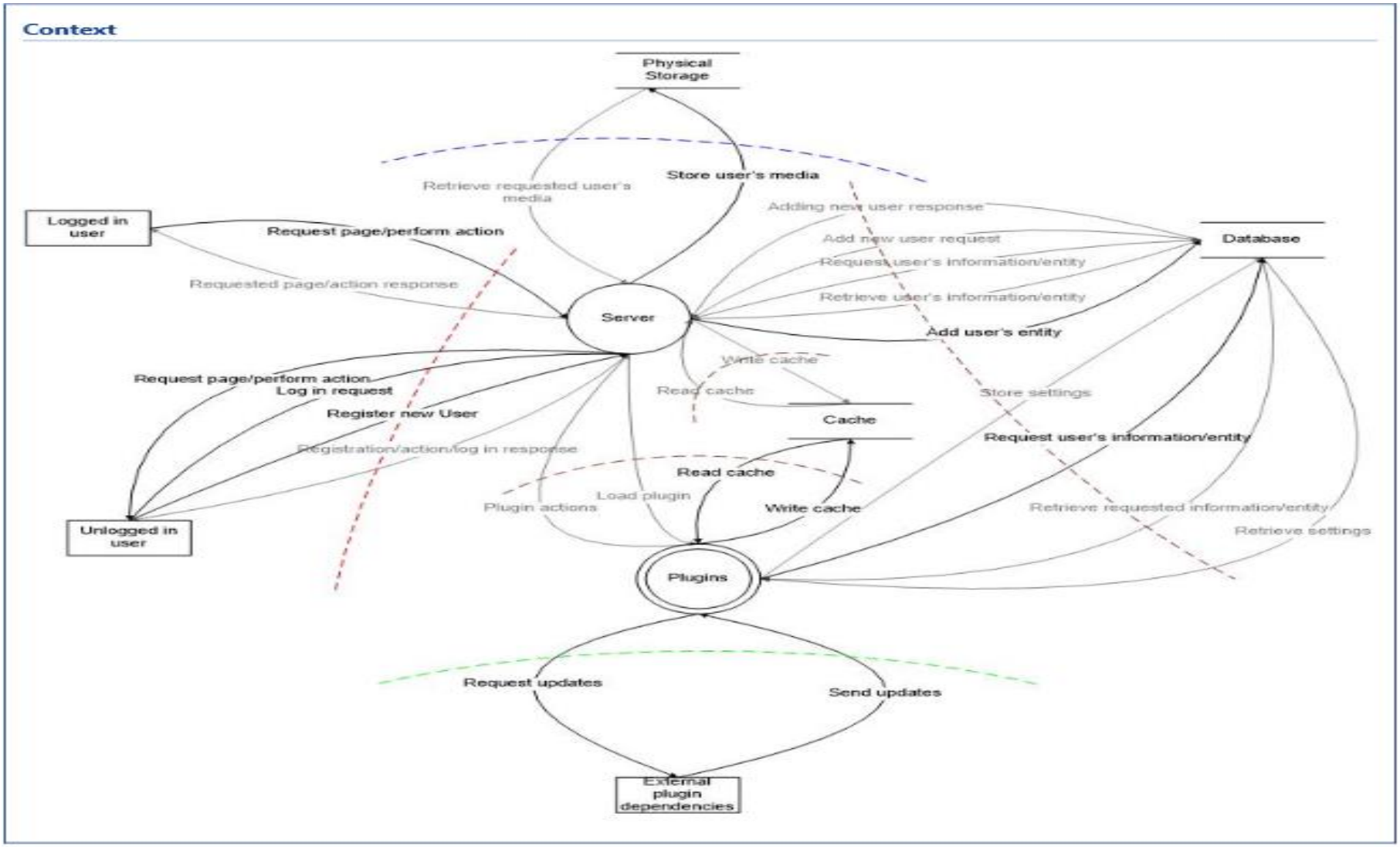



**Figure 44: Level 1 DFD**