# Threat Pilot Software Requirements Specification

First Edition

By Sara Shikhhassan 101142208,
Sam Al Zoubi 101140949,
Tejash Patel 101131066,
Jatin Kumar 101092120

## Table of Contents

# 1. INTRODUCTION

## 1.1 Purpose

This Software Requirements Specification document outline’s Team 8’s solution in developing Threat Pilot, a new threat modelling tool that uses the advantages of existing modelling tools while resolving their weaknesses. This project will be the first of numerous initiatives that will contribute to Threat Pilot's long-term objective. A threat modelling tool identifies and mitigates possible risks and impact levels of a system's security needs, elicits those needs, and informs the user of potential attack scenarios and vulnerabilities. Depending on the user's requirements, the vulnerabilities are portrayed in a variety of ways, such as a data flow diagram, interactive dashboard, or comprehensive report. This paper will define the components in order to confirm and support crucial design decisions while developing Threat Pilot.

## 1.2 Intended Audience

Professor Jason Jaskolka is overseeing the implementation of this project, which is a prototype for the Threat Pilot modelling tool. As this is a multi-year project, this SRS is meant for the next prospective students who will continue the effort to establish Threat Pilot, and it intends to define the needs of software to be developed after a comprehensive review of the existing modelling tools available.

## 1.3 Project Scope

Threat modelling is an important aspect of the system development life cycle because it elicits and understands the system security needs that must be implemented to safeguard the system's specified assets by recognising which portions of the system are most vulnerable to an attack. Threat Pilot, a novel threat modelling tool, is offered to assist system engineers in better eliciting important security needs for their systems. System engineers must have a tool that provides effective and accurate threat management solutions and analysis on their system designs in order to do so. As a result, a tool with clear objectives and usability is critical to reaching this goal. The goal of this project is to elicit Threat Pilot's requirements and software architecture design.

# 2. OVERALL DESCRIPTION

## 2.1 Product Perspective

This project is intended to be utilised by customers from various backgrounds with little knowledge of cyber security, to enable system defenders to systematically examine a potential attacker's profile, the most likely attack paths, and the assets most wanted by an attacker, allowing for educated security risk decisions. The threat modelling activity's results can assist systems engineers in eliciting security requirements to safeguard the identified assets by necessitating the deployment of certain security controls and countermeasures. Below is a diagram [1] which clearly outlines the software-to-be with its environment, subsystem interconnections and external interfaces.

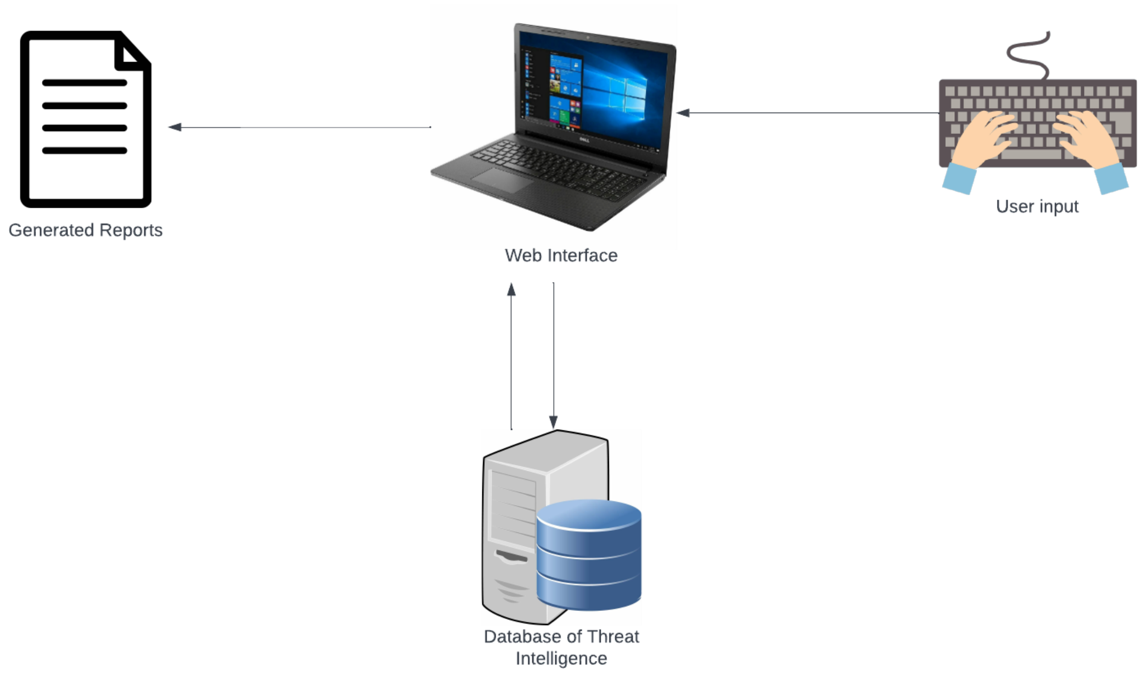

Figure 1: Threat Pilot’s high level system function [2]

## 2.2 Product Functions

The Threat Pilot's functions can be divided into three steps; running a threat model through a web application, applying threat intelligence, and producing reports. The system shall be implemented in the manner described below.

*System modelling*: This entails identifying assets, security controls, and threat agents based on user input in the form of Data Flow Diagrams (DFDs).

*Apply Threat Intelligence*: Identifying threats is the most essential task in the Threat Modeling tool. Threat Pilot would take the approach of either allowing the user to pick a standard framework such as STRIDE, LINDUNN, or CIA to produce threats or allowing the system to generate threats utilising databases from other frameworks/libraries.

*Report*: The Threat Pilot's report must be detail-oriented, consisting of information such as DFD diagrams, summary tables, system descriptions, and threat data such as name, priority, and likelihood of the threat.

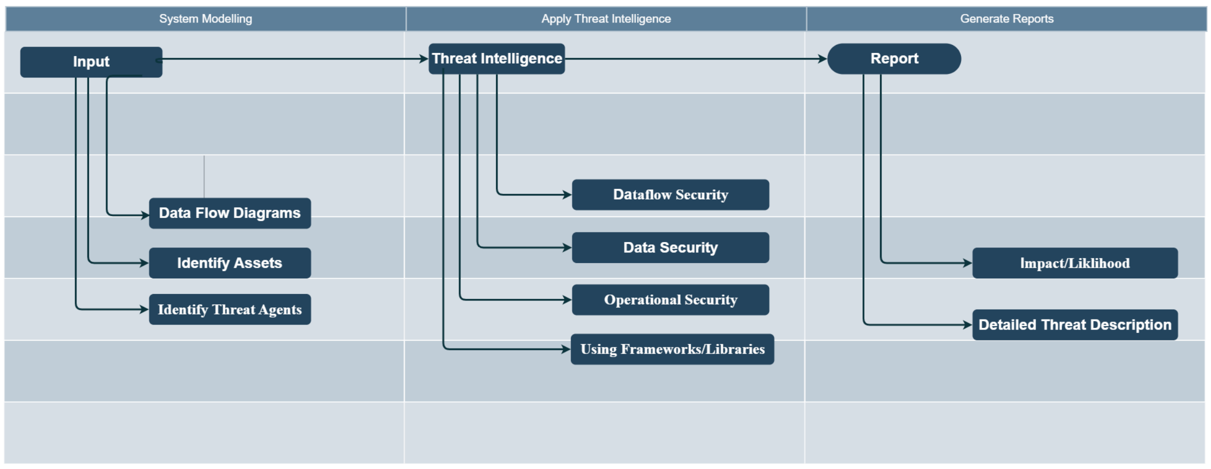

Figure 2: Threat Pilot product functions [3]

## 2.3 User Class and Characteristics

The client, as the primary user of the system, is expected to be familiar with certain framework expertise in order to utilise the system, such as CIA, LINDUNN, and/or STRIDE, however documentation detailing the various frameworks will be supplied. As the basic premise of Threat Pilot, the system's developers are expected to be extremely knowledgeable in software security, web development to maintain the interface, and database management systems to maintain the threat intelligence data.

## 2.4 Operating environment

The Threat Pilot system shall be available on cross-platforms, and operable on operating systems. The system shall be a web-based application that is supported with Chrome, Mozilla, Safari, and Opera web browsers.

## 2.5 User Documentation

The Threat Pilot system must contain thorough usability documentation, such as user manuals, as well as some demonstrational video lessons on how to use the system and example demonstrational clips. They will be accessible via a hyperlink on the system's primary dashboard.

# 3. EXTERNAL INTERFACE REQUIREMENTS

## 3.1 User Interfaces

- The system shall follow an MVC design pattern for the GUI.
- The system shall display the home page to the user once the application has been opened.
- The system shall have “Create Threat Model”, “Load Existing Model”, “Create New Template” and “Open Template” options on the home page.
- The system shall have a “Documentation” button at the top right of every page that redirects the user to the tools documentation
- The “Load Existing Model” option shall open the users directories and allow only acceptable formats for uploads
- The “Create Threat Model” option creates a new page with a list of available DFD elements on the left hand side. Available threat frameworks on the top right hand side. Threats listed on each element. Modification panel on threats on the bottom of the page.
- The Modification panel will display to the users the threats description and a “delete” button next to each threat for removal purposes.
- In the “Create Threat Model” mode there shall be “save”, “undo”, “redo”, “export”, and “generate threats” buttons at the top.

- The “save” button shall save the opened DFD model in a directory, The “undo” button undo an action. The “redo” button reverses an undo. The “export” button allows the user to generate a threat analysis report on the system model. The “generate threat'” button generates threats from a specific threat modelling framework selected.
- The “Create New Template” option opens a new page with the buttons “New Category”, “New Threat type” and “Delete”.
- The “Open Template” option lists all saved templates and allows the users to select and make any modifications.
- Buttons “Edit”, “View” and “File” will be displayed at the top panel of every page after the home page.

## 3.2 Hardware Interfaces

The application must run on all OS systems with a minimum of 2 GB RAM and 2 CPU. The system does not have any other hardware interface requirements.

## 3.3 Software Interfaces

- The system shall communicate with a database for storing and retrieving data needed by the application such as user details, project information, detailed threat list, stencil information. The database used shall be a SQL, or noSQL database.
- The system shall have separate frontend and backend applications. The frontend shall handle various user interactions with the system while the backend shall comprise the rules engine as well as the server logic needed to handle application requirements.
- The system shall utilise open source libraries such as diagram drawing library, user authentication library in the chosen language to assist with the development of the application.
- The system shall communicate with github and local file systems for saving and retrieving user projects.

## 3.4 Communication Interfaces

The communication between the client and server should utilise a REST-compliant web service and must be served over HTTP Secure (HTTPS). The client-server communication must be stateless. The tool shall be able to connect with Github. The tool shall be able to communicate with the local file system to save and upload the system model. The tool should only connect with external tools that use secure end-to-end encryption.

## 4. FUNCTIONAL REQUIREMENTS

- The system shall enable users to create a new project or load existing models.
- The system shall enable users to import or export part of a diagram that can be used by other models.
- The system shall enable users to input their data using the Data Flow Diagrams (DFD).
- The system shall provide a number of different stencils and each stencil may have different element properties.
- The system shall be able to link the stencils together using another stencil or edge of stencil and each stencil shall have a corresponding icon related to its function.
- The system shall allow stencils to be modified by name and have a list containing separate categories for stencils.
- The system shall use user's input element properties to predict security threats.
- The system shall identify software assets, security controls, and threats and their locations to create a security model.
- The system shall support STRIDE, LINDDUN and CIA framework to generate threats about the vulnerabilities exposed to attackers that might be used to break in.
- The system shall also have threat intelligence from the publicly maintained threat library, MITRE's CAPEC and from proprietary information collected by the developers to know the granular model about what attackers would do after they break in.
- The system shall be scalable to support any other framework or threat library if needed in future.
- The system shall automatically apply suggested threats to the model.
- The system shall be able generate threats on standalone stencils without any other interaction i.e. no data flow to other stencils.
- The system shall create a traceability matrix of missing or weak security controls along with their impact and probability.

- The system shall be able to prioritise the threats using guidelines for quantifying the likelihood and impact of each threat and rank them as high, medium, low or notice priority.
- On identifying threats, the system shall provide information including confidentiality, integrity, availability and ease of exploitation.
- The system shall allow users to add new threats or edit the existing threats to the model.
- The system shall link threats to their corresponding stencil through icons.
- The system shall allow users to edit threats by clicking on the specific stencil to which the threat applies.
- The system shall enable users to view security recommendation reports based on their input at any point in time.a
- The system shall generate reports in PDF format and must contain the date, DFD diagrams, summary table, system description, and detailed list of threats.
- The threat details shall contain the name, priority, status, category, description, justification and mitigation of the threat.
- The system shall allow users to save the generated threat reports and the diagrams.
- The system shall allow users to report issues regarding the Threat Pilot system.

# 5. NON-FUNCTIONAL REQUIREMENTS

## 5.1 Interface Requirements

- The system shall have a Threat dashboard which will allow users to view severity of each vulnerability and asset-level risk.
- The system shall have a Mitigation dashboard which will allow users to see the countermeasures for security control.
- The system shall have detailed documentation about the usability of the tool.

## 5.2 Performance Requirements

- The system shall easily navigate through the project management dashboard.
- The system shall offer smooth experience to users when managing/modifying threats i.e. it should not be buggy or slow.

## 5.3 Architectural Requirements

- The system shall be a web-based application and support the popular web-browsers on the popular operating systems.
- The system shall be available offline.
- The system shall have a github integration.
- The system shall be scalable to integrate with other existing tools/workflow (for example, draw.io, JIRA) and work in conjunction to make the threat modelling process time-efficient.

## 5.4 Compliance Requirements

- The system shall conform to the established look and feel of the Carleton University's branding guidelines including colours and logos.
- The system shall comply with the requirements listed on Carleton University's Fourth year Project website

## 5.5 Development Requirements

- The system's derivables shall follow the deadlines listed on Threat Pilot's proposal timeline.
- The system does not require any other special component, since all of the work can be done on members' personal computers.
- The system shall be free to use for the users.

## 5.6 Accuracy Requirements

- The system shall provide accurate threats based on the user input.
- The system shall perform a check to ensure that client input is valid.

## APPENDIX I - REFERENCES

[1] Sara Shikhhassan, *Threat Pilot's high level system function*. 05-Dec-2022. Created using lucidchart.com

[2] Stockvault.net. (n.d.). *Stockvault: Free Stock Photos*. Free Stock Photos, Images, and Vectors. Retrieved December 5, 2022, from https://www.stockvault.net/

[3] Jatin Kumar, *Threat Pilot product functions*. 05-Dec-2022. Created using diagrams.net

# APPENDIX II - WORK BREAKDOWN

Jatin Kumar: Overall description, Functional requirements, Non-functional requirements

Sam Al Zoubi: External interface requirements

Sara Shikhhassan: Introduction, overall description, some requirements

Tejash Patel: External interface requirements

| | | | |
|---|---|---|---|
| [illegible] | [illegible] | [illegible] | [illegible] |
| [Jatin Kumar], Team Member | [Sam Al Zoubi], Team Member | [Sara Shikhhassan], Team Member | [Tejash Patel], Team Member |
| Date: Dec. 5, 2022 | Date: Dec. 5, 2022 | Date: Dec. 5, 2022 | Date: Dec. 5, 2022 |