SECURITY TESTING WITH MISUSE CASE MODELING

by

Samer Yousef Khamaiseh

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

December 2016

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Samer Yousef Khamaiseh

Thesis Title: Security Testing with Misuse Case Modeling

Date of Final Oral Examination: 05 October 2016

The following individuals read and discussed the thesis submitted by student Samer Yousef Khamaiseh, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Dianxiang Xu, Ph.D.	Chair, Supervisory Committee
Jyh-haw Yeh, Ph.D.	Member, Supervisory Committee
Jidong Xiao, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Dianxiang Xu, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

DEDICATION

Dedicated to my parents

ACKNOWLEDGEMENTS

This research and experience were possible because I have had the extraordinary advice and assistance of my supervisor Dr. Dianxiang XU. Dr. XU has provided me with critical experience as a graduate student in my desire to be a scientific researcher in the future. Dr. XU's advice, help, and guidance have provided me space and trust that any effective researcher needs in order to be successful.

I also, emphatically and truly, wish to thank my parents, Yousef and Amenah, for their unlimited support, encouragement, and assistance throughout my lifetime. They have always told me that I can accomplish anything that I have set out to achieve. My parents have also taught me to be patient and committed into what I endeavor to accomplish. My parents are the main reason for any personal success that I have had and that I will have in the future. Many thanks to my sisters and brothers for their help and unlimited willingness to maintain their support and interest in my studies abroad.

Many thanks to Dr.Izzat Al-Smadi for his help and support throughout my studies. Dr. Izzat has engendered in me the basic requirements of being an effective researcher. Special thanks go to my friends Zaid Al-Omari and Tom Gabel for standing with me through this process, for providing unlimited help, support, and encouragement, and that they have always trusted in my capabilities in combination with pushing me to achieve my goals.

Many thanks go to the Department of Computer Science for its support, it has been my honor to be a graduate student in this department at Boise State University.

ABSTRACT

Having a comprehensive model of security requirements is a crucial step towards developing a reliable software system. An effective model of security requirements which describes the possible scenarios that may affect the security aspects of the system under development can be an effective approach for subsequent use in generating security test cases.

Misuse case was first proposed by Sinder and Opdahl as an approach to extract the security requirements of the system under development [1]. A misuse case is a use case representing scenarios that might be followed by a system adversary in order to compromise the system; that is a behavior that should not happen in a system.

As an effective approach used to model potential threats to the system under development, misuse cases are an effective approach for suggesting mitigation mechanisms. A mitigation use case is a use case that represents the countermeasure requirements of a misuse case.

By describing the security threats that may be exploited from the adversary's point of view, a misuse case provides an effective basis for security testing that addresses the interactions between the adversary and the system under development. Security testing also needs to verify the security mechanisms of the system against misuse cases. Thus, by representing the security requirements of the system, mitigation use cases can also be a good basis for security testing.

Misuse cases and mitigation use cases are ordinarily described in natural language. Unfortunately, this approach has difficulties and limits the ability to generate security test cases from the misuse cases and mitigation use cases. This thesis presents a new, structured approach to generating security test cases based on the extracted security test model from the textual description of the misuse cases accompanying mitigation use cases, represented as a Predicate/Transition (PrT) net.

This approach will enable the system developers to model the misuse cases accompanying mitigation use cases and then generating security test cases based on the resulting security test models, ensuring that the potential attacks are mitigated appropriately in the software development process.

This approach has been applied to two real-world applications, FileZilla Server, a popular FTP server [19] in C++ and a Grant Proposal Management System (GPMS) in Java. Experiment results show that the generated security test cases are efficient test cases that can reveal many security vulnerabilities during the development of GPMS and can kill the majority of the FileZilla Server mutants with seeded vulnerabilities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MISTA	Model-Based Integration and System Test Automation
SUD	System Under Development
SUT	System Under Test
PrT net	Predicate/Transition Net
GPMS	Grant Proposal Management System
OWASP	Open Web Application Security Project
STRIDE	Spoofing, Tempering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege
DoS	Denial of Service
MID	Model-Implementation Description
MIM	Model-Implementation Mapping

CHAPTER ONE INTRODUCTION

1.1 Background

Software security testing is one of the most important steps in developing a secure software system. It seeks to validate and verify that a software system meets the system's security goals and requirements. An effective security testing process addresses undiscovered security vulnerabilities and design flaws by using different attack scenarios.

Early consideration and addressing of software security requirements, instead of postponing discovery until the final stages of development, is a crucial step to yield a secure software system. It allows the system developers to envisage the threats posed to the software system and the countermeasures to the threats. The importance of addressing software security in the early stages of the development lifecycle is now widely acknowledged [2]. Different research studies show that, for most cases, vulnerabilities in the software security are caused by flaws in design and implementation of the software [3]. Therefore, addressing and examining the potential security threats in the early phases of the software development process enables system developers to ensure the security level of the system design and investigate alternatives which may be implemented to meet the security goals of a software system.

Misuse case modeling is an efficient method of eliciting security requirements [4]. The idea is to define potential security threats to the system under development by creating a negative use case from the system adversary point of view and define mitigation use cases that can mitigate the security threats. Misuse case modeling inherits many characteristics

of the use case modeling. For instance, using natural language to present misuse cases allows stakeholders with a non-technical background to be involved in the security requirements process.

1.2 Problem Statement

The textual description of misuse cases demonstrates the potential threats to the system under development. The textual description of mitigation use cases presents the security requirements that should be implemented to ensure that the system is resistant to those misuse cases.

One of the major difficulties in security testing is in identifying or targeting the presence of system adversaries. Misuse cases can be an effective basis for security testing as they establish the presence of an adversary by describing the malicious scenarios that the adversary may follow to compromise the system. Another significant issue in security testing is the lack of a systematic approach in selecting and validating security test cases to ensure the security of the system. Mitigation use cases can be a good source of security testing as they describe the security requirements.

Misuse cases and mitigation use cases are commonly described in natural language; such an approach offers many practical advantages. They are easy to describe and understand. However, misuse cases and mitigation use cases are not directly amenable to security testing and formal analysis.

This thesis presents a new structured approach for extracting security test models from the textual description of misuse cases and mitigation use cases. Security test models, as represented by Predicated/Transition nets, are used to generate security test cases by the

MISTA tool. Two case studies are employed to demonstrate the feasibility of the presented approach and the effectiveness of the generated security test cases.

1.3 Objectives

The main objective of this research is to develop a new systematic approach for extracting security test models from the textual descriptions of misuse cases and mitigation use cases. It will enable the misuse cases and mitigation use cases to be used directly in the security testing process.

Eliminating redundancy in security test cases represents another objective of this work. By improving the quality of the generated security test cases, this approach provides two new techniques to combine the resulted security test models. The first technique, based on STRIDE, combines resulting security test models that have the same STRIDE category(s) into one security test model. The second technique, based on Use Case, combines all security test models related to a specific use case into one security test model.

Having the ability to automatically generate executable security test cases from the generated security test model represents another objective. During this research work, the MISTA tool was chosen to automatically generate security test cases from the security test models. The MISTA tool generates test cases in various programming languages and provides test generators for comprehensive coverage criteria of test models, including reachability coverage, reachability, state coverage, transition coverage and others.

1.4 Thesis Organization

The rest of this thesis is structured as follow. Chapter 2 reviews related work. Chapter 3 presents the proposed approach. Chapter 4 describes the FileZilla server case study. Chapter 5 discusses the GPMS case study. Chapter 6 concludes the thesis.

CHAPTER TWO RELATED WORK

Misuse cases were first proposed by Sinder and Opdahl [1] as a way to elicit security requirements by addressing the potential threats to the system under design and producing additional functionalities to mitigate those threats. The first practical using of a misuse case was done by Alexander [6] in a design workshop that addresses the security and safety issues, the conclusion from this work was that misuse cases are an intuitive approach of discussing and addressing the trade-offs between different design approaches.

Different research works have tried to evaluate the effectiveness of misuse cases as an approach in addressing, extracting, and documenting the system security requirements, Meaher [7] applied the misuse cases techniques in an industrial setting, the researcher setup a design workshop and he explained the concept of the misuse cases to the participants and asked them to find security threads and model them by using the misuse cases textual descriptions and diagram. The conclusions from this work were, the misuse cases technique are promising technique in extracting the system threats and the mitigations mechanisms and also misuse case are easy to understand and improved the participant security awareness.

Guttorm et al. [8] have described that the misuse case modeling approach has been used by many of EU-funded project. The research reported that the technique and notation were helpful in the process of eliciting security requirements and easy to understand. Breivik GF [9] have reported that the misuse case modeling has been used to represent the OWASP security threats. The misuse cases have been defined in pattern form and validated

by interviewing with different stakeholders. The knowledge gained from the project confirms the lenience of understanding misuse cases and their notations. I. A. Tøndel et al., [10] suggested an approach of combining the attack trees and misuse cases for extracting security requirements and threat modeling in the requirement phase of the software projects and also they proposed to create UML activity diagram that represents the security use cases details to improve the security awareness in the developments teams, this approach need more validations by doing more experiments in order to verify the usefulness for the development team members and other improvements.

Different research works extend the misuse cases technique by extending the methodology of misuse cases and the notations of misuse cases, J. J. Pauli and D. Xu [11] presented an approach to the architectural design and analysis of secure software systems based on the extracted system requirements in the form of use cases and misuse cases. Saleh and Habil [12] proposed a new security requirements behavior model (SRBM) for obtaining trustworthiness web application and web services, they extend the generic template of misuse cases that proposed by Sinder and security use cases by Firesmith and the operational model to be more flexible and adaptable to the changes of web services and web application requirements. Røstad [13] proposed some extensions to the misuse case diagram by adding extra notations to distinguish between inside and outside attackers. Dimitrakos et al. [14] introduced other notations such as icons, coins of assets and stacks and others to recognize different types of threats. They also incorporate misuse cases with different UML diagrams and conducted different experiments to test this approach in e-Business projects such as Skipense. McDermott and Fox [15] developed an abuse case

model by adopting object-oriented modeling approach and use cases to capture and analyze security requirements.

Different misuse case models have been used to design secure software systems, UML Misuse Deployment Diagram has been used to model the potential attacks and design mitigation approaches that should be implemented in order to prevent attacks in the early stages of the software development process. J. Whittle et al. [16] introduce a new approach for modeling and executing misuse cases scenarios by using an extended interaction overview diagram (EIODs), this research work, integrates executable modeling of scenarios and weaving the aspect scenarios, the resulted model allows the stockholders to brainstorm the potential security threats and capture the mitigation mechanisms. This approach required a lot of work and need a technical specialist to follow it.

Threat modeling has proven an efficient source of security testing as the threat models able to describe software security threats. Xu et al. [23] presented a new approach to automate security testing by using threat models. In their research work, threat models were built in the fashion which follows: a) identify the software system functionalities and security goals; b) identify security threats for each identified functionality in systematic manner by using STRIDE classification [5]; and c) create security threat net represented in Predicate/Transition net for each identified threat. In order to automatically generate security test cases from the created threat nets, the MISTA (previously called ISTA) tool generates test cases based on TMID specification. TMID specification contains a threat model that is represented in PrT net and Model Implementation Mapping (MIM). In this research work, two case studies have been conducted, FileZilla FTP server and Magento, web-based online shopping [28]. In both case studies, mutants have been created, and the

security test cases have been executed against them. The results of this research work indicate that the generated security test cases were efficient in killing the majority of the mutants in both case studies. However, this research does not discuss how to create threat nets (i.e., security test models). This thesis aims to address this issue.

CHAPTER THREE THE APPROACH

As shown in Figure 1, the approach consists of three steps: (1) conduct misuse case modeling, (2) create security test model, and (3) generate security test cases from the security test model by using The MISTA tool. In the following, we elaborate on each of them.

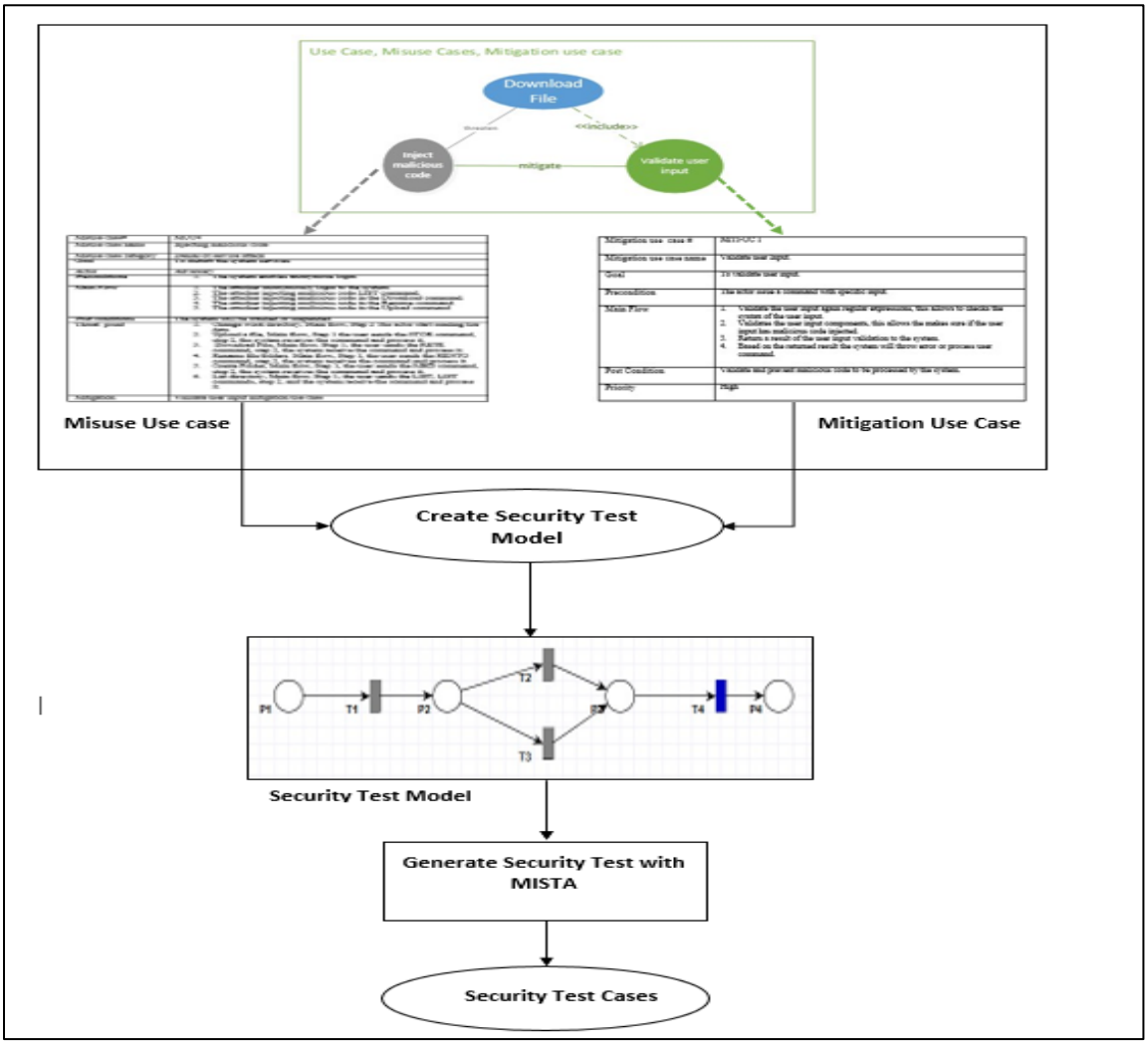

Figure 1: The Approach

3.1 Misuse Case Modeling

Misuse case modeling is the first step in our approach. It describes security attacks against use cases as well as the security features needed to mitigate them.

3.1.1 Defining Use Cases

Use cases describe the functional requirements of the system under development. Use case modeling is a structured approach to addressing the interactions between a system and its actors. In this research, the following steps are used to define use cases:

- Define use case actors, their desired system functions, and interactions between actors and system functions according to system documentation, interviews with the system stockholders, and user experiences.
- Create a use case diagram which depicts the actors, use cases, and interactions.
- Create use case textual description.

Figure 2 provides a sample use case diagram that consists of "Create New Proposal Document", "Submit a Proposal by PI", "Save Proposal", and "Notify Users" use cases. The arrows between the actor and the use cases represent the relationship between the use cases and the actor. An "include" relationship is signified by using an arrow that is labeled "include". The extend relationship is represented by an arrow labeled "extend".

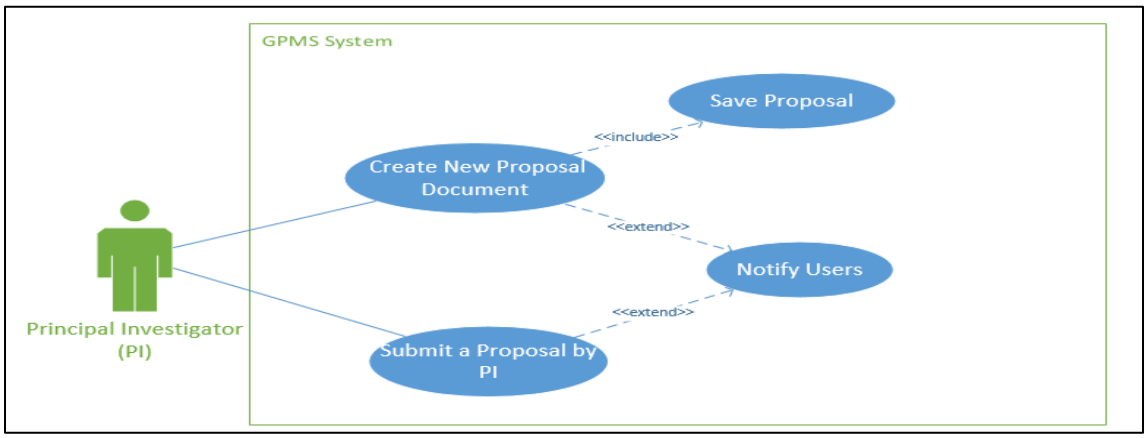

Figure 2 : A Sample Use Case Diagram

Table 1 shows the template of use cases used in this thesis. Generally, the description of a use case consists of; use case ID, name, actor, goal, precondition, post-condition, extension points, alternative flow, exception flow, and recovery flow that describe the system requirement, goals and the interaction between the actor and system. The extension points field is used to represent the extend relation with other use cases and an underlined step in the main flow is used to indicate to the reader the "include" relationship of other use cases. Table 2 shows a sample use case description.

Table 1. Use Case Template.

Use case #	Represents the use case number, that can be used for tracing the use cases.
Use case name	Represents the name of the use case.
Actor	Represents the actors of the use cases, by listing all the stakeholders that will use the use case.
Goal	Represents the target of the use case.
Preconditions	Represents the conditions that the system should ensure to be true before starting the use case
Main Flow	Represents a sequence of steps that describe how the use case goal can be achieved. Main flow represents the interactions between the actors and the system feature.
Post-Condition	Represents the states of the system after the execution of the use case.
Extensions Points	In some cases, a use case may extend other use cases whose details described in other use case description.
Alternative flow	Represents a set of alternative steps that can be performed instead of one or more step in the main flow.
Exception flow	Represents a set of conditional steps that are a response to the exceptions in one or more step in the main flow that prevent the use case from achieving its goal.
Recovery flow	Represents a set of conditional steps that response to a failure at one or more step in the main flow and how the system should react to accomplish the use case.

Table 2: Description of Use Case "Submit a Proposal by PI"

Use case #	UC-3
Use case name	Submit a proposal by principal investigator (PI).
Actor	Principal Investigator (PI).
Goal	To submit the proposal by PI.
Preconditions	1. The PI created the proposal and signed it. 2. The Co-PI(s) signed the proposal. 3. The proposal status not submitted.
Main Flow	1. The actor login to account. 2. The actor selects "My proposals" action. 3. The actor selects the proposal by selecting the edit proposal action. 4. The system opens the proposal in edit mode. 5. The actor signs the proposal. 6. The actor selects the submit action. 7. The system sends a notification to the department chair, PI, Co-PI(s) and Senior Personnel. 8. The system records the request in the user audit log.
Post-Condition	1. The proposal status changed to waiting for chair approval. 2. The actor has read access to the proposal.
Extension Points	1. Step 7, extends Notify users use case.
Alternative flow	**2.a The actor uses the research engine** 2.a.1 The actor inserts the proposal information in the search fields. 2.a.2 The system returns the search result. 2.a3 The actor selects the proposal. The use case continuous at *The actor selects the submit action in MF*
Exception flow	**4.a CO-PI(s) not signed the proposal** 4.a.1 The system shows an error message that CO-PIs are not signed on the proposal.
Recovery flow	None.

3.1.2 Defining Misuse Cases

A misuse case represents a potential security threat against some of the use cases. Misuse case is the inverse of a use case, and the actor of the misuse case is the inverse of the use case actors (i.e., adversary). We define misuse cases as follows;

- Define the misuse case actors, what potential security threats they represent, and the interaction between the actors (i.e., Adversaries) and the system functions in structured and systematic manner. This is done by applying all potential STRIDE threats (spoofing identity, tampering with data, repudiation, information disclosure, denial of service, and elevation of privilege) and security goals (Authentication, Authorization, Confidentiality, Integrity, Accountability, Availability, Non-repudiation) to each step in the main flow and alternative flow of use cases.
- Extend the use case diagram to include the misuse cases. Use case/misuse case diagram depicts the misuse cases and threaten use cases.
- Create a textual description for each misuse case.

Figure 3 is a sample use case/misuse case diagram that consists of the use cases denoted by blue ovals, and the actor in the regular use case diagram (see Fig.2). The diagram also shows the threat in terms of misuse-actor and misuse case. The misuse actor and misuse case are colored with inverted colors like black or gray colors to distinguish them from the use cases and the use case actors. Additionally, "threaten" is the relationship between the use case and misuse case. Each use case may be threatened by several misuse cases and each misuse case may threaten many use cases.

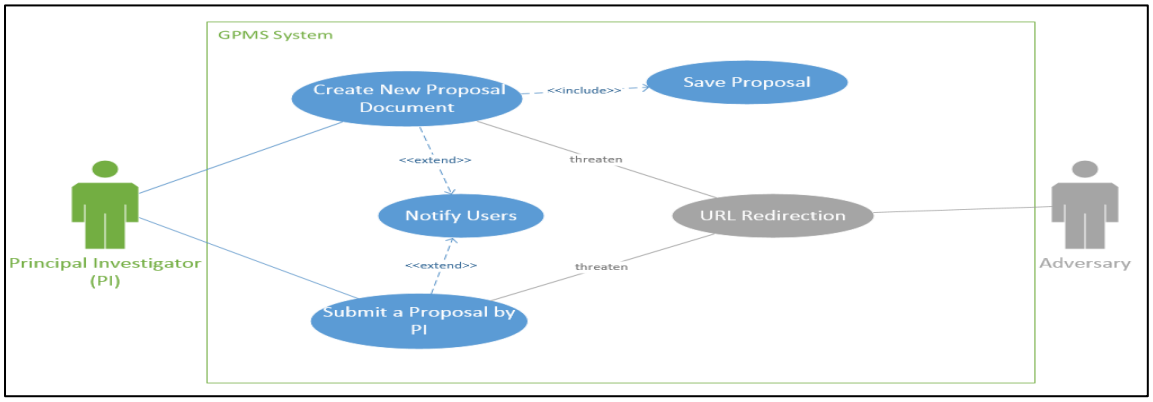

Figure 3: Use Case/Misuse Case Diagram.

Table 3 shows the template for misuse case textual description used in this thesis. Similar to the use case description, misuse case description has; misuse case ID, name, goals, actor, preconditions, post-conditions, and main flow fields that describe in detail what the adversary might try to do. The differences in misuse case description from the use case description are: misuse case category field is used to classify the misuse case category according to STRIDE classification, a threat point field that lists the use cases threatened by the misuse case, and mitigation filed representing the mitigation use case that can mitigate the misuse case. Table 4 shows an example of misuse case textual description.

Table 3: Misuse Case Template

Misuse case #	Represents the misuse case number. This field used for tracing and organizing the misuse cases.
Misuse case name	Represents the misuse case name.
Misuse case category	Represents misuse case STRIDE classification
Goal	Represents the goal of misuse case
Actor	Represents the actors of the misuse cases, by listing the possible adversaries.
Preconditions	Represents the conditions that should be true before performing the attack. For example, having username and password of a system user
Main Flow	Represents a sequence steps that demonstrate the scenario of the attack.
Post-conditions	Represents the states of the system after performing the misuse case.
Threat point	Maps each misuse case to any use case that threaten by
Mitigation	Map the misuse case to any mitigation use case that mitigated by.

Table 4: Description of "URL Redirect" Misuse Case

Misuse case #	**MUC-1**
Misuse case name	URL Redirect Attack.
Misuse case category	Information disclosure, DoS and elevation of privilege.
Goal	To redirect the user to a malicious website.

Actor	Adversary
Preconditions	1. The actor has an account on the system or has other user login information.
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor selects a proposal and opens it by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions.
Post-conditions	1. The system will redirect the user to the malicious site.
Threat point	1. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 2. Approve/Disapprove proposal by IRM, Main flow, step 3, The actor sign the proposal.
Mitigation	Cross-site Scripting (XSS) prevention.

3.1.3 Defining Mitigation Use Cases

Mitigation use cases represent the countermeasure requirements of the misuse cases. In order to define the proper mitigation use case for each misuse case identified, the following steps have been applied;

- Define the mitigation use cases by applying STRIDE and the Open Web Application Security Project (OWASP) [18] to each misuse case. STRIDE and OWASP were used as they represent both the application threats and the proper countermeasures.
- Extend use case/misuse case diagram to include the mitigation use cases.
- Create a mitigation use case textual description.

Figure 4 is a sample use case/misuse case/mitigation use case diagram. This diagram consists of the use cases, misuse cases, and mitigation use cases. “Include” is the relationship between use case and mitigation use case and “mitigate” is the relation between misuse case and mitigation use case. In Figure 4, the mitigation use case (i.e.

Cross-Site Scripting Mitigation) is colored green to distinguish it from the use cases, and misuse cases. An "include" relation is used to connect the use cases (i.e., Create New Proposal Document, Submit Proposal) with mitigation use case, and the "mitigate" relation used to connect the misuse case (i.e., URL Redirection Attack) and mitigation use case.

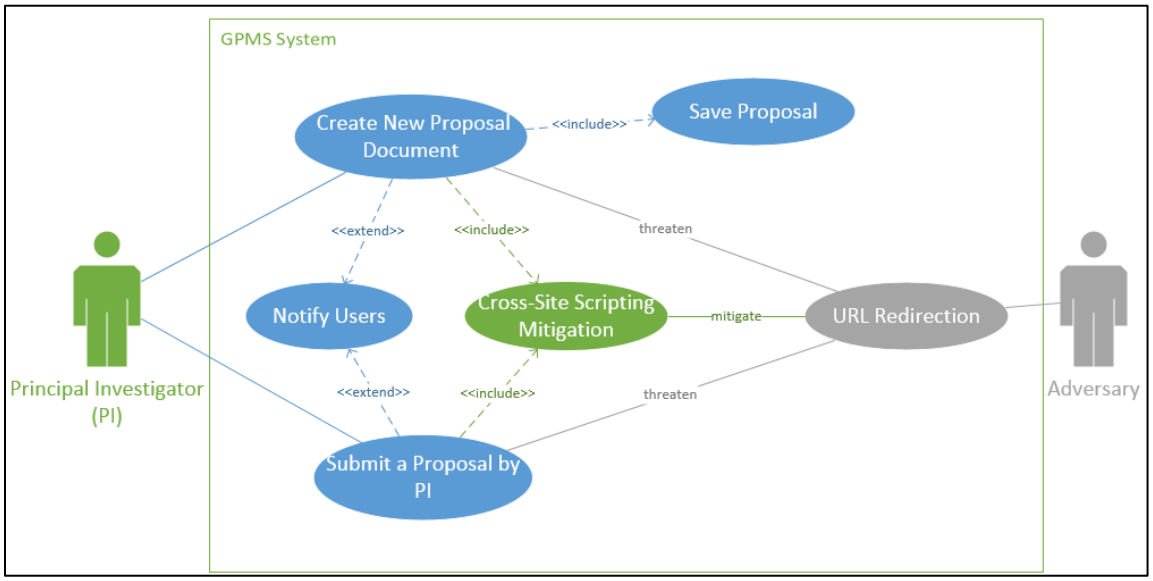

Figure 4: Use Case/Misuse Case/Mitigation Use Case Diagram

Table 5 provides the template of mitigation use cases used in this thesis. Similar to the use case and misuse case description, a mitigation use case description has: mitigation use case ID, name, goal, precondition, and post-condition fields. The differences between the mitigation use case description and use case description are: (1) the priority field that represents the importance of having the mitigation use case and, (2) the exclusion of an actor field. Table 6 shows a sample mitigation use case textual description.

Table 5: Mitigation Use Case Template

Mitigation use case #	Represents the mitigation use case number. This field used for tracking and organizing the mitigation use cases.
Mitigation use case name	Represents the mitigation use case name.

Goal	Represents the goal that should be achieved from the mitigation use case.
Precondition	Represents the conditions that initiate the mitigation use case.
Main Flow	Represents a sequence of steps that describe the security requirement implementation.
Post Condition	Represent the system states after executing the mitigation use case
Priority	Represent the priority of mitigation use.

Table 6: Description of "Cross-Site Scripting" Mitigation Use Case

Mitigation use case #	**MITI-UC 1.**
Mitigation use case name	Cross-site scripting (XSS) prevention.
Goal	To prevent the XSS attack.
Precondition	1. The actor injects malicious script code. 2. The actor uploads malicious script code.
Main Flow	1. Validate user input by using whitelist technique. 2. Use Output Escaping technique to ensure any JavaScript code is converted to safe display. 3. Use HTML escape JSON values and decode the JSON values and safely parse it.
Post Condition	1. The XSS attack malicious code will not be executed. 2. The application will render the web pages safely and appropriately. 3. The application protects the user data.
Priority	High.

3.2 Extract Security Test Model

A security test model is represented as a Predicate/Transition (PrT) net in the MISTA tool. A PrT net is a 7-tuple <P, T, R, L, Σ, ϭ, M0>, where,

1. P is a set of places (circles) that represent a state or condition.
2. T is a set of transitions (rectangles) that represent functions or events.
3. R is a set of normal arcs, representing a relationship between a place and a transition.
4. L is a labeling function on arc R. each label is a tuple of variables or constant in Σ.

5. Σ is a set of constants and relations (e.g. arithmetic relation).
6. Ϭ is a guard function on T.
7. M0 is initial marking, where, M0 (P) is the set of tokens in predicate P, and each token is a tuple in Σ.

3.2.1 Transforming Misuse Case Textual Description into PrT Net

We extract PrT net based on the main flow of a misuse case which depicts a sequence of steps to be followed by the adversary to compromise the system. Misuse case main flow description includes different types of steps that represent an attack on the system. For example, a Repetitive step represents a reiteration of an attack, for instance, one in which new user accounts are created automatically. In this research, four types of steps have been defined; Simple step, Repetitive step, Conditional step, and Concurrent step. The representation of those steps in PrT net is discussed in the following sections.

3.2.1.1 Mapping a Simple Step into PrT Constructs

Simple Step represents executing a simple operation in the system environment by the actor. One example would be; *select login action* or *submit proposal action*. This step can be represented in PrT net as follow:

1. Define one transition “t” that represents the Simple Step.
2. Define input place P-IN and output place P-OUT.
3. Use input arc R-IN to connect the input place P-IN with transition “t” and output arc R-OUT to connect the transition “t” with output place.
4. Use arc label to define the input parameters by labeling the R-IN arc, and use R-OUT label to define the output parameters if exist.

In Table 7, the first step in the main flow is an example of a Simple Step. Figure 5 shows the representation of a Simple step by the first step in Table 7.

Table 7: Automatically Register User Account Misuse Case Description

Misuse case #	**MUC-1**
Misuse case name	Automatic register user accounts
Misuse case category	Denial of service attack (DoS).
Goal	To automatically create malicious user accounts and/or crash the service.
Actor	System, adversary.
Preconditions	1. The system is up and running.
Main Flow	1. The actor access to the system main page. 2. The actor automatically and repeatedly does: 2.1 The actor selects the SingUp page link. 2.2 The actor fills the required fields. 2.3 The actor selects SignUp action.
Post-conditions	1. Create malicious users accounts. 2. The signup process will be suspended or crashed.
Threat point	1. Create user account, Main flow, Step 4, The system receives the signup request.
Mitigation	Honey Token Form component mitigation use case

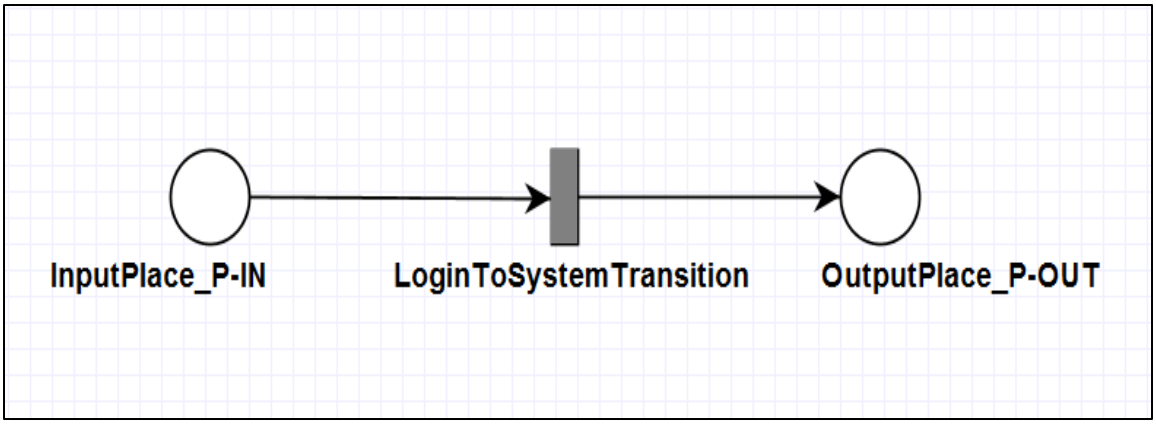

Figure 5: Simple Step Mapping Example

3.2.1.2 Mapping a Repetitive Step into PrT Constructs

The Repetitive step represents a repetitive action executed by the actor on the system environment. In distinguishing repetitive step from other steps, a keyword such as

a repeat, iterate, loop, and other keywords demonstrate that the step is repetitive. In misuse case or use case textual descriptions, a Repetitive step has sub-steps which represent the repetitive blocks as shown in Table 7 above. Step 2 represents an example of a repetitive step and the sub-steps 2.1, 2.2, and 2.3 are the repetitive block. Repetitive steps can be mapped to PrT constructs as follow:

1. Define a transition "t(i)" for each repeatable block. For example, in step 2 we have 3 sub-steps (repeatable blocks), so we create 3 transition "t(i)" (e.g. i = 1,2, 3…., n).
2. Define input place P-IN for each transition "t(i)". For step 2 in the table, we create 3 P-IN places.
3. Connect each input place P-IN with transition "t(i)" by using input arc.
4. Connect transition "t(i)" with P-IN(i+1) by using output arc. For example, connect the transition of first repeated block with input place of the second transition.
5. Connect the last transition "t(n)" with first input place "P-IN (i=1)" by using output arc. For example, connect the transition of sub-step 3 with input place of the first transition.
6. Define the guard condition that control the loop in the last transition "t(n)". The guard condition can be explicitly or implicitly defined in the repetitive step. The guard condition will have two variables, the first variable (e.g. X) represent the number of iterations and other variable (e.g. Y) used to update the loop control value.
7. Use Arc label to pass the loop control variables between transitions. The Arc label between the last transition and first transition will hold the second variables where

other Arc labels will hold the first variables. Figure 6 shows an example of mapping step 2 into PrT construct, each transition corresponds to each sub-step, the guard condition has the variables X and Y, and arc labels defined to hold the guard condition variables between transitions.

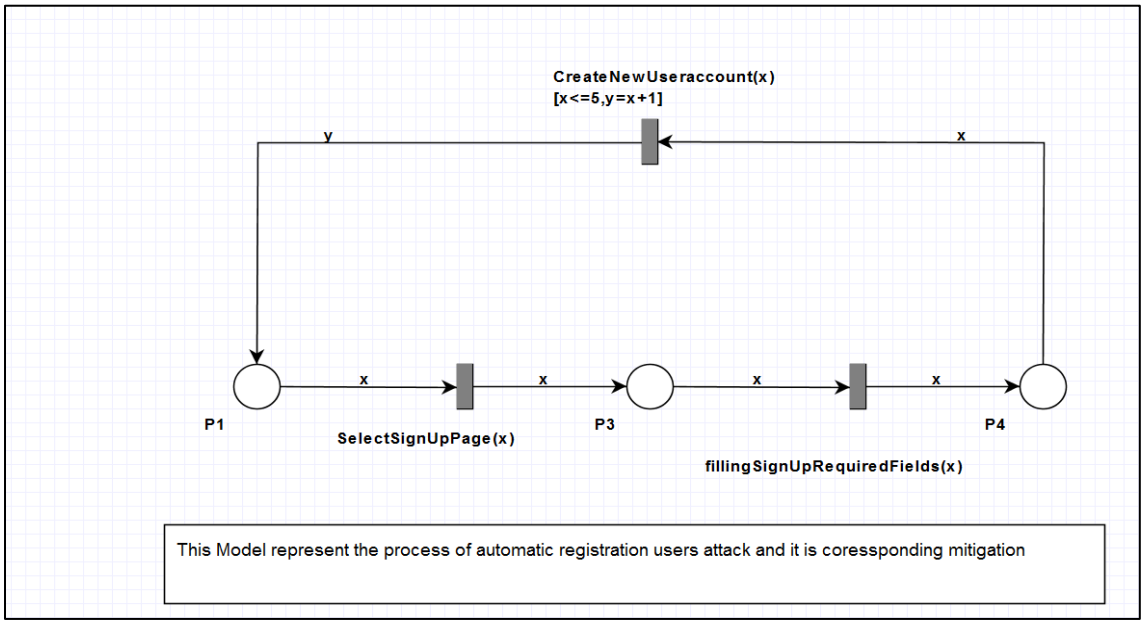

Figure 6: Repetitive Step Mapping Representation Example

3.2.1.3 Mapping a Concurrent Step into PrT Constructs

A Concurrent or Parallel step represents a parallel/concurrent action executed by an actor on the system environment. Keywords like parallel, concurrent, and other keywords indicate that it is a concurrent step. Similar to the repeatable step, the concurrent step has sub-steps or blocks that represent the parallel blocks. Concurrent step is mapped to PrT constructs as follows:

1. Define a transition “t(i)” for each block.
2. Define output place “P-OUT” for each transition “t(i)” defined in the previous step.

3. Use output arc “R-OUT” to connect each transition with corresponding transition t(i).
4. Define Input Place “P-IN”, that will be representing the input place for all of the transitions.
5. Use input arc “R-IN” to connect the “P-IN” with all transitions “t(i)”.
6. Use arc label for “R-IN” to define the input parameters and arc label for “R-OUT” to define the output parameter if that parameter exists.

Figure 7 shows an example of mapping a Concurrent into PrT construct. The “INPUT PLACE” represents the input place for the concurrent net and the “Input-T1, Input-T2 and Input-T3” are the input parameter for T1, T2, and T3 respectively and “Output-T1, Output-T2, and Output-T3” represent the output parameter for T1, T2, and T3, where, “T1-Output-Place, T2-Output-Place, and T3-Output-Place” represent the output place for the T1, T2, and T3 respectively.

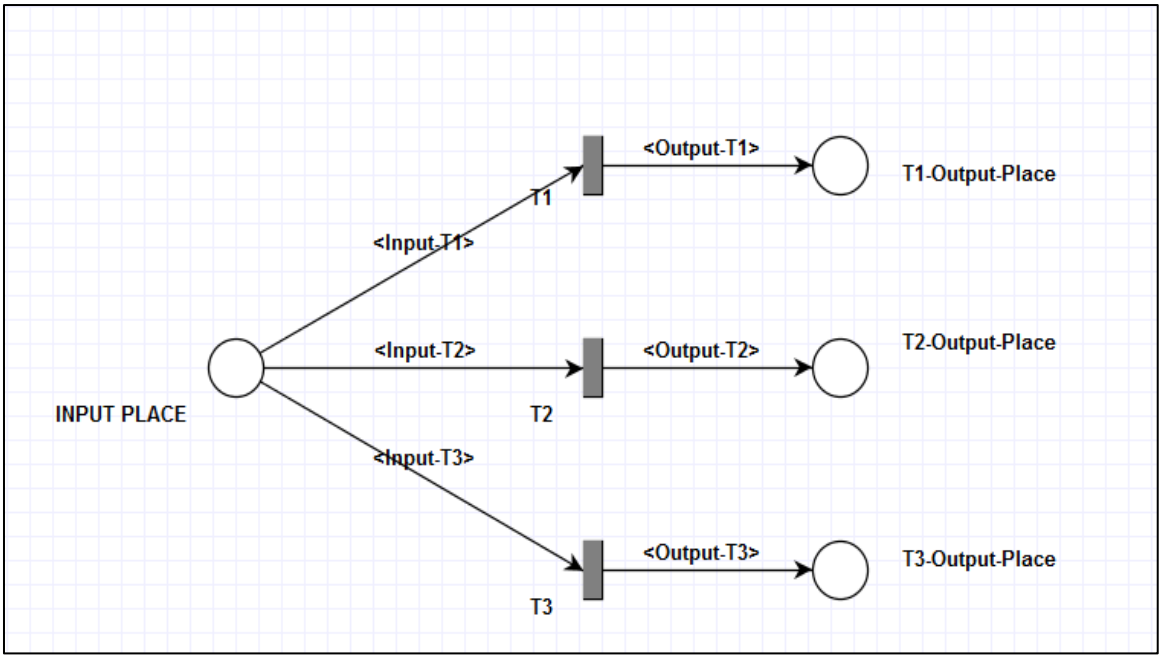

Figure 7: Parallel Step Mapping Representation Example

3.2.1.4 Mapping a Conditional Step into PrT Constructs

A Conditional step is a simple step with a guard condition. Conditional step contains two parts, IF and ELSE parts. The guard condition of the conditional step can be extracted from the IF part. Conditional step can be mapped into PrT construct as follows:

1. Define transition “t” for IF part and other transition “t” for the EISE part.
2. Define output place for each transition in previous step P-OUT.
3. Define a single input place P-IN for the conditional step (i.e. for the IF part and ELSE part).
4. Use input arc R-IN to connect the P-IN with all transitions.
5. Use output arc R-OUT to connect each transition with corresponding output place P-OUT.
6. Define guard condition for each transition.
7. Use input arc label to define the input parameter for each transition and output arc label to define the output parameters.

Figure 8 shows an example of mapping a Conditional step into PrT construct. The model has one input place which is labeled “Conditional-Input-Place” and two transitions, the first one is “Transition-T1-IF-PART” that represents the IF part in a conditional step which is a transition that will be fired if the input value “Input” is equal to 5, whereas, the “Transition-T2-ELSE-PART” represents the ELSE part that will be fired if the “input” value is not equal to 5.

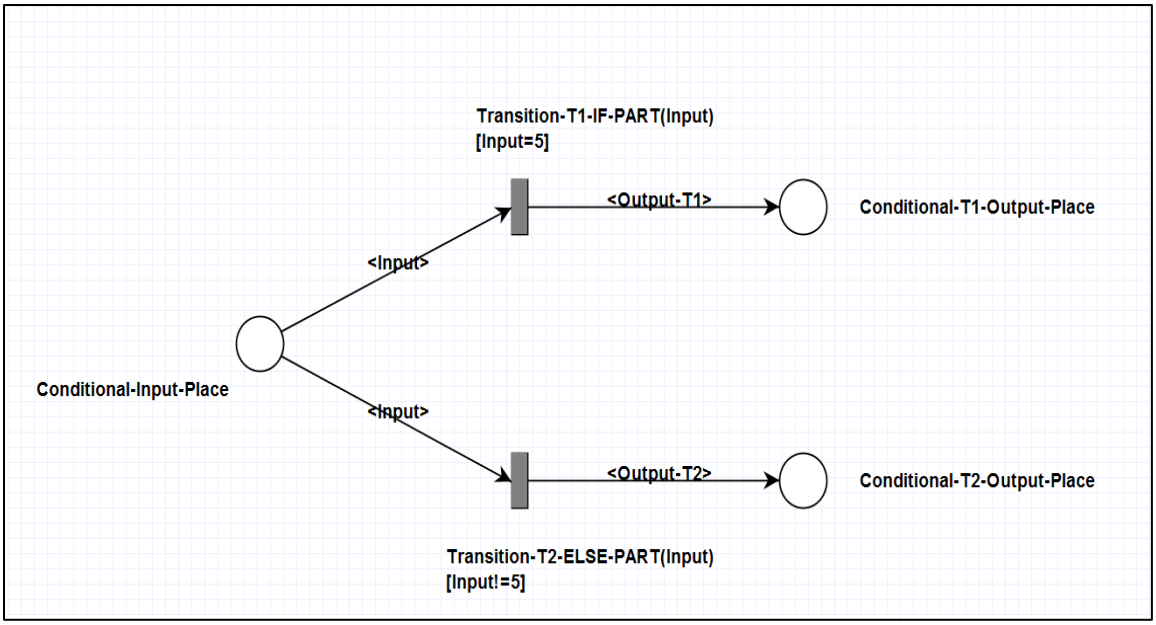

Figure 8: Conditional Statement Mapping Example.

3.2.1.5 Mapping Misuse Case Main Flow into PrT Constructs

The misuse case main flow may have a combination of more than one step of different types, such as simple step and/or repeatable step, representing the attack scenarios. In order to extract the PrT constructs from the main flow that represents the misuse case security test model, the following steps have been followed:

1. Misuse case main flow consists of "I" steps (e.g. I = 1, 2, 3... n). Starting from the first step of the main flow of the misuse case, define the step type (e.g. Simple step).
2. Based on the step type, use the steps that have been specified and explained in the previous sections to model the step in PrT constructs.
3. Repeat the previous two steps through to the final step of the main flow.
4. Use direct arc to connect the outplace P_OUT of step (i) with the transition of step "(i+1)". Starting from the first output place P_OUT.

5. Verify the precondition of each step (i). The precondition of the transition represents the input places and associated arc labels, however, some of the transition “t (i)” has two input places. The first input place comes from step modeling itself, based on its type. The other input place will come from step 4; if the two input places are identical, the input place and directed arc that was created by step modeling based on its type should be removed, keeping the input place which was created from step 4. Figure 9 shows an example of the extracted security test model from Automatically Register User misuse case (see Table 7).

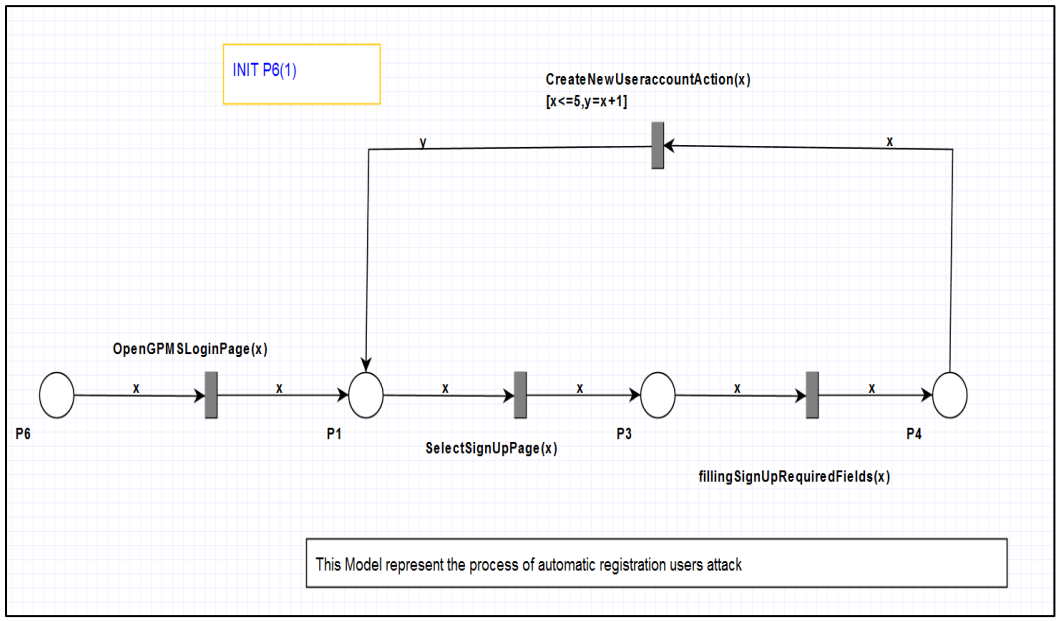

Figure 9: Misuse Case Mapping Representation Example.

3.2.2 Transforming Mitigation Use Case into PrT Constructs.

The textual description of mitigation use case describes the security requirements to prevent attacks. Based on the understanding of textual descriptions of mitigation use cases, a “validation point” is extracted which represents successful resistance to an attack. A “Validation point” represents a test oracle for the system regardless of whether it secures

against attacks. For example, in an FTP server after executing a command, the FTP server responds with an integer value, the integer value returned represents a validation point (e.g. 0 if successful, any other value if failed). In GPMS application, after performing a redirect to another website attack, the validation point will be a GPMS proposal document and not another website. To represent the mitigation use cases in PrT constructs, these steps have been applied:

1. Define a transition “t” for the validation point.
2. Define a P_OUT place for the transition “t”.
3. Use output arc to connect the transition “t” with P_OUT place.

3.2.3 Combine Misuse Case and Mitigation Use Case into PrT Constructs

A security test model consists of misuse cases PrT constructs and mitigation use cases PrT constructs. The security test model represents the attacks and the mitigation mechanism. To combine the misuse cases PrT with mitigation use case PrT:

1. Use input arc “R_IN” to connect the final place of the misuse case PrT that represents the P_OUT place of the last transition in the main flow of the misuse case with mitigation transition “t”.
2. After combining those PrT constructs, define the initial marking for the security test model *M0*. Places can hold structured data and primitive, called token. Each token is a tuple of constants. A constant is a number, symbol, and string. Marking is a distribution of tokens in all places of a function net, collecting the tokens from all places of the net will be viewed as initial marking(*M0*). Figure 10 shows an example of security test model.

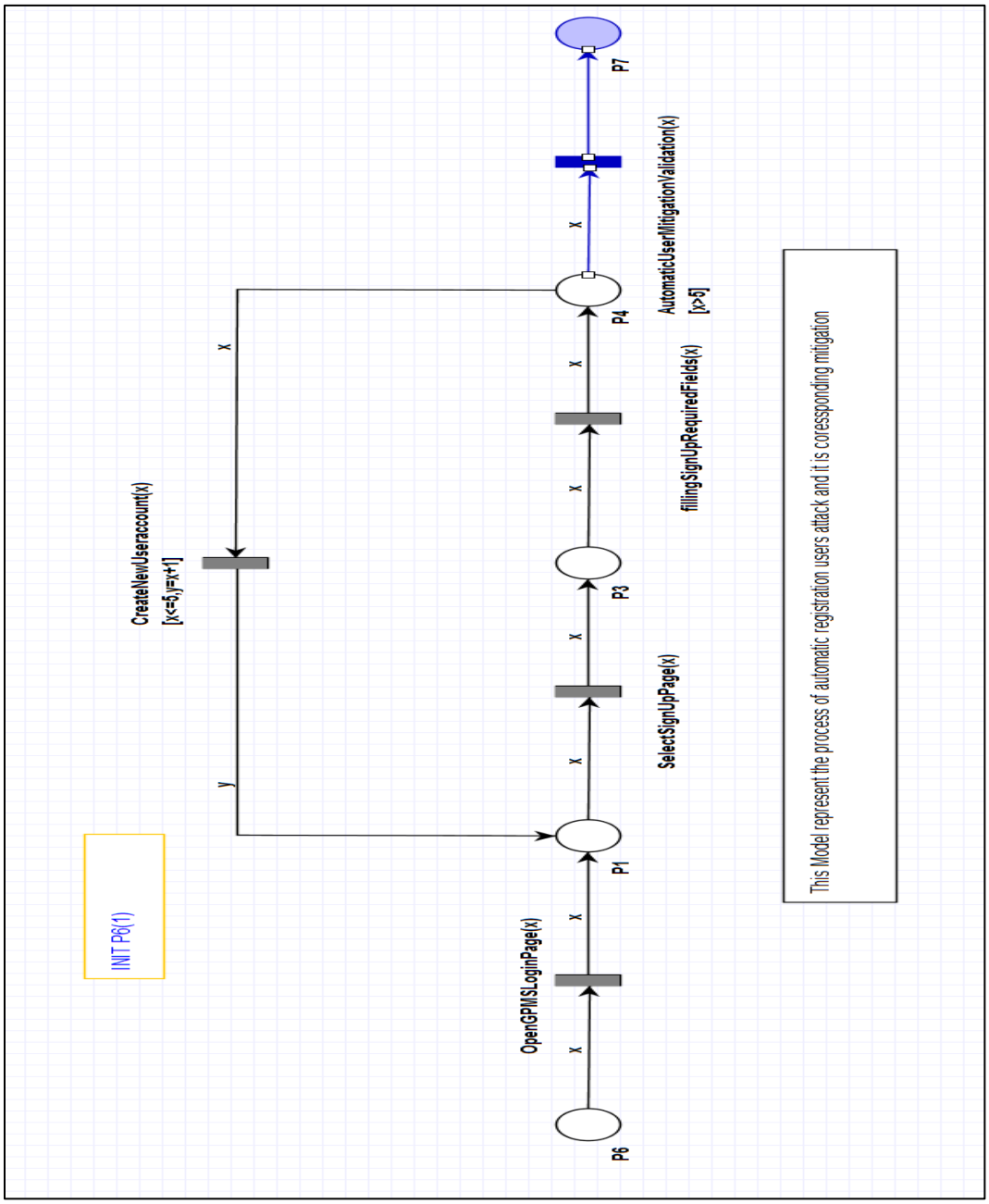

Figure 10 : Security Test Model Example.

3.3 Combine Security Test Models.

After describing the process of extracting security test model from the misuse case textual description which accompanies mitigation use case, it was discovered that each misuse case and the related mitigation use case has a security test model. At this point,

there was a proposal to combine the security test models into one security test model. There are two ways to combine the security test models that have been created. The first method is based on STRIDE and the second is based on the Use Case. The main objectives of combining the resulted security test models were to improve the quality of generated security test cases by reducing the possibility of generating duplicated security test cases and reduce the time and effort needed for the modeling process.

3.3.1 Combine Security Test Models Based on STRIDE

The misuse case textual description has a misuse case category field, representing the misuse case category based on STRIDE methodology. This field has been used to combine the misuse cases that have the same STRIDE category to in one security test model. To combine the misuse cases based on STRIDE, the following steps have been applied:

1. From misuse case category, collect all the misuse cases that have the same STRIDE classification.
2. In one security test model, map each misuse case into PrT construct based on the mapping or modeling techniques discussed in the previous sections. In this step, each misuse case will be modeled to cover all the threaten use cases listed in the threat point of misuse case textual description.
3. Model each mitigation use case for each misuse case based on the mitigation modeling techniques discussed in the previous section.
4. Combine the transitions that perform the same functionality and have the same input and output into one transition “t”. For example, the first and last transition in Figures 11 and 12 are merged into one transition in figure 13.

5. Use arc label enumeration to specify each misuse case with corresponding mitigation use case.
6. Define the initial marking M0 for the security test model. Figure 13 shows an example of combining misuse cases based on STRIDE.

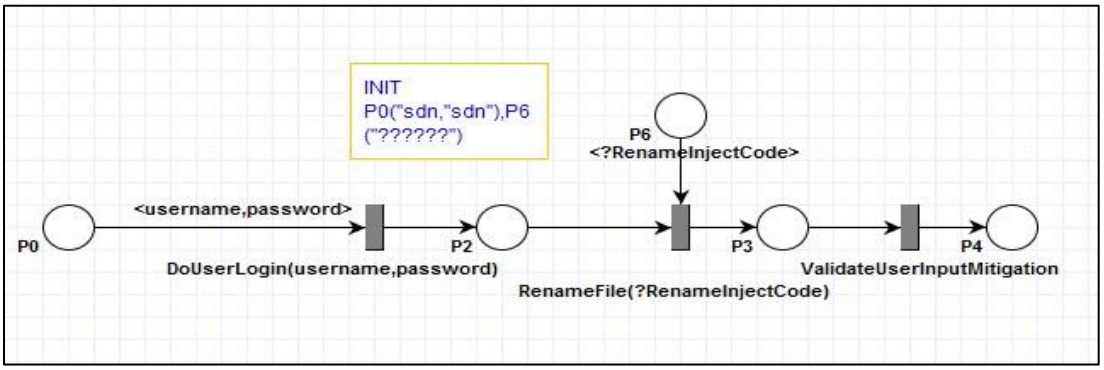

Figure 11: Rename Security Test Model.

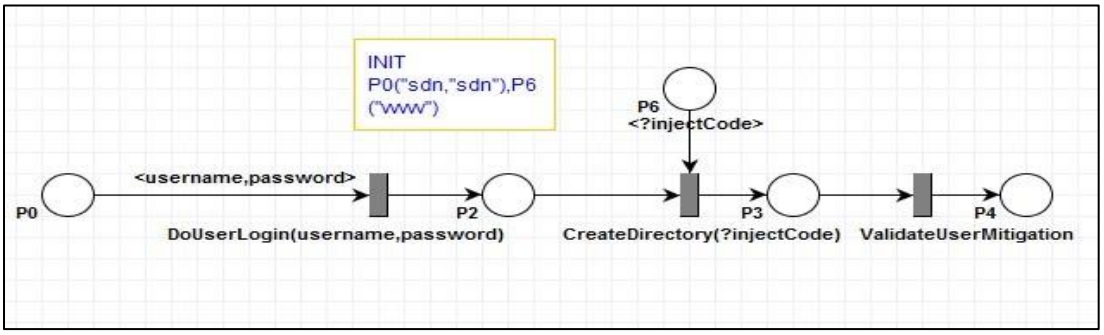

Figure 12: Create Directory Security Test Model.

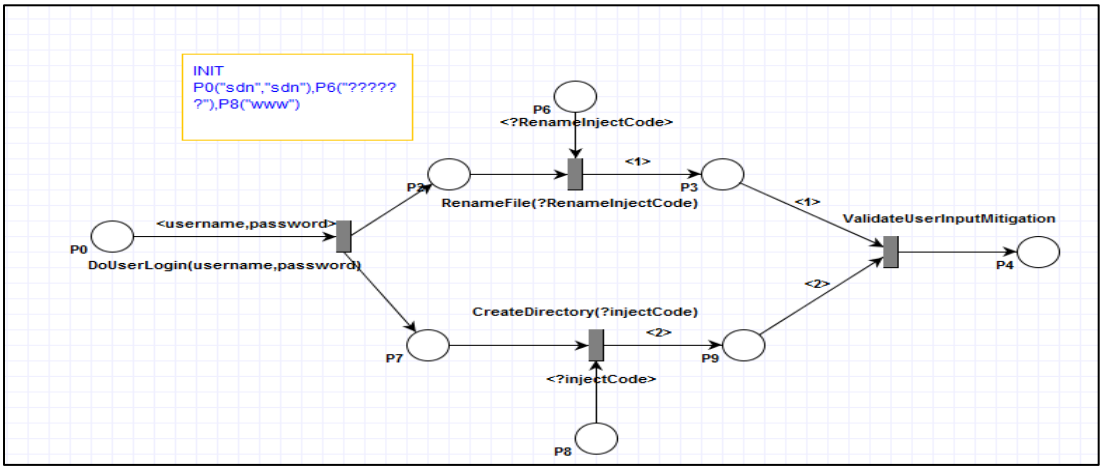

Figure 13: STRIDE Security Test Model Example

3.3.2 Combine Security Test Models Based on Use Case.

Combining the generated security test models based on Use Case represents the second technique of improving the quality of the generated security test models. This technique targets all possible specific functional requirement attacks on the system, represented by misuse cases. To combine all misuse cases that belong to a specific use case, the following steps have been followed:

1. From the threat point of misuse cases textual description, collect all the misuse cases that belong to the targeted use case.
2. In one security test model, model each misuse case based on the mapping or modeling techniques discussed in the previous sections.
3. Combine all the transitions that perform the same functionality in one transition.
4. Model each mitigation use case for each misuse case based on the mitigation modeling techniques discussed in the previous section.
5. Use arc label enumeration to specify each misuse case with corresponding mitigation use case to map each misuse case with related mitigation use cases in the generated security test case.
6. Define the initial marking M0 of the security test model.

Figure 14 shows an example of combining all misuse cases and corresponding mitigation use cases in Creating a New Proposal Document use case. The first 12 transitions are performing the same functionality in each misuse case, so each of them combined into one transition that represents the functionality provided by that transitions. The arc label enumerations have been used to map each misuse case with corresponding mitigation use case in the resulted security test cases.

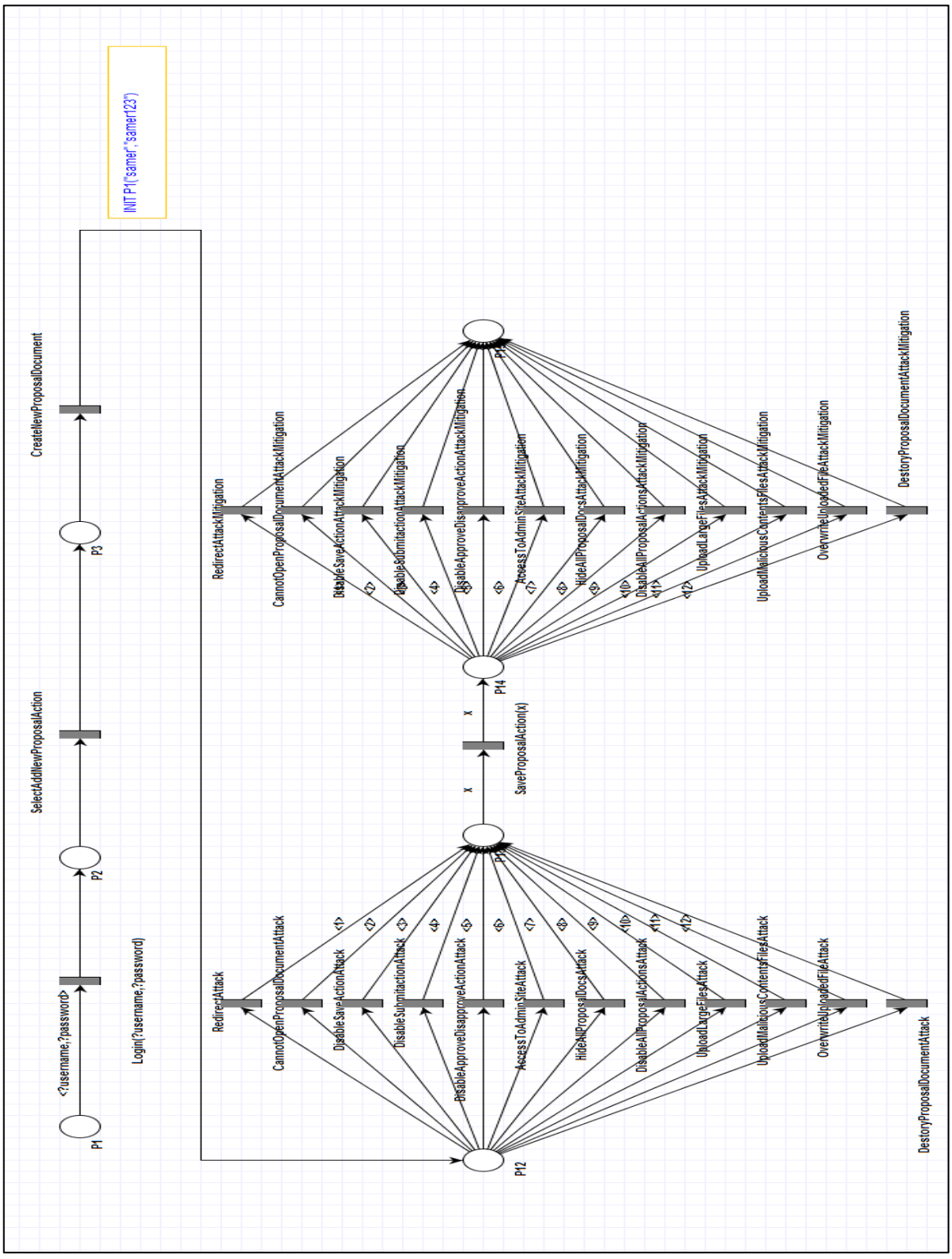

Figure 14: Security Test Model Based On Use Case Example

3.4 Generate Security Test Cases

The MISTA tool is used to generate security test cases from the security test model. The MISTA tool is a Model-Based Integration and System Test Automation. It generates executable test code in different languages (e.g. Java, C, C++, C#, VB, HTML) based on a given Model Implementation Description (MID). MID consist of a test model, Model Implementation Mapping (MIM), and user-provided Helper Code (HC). To create test cases by using MISTA we follow the following steps.

1. Build Test Model. MISTA uses function nets high-level Petri nets as a primary notation for a test model. We use the security test model that has been created in the previous step to be the test model.
2. Create MIM Specification. MIM maps the elements of the test model into implementation constructors for test code generation. MIM include different options elements such as hidden event/conditions, object, methods, options, accessors, and mutators. However, we used object option that maps constants in all token to the objects, the constants we have in the test model like injecting code. We also used methods that map the events/transitions in the test model to a block of code.
3. Create Helper Code. It allows providing additional code to make the generated test code executable. HC includes different options such as header code, teardown, and setup etc. We used helper code setup option to initialize the instance variables before the test cases start and we used header code to include the needed header files and other global variables. After fulfillment the previous steps, we created security test cases in C language for FileZilla server and security test cases in Java

for GPMS system by using reachability tree coverage. Figure 15 shows an example of generated security test case for FileZilla server in C and Figure 16 shows an example of security test case in Java for GPMS system.

```
void test3() {
	printf("Test case 3\n");
	setUp();
	/* Create Directory Code Block*/
	int CreateDirMitigationEffect = FTPMkdir(&connInfo, remoteDir, kRecursiveNo);
	/* Create Directory Code Block*/
	/* Create Directories Mitigation Effect Code Block*/
	if (CreateDirMitigationEffect < 0) {
		FTPPerror(&connInfo, CreateDirMitigationEffect, kErrCouldNotStartDataTransfer, "Could not get ",FTPStrError(CreateDirMitigationEffect));
			es = 1;
		} else {
			es = 0;
		}
	/* Create Directories Mitigation Effect Code Block*/
}

void test4() {
	printf("Test case 4\n");
```

Figure 15: A Sample FileZilla Server Security Test Case.

```
package MISTA_SecurityTestCases;

import junit.framework.TestCase;

public class CreateNewProposalAttackTester_RT extends TestCase{

    WebDriver driver = new FirefoxDriver();

    @Test
    public void test1() throws Exception {
        System.out.println("Test case 1");
        SeleniumUtilities.userLoginToTheSystem(driver,"samer","samer123");
        SeleniumUtilities.createEmptyProposalDocument(driver);
        SeleniumUtilities.fillProjectInformationSectionFields(driver,"Redirect Attack MISTA tool test A");
        SeleniumUtilities.fillSponsorAndBudgetInformationSectionFields(driver);
        SeleniumUtilities.fillCostShareInformationSectionFields(driver);
        SeleniumUtilities.fillUniversityCommitmentsSectionFields(driver);
        SeleniumUtilities.fillconflictOfInterestAndCommitmentInformationSectionFields(driver);
        SeleniumUtilities.fillComplianceInformationSectionFields(driver);
        SeleniumUtilities.fillAdditionalInformationSectionFields(driver);
        SeleniumUtilities.fillCollaborationInformationSectionFields(driver);
        SeleniumUtilities.fillProprietaryAndConfidentialInformationSectionFields(driver);
        SeleniumUtilities.fillCertificationAndSignatureSectionFields(driver,GPMSAttacksSourceCode.getXXSScriptRedirectAttack());
        SeleniumUtilities.saveProposalDocument(driver);
        SeleniumUtilities.redirectAttackMitigation(driver);
    }
```

Figure 16: A Sample GPMS Security Test Case.

CHAPTER FOUR CASE STUDY I: FILEZILLA FTP SERVER

In order to validate the applicability of the proposed approach, two case studies have been conducted: FileZilla FTP server, and GPMS. The two applications have different business logic, user and system requirements, and programming languages.

File Transfer Protocol (FTP) is a standard protocol, used widely with remote computer systems and transferring files between systems. FileZilla FTP server is a popular FTP server implementation which, as of April 2016, is the seventh most downloadable program on Source Forge [21]. FileZilla server 0.9.53 has 90,653 line of C++ code and 123 classes used in this case study.

4.1 Misuse Case Modeling

4.1.1 Defining FileZilla FTP Server Use Cases

FileZilla server documentation, comprehensive understating of the FileZilla server source code, and the full documentation of FTP specification in the Request for Comment (RFC) published by internet Engineering Task Force (IETF) are considered the major source of defining the use cases of FileZilla FTP server. FileZilla FTP server offers all FTP operations such as download files, upload files, create a directory, login…etc. In addition, FileZilla server offers administrative services. After defining the use cases, a textual description for each identified use case has been created. Table 8 is an example of use case textual description. Then the use case diagram is created, shown in Figure 17.

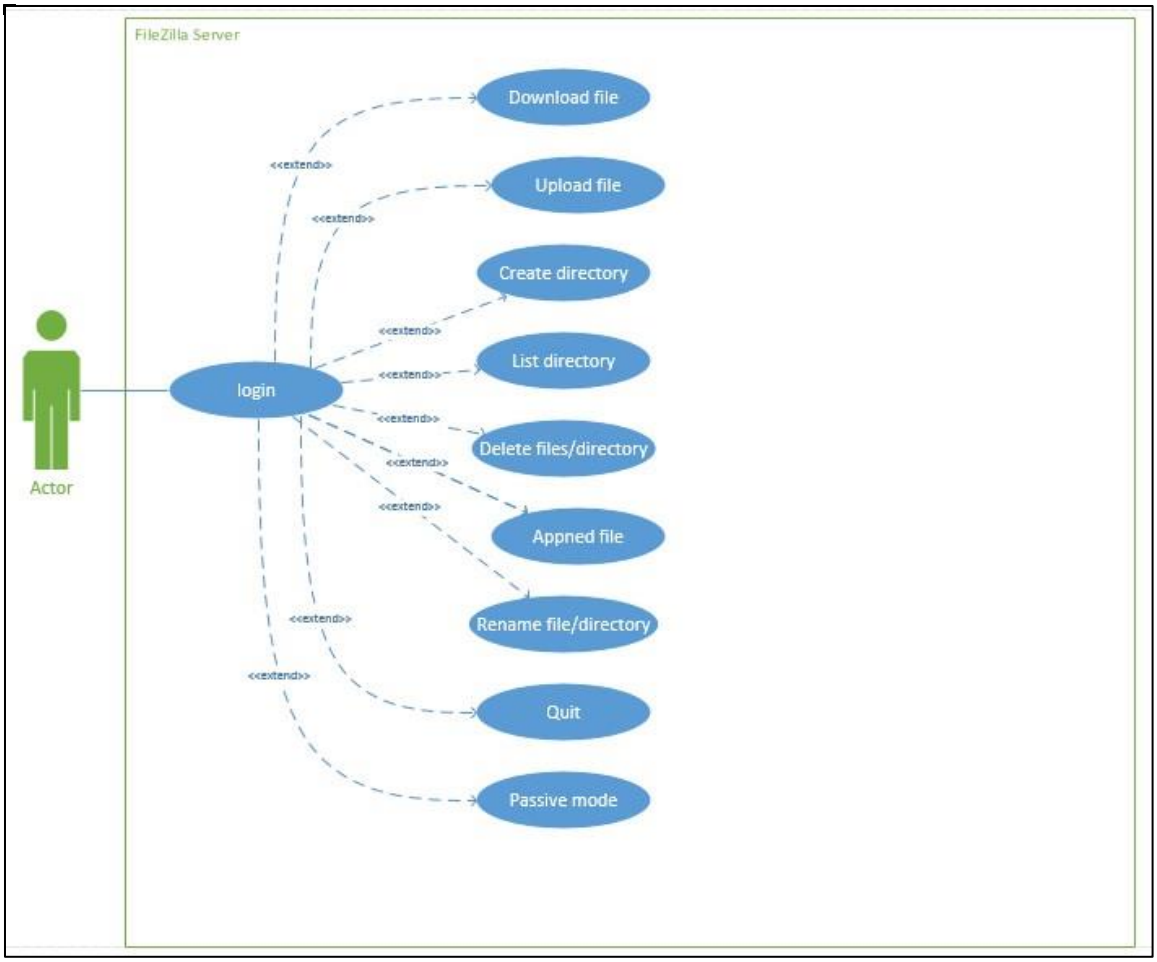

Figure 17: FileZilla Server Use Case Diagram.

Table 8: FileZilla FTP Server Delete Files Use Case Textual Description.

Use case #	**UC2**
Use case name	Delete files and/or Directories.
Actors	User, System.
Goal	To delete a file or/and directories from the client machine to FTP server.
Preconditions	1. The system is up and running. 2. The actor logged into the system.
Main Flow	1. The actor provides the "DELE" command with remote destination folder/directory path and its name by using command line tool. 2. The system receives the "DELE" command request. 3. The system validates the destination file path**.** 4. The system deletes the file or the directory.

	5. The system provides the actor a summary of file/directory deleting process. 6. The system records the request in the log file.
Post-conditions	1. The file or directory is deleted successfully.
Extension Point	NONE
Alternative Flow	**1a. The actor uses GUI client application.** 1.a.1. The actor selects the file/directory that wishing to delete in the remote destination folder. 1.a.2. The actor deletes the file or directory from the available actions provided from file or directory properties. The use case continuous at System validates source file path in the MF.
Exception Flow	**3.a. The file name does not exist.** 3.a.2 The system throw error message telling the client that the file does not exist.
Recovery Flow	**4a. The destination path does not exist or invalid file/folder name.** 4. a.1 The system notifies the actor that the destination does not exist. The use case continuous at the actor provides "DELE" command with existing destination folder path or with valid file/folder name in the MF.

4.1.2 Defining FileZilla FTP Server Misuse Cases

Misuse cases are defined in examining all of the security goals and each STRIDE type of every step in the main flow and alternative flow for each use case. Table 9 shows the misuse cases created corresponding to STRIDE. After defining misuse cases, a use case/misuse case diagram has been created and is shown in Figures 18. After that, a textual description for each misuse case has been created. Table 10 is an example of the misuse case textual description.

Table 9: Misuse Cases Corresponding to STRIDE

Misuse case name						
Inject malicious code						

Unauthorized modified/ access data						
Crack user passwords						
Overflow login table misuse case						

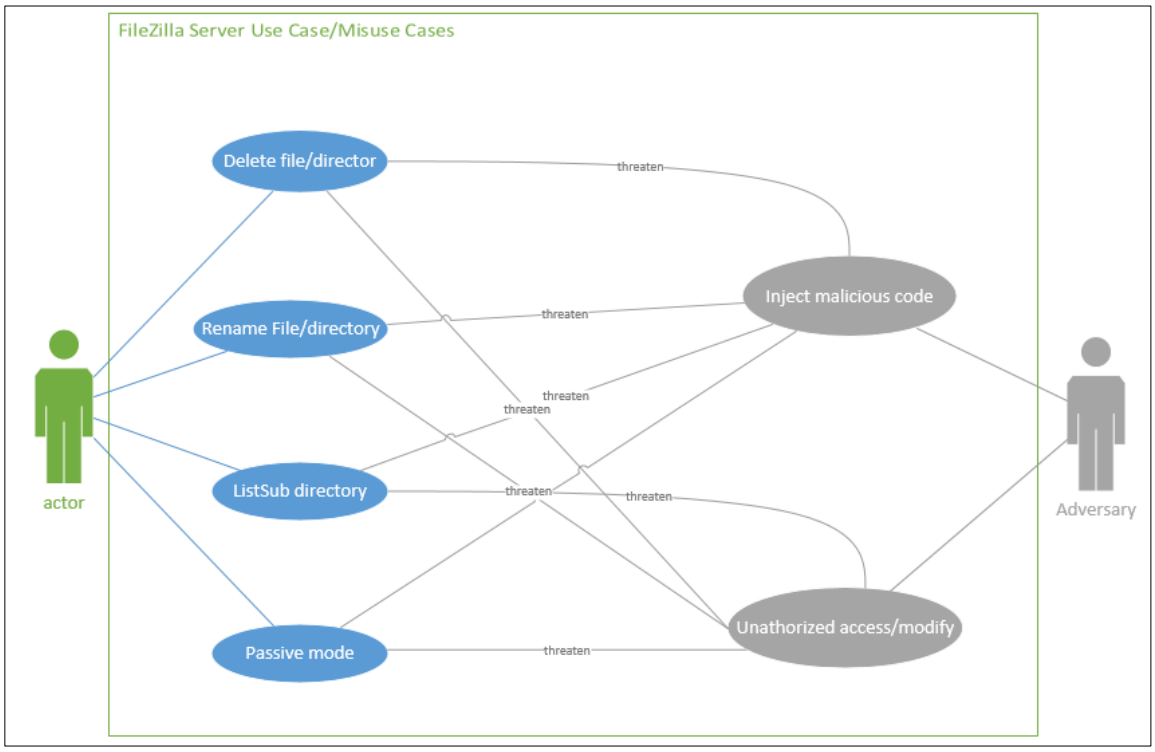

Figure 18 : FileZilla FTP Server Use Case/Misuse Case Diagram

Table 10: Inject Malicious Code Misuse Case Description

Misuse case#	**MUC 2**
Misuse case name	Injecting malicious code Attack
Misuse case category	Denial of service attack
Goal	To disturb the system services
Actor	Adversary
Preconditions	1. The attacker has an account on the system.
Main Flow	1. The attacker anonymously login to the system. 2. The attacker issues MKD command with injecting malicious code.
Post-conditions	The system will be crashed or suspended.

Threat point	1. Rename File, Main flow, Step 1, the user sends the RNTO command, step 2, the system receives the command and process it.
Mitigation	Validate user input mitigation use case

4.1.3 Defining FileZilla FTP Server Mitigation Use Cases

For each misuse case, the proper mitigation use cases have been identified by using STRIDE and other security references such as OWASP. A textual description has been created for each. Table 11 is an example of mitigation use case textual description. Figure 19 shows an example of FileZilla FTP Server use cases/misuse cases/mitigation use cases diagram.

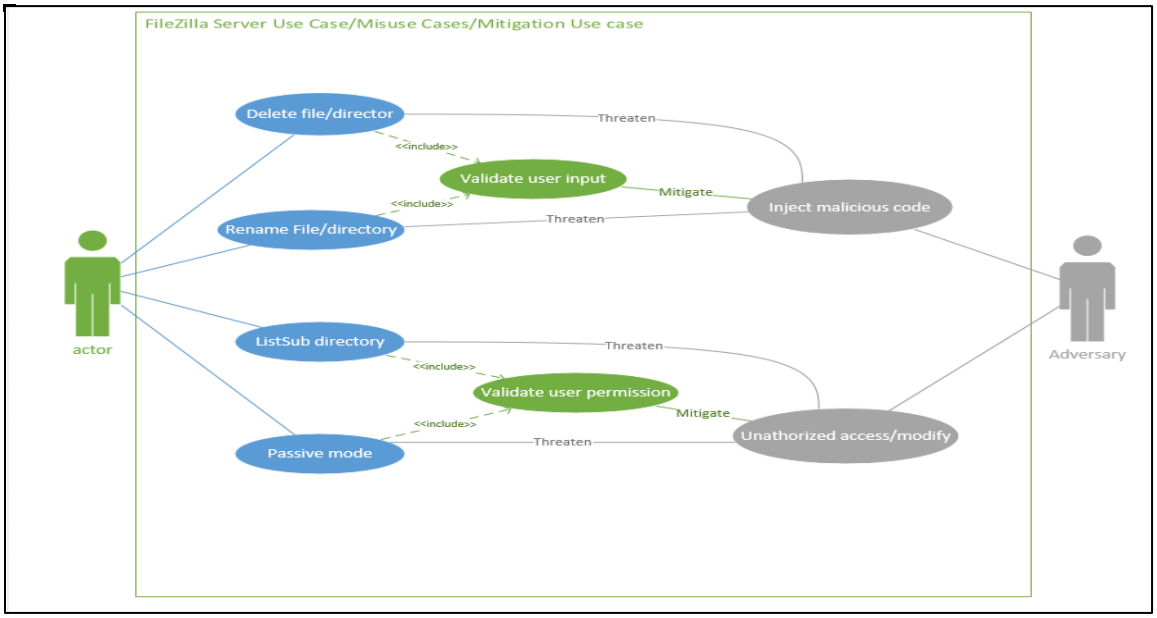

Figure 19: FileZilla Server Use Case/Misuse Case/Mitigation Use Case Diagram.

Table 11: Validate User Permission Mitigation Use Case Description.

Mitigation use case #	**MITI-UC 2**
Mitigation use case name	Validate user permission
Goal	To validate the user permission before access and/or modifying data.

Precondition	The actor login to the system as an anonymous or legitimate user.
Main Flow	1. Retrieve the actor permission. 2. Read the actor permission for the requesting command. 3. Return the right actor permission for the requesting command. 4. Based on the returned result the system will throw error
Post Condition	Read and validate the right the permission for the actor.
Priority	High

4.2 Create Security Test Model

Security test models have been extracted based on the textual description of misuse cases accompanying mitigation use cases by using the extraction techniques that have been discussed in the previous sections of chapter 3. Security test models have been combined by using the STRIDE and Use Case combination techniques that were discussed previously. Figure 20 shows an example of a STRIDE combination and Figure 21 shows Use Case.

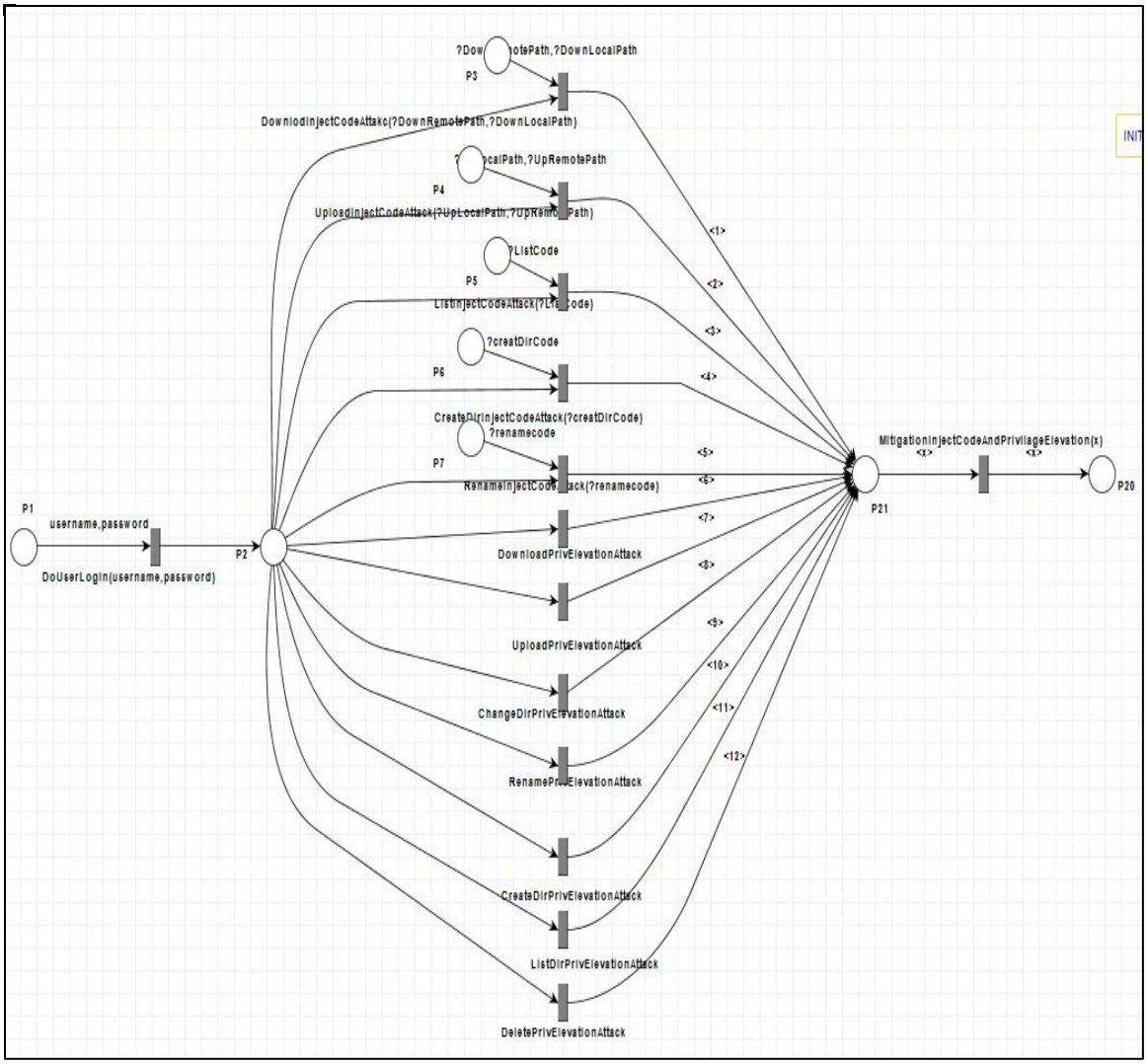

Figure 20: Security Test Model Based on STRIDE

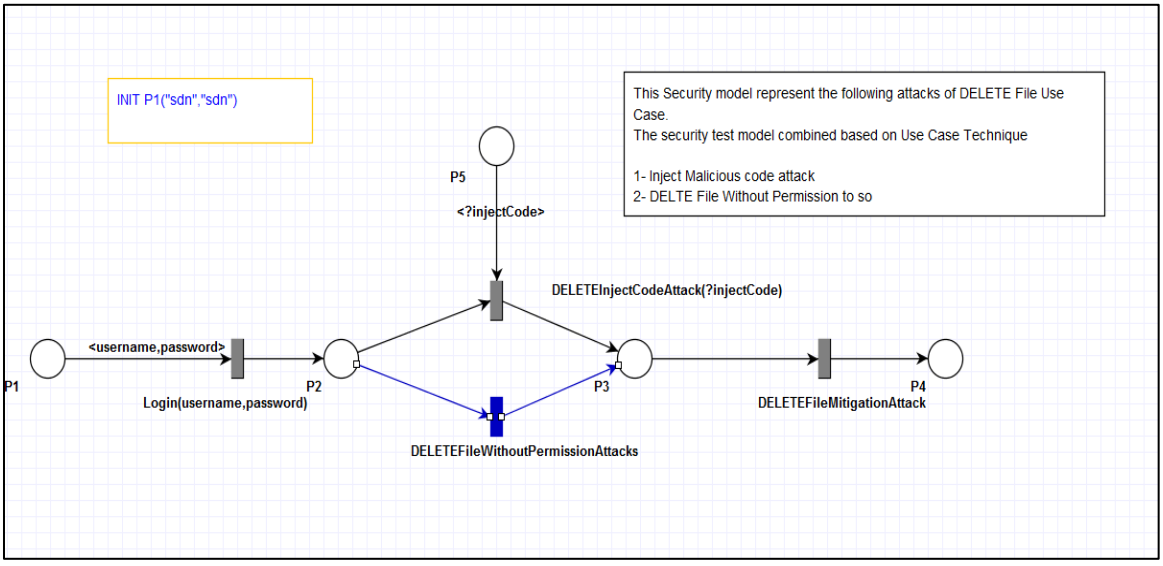

Figure 21: Security Test Model Based on Use Case.

4.3 Generate Security Test Cases

FileZilla FTP server security test cases have been generated based on the extracted security test models by using MISTA tool. MISTA tool generates test cases based on the given MID. A MID has been created which consists of the test model, MIM, and HC. After implementing all of these, MISTA generates test cases. Figure 16 gives an example of these generated test cases.

```
void test1() {
   printf("Test case 1\n");
   setUp();
   /* Login Code Block. */
   if(resultconn !=0){
   InitWinsock();
   result = FTPInitLibrary(&libInfo);
   if (result < 0) {
            fprintf(stderr, "simpleget: init library error %d (%s).\n", result, FTPStrError(result));
            DisposeWinsock();exit(4);}
            result = FTPInitConnectionInfo(&libInfo, &connInfo, kDefaultFTPBufSize);
       if (result < 0) {
            fprintf(stderr, "simpleget: init connection info error %d (%s).\n", result, FTPStrError(result));
            DisposeWinsock();exit(3);
       }
        strncpy(connInfo.host, remoteHost, sizeof(connInfo.host) - 1);
        strncpy(connInfo.user, "sdn", sizeof(connInfo.user) - 1);
        strncpy(connInfo.pass, "sdn", sizeof(connInfo.pass) - 1);
       connInfo.debugLog = stdout; resultconn = FTPOpenHost(&connInfo);
   }
   /* Login Code Block */
   FTPCmd(&connInfo, "RMD %s", DeleRemoteDir);
   /* Mitigation Effect Code Block*/
   if (mitigationResult < 0) {
       FTPPerror(&connInfo, mitigationResult, kErrCouldNotStartDataTransfer, "Could not get ",
       FTPStrError(mitigationResult));es = 1;} else {es = 0;}
   /* Mitigation Effect Code Block*/
}
```

Figure 22: FileZilla FTP Server Security Test Case.

4.4 Evaluation of Security Test Cases.

After applying the presented approach, 11 use cases have been identified, covering the majority of the FileZilla FTP server functionalities. Table 12 shows; the use cases, the number of misuse cases, and the number of security test cases have been generated for each use case.

In the first step, security test cases were executed against the FileZilla. In order to further the evaluation of the security test cases, we used a mutation testing method. Mutation testing is a method of injecting faults into original source code of the software

system to test whether the test cases can find the injected faults or not. A mutant is a version of software source code with injected faults. 38 security mutants were created using common vulnerabilities in C++ and security problems with FTP. This process is shown in Table 13.

The vulnerability is revealed; a mutant is said to be killed if one of the security test cases successfully attack. Security test cases were executed against the created mutants, killing 33 out of 38 mutants. The five remaining mutants have logical errors in the administration functions provided by FileZilla server that could lead to different attacks such as DoS. The behavior of these security vulnerabilities is not included in the misuse case modeling. Table 14 shows the results of performing the security test cases against the mutants. These results show the mutant name, the mutant description, the STRIDE category, the expected result, and the actual results of executing the security test cases.

Table 12: FileZilla FTP Server Use Case, Misuse Cases, and Test Cases

Use case	Number of misuse cases	Number of test cases
Download file	2	2
Create Directory	2	2
Upload file	2	2
Delete file/directory	2	2
Rename files/directory	2	2
List Directory	2	2
List Subdirectories	2	2
Append file	2	2
Logout	2	2
Login	2	2
Change to passive mode	2	2
Total	**22**	**22**

Table 13: FileZilla Server Security Mutants

Vulnerability type	Number of mutants	Number of mutants killed
Buffer overflow	3	3
Logic errors	8	8
Password management errors	2	2
Memory leak	2	2
Format String	2	2
Integer overflow	1	1
Incorrect access control	14	11
Business logic flaws	6	4
Total	38	33

Table 14: Security Test Cases Execution Results

Mutant	Mutant Description	STRIDE Category	Mutant Type	Expected Result	Actual Result
Mutant 1	Create directory without permission.	T & E	Incorrect access control	Cannot create the directory.	Directory created successfully
Mutant 2	Delete directory without permission	T & E	Incorrect access control	Cannot delete directory	Directory deleted successfully
Mutant 4	Delete File without permission	T & E	Incorrect access control	Cannot delete file	File deleted successfully
Mutant 5	Upload file without permission	T &E	Incorrect access control	Cannot upload file	File uploaded successfully

Mutant 6	Rename File/Dir without permission	T &E	Incorrect access control	Cannot rename file/Dir	File/Dir renamed successfully
Mutant 7	Banned IP can still log in	E	Logic errors	IP will be banned for a certain amount of time	User can login
Mutant 8	Download file without permission	I & E	Incorrect access control	Cannot download file	File downloaded successfully
Mutant 9	No password needed for login	S & E	Logic errors	Cannot login	User Login successfully
Mutant 10	Show user passwords in clear text	S & R & I	Password management errors	Password should willed be starred (*)	Password printed in clear text on the server interface
Mutant 11	No logs kept even if logging enabled	R	Business Logic errors	User login should be Logged	No logs kept
Mutant 12	Filtered IP can still log in	E	Incorrect access control	Cannot login	User login successfully
Mutant 13	Locked Server can be accessed	E	Incorrect access control	User cannot log in	User login successfully

Mutant 14	A user with no list permissions logging on causes FTP service to quit when it gets to listing the directory with the LIST command.	D	Incorrect access control	Server stay running and does not list directories	Server crashed
Mutant 15	If the user tries to delete directory all files inside the subdirectories are deleted but none of the directories are deleted.	T &E	Incorrect access control	Cannot delete files and directory	Files and directories are deleted successfully
Mutant 16	A user with no permissions can delete directories that are empty and directories that are full of files and other directories.	T &E	Incorrect access control	Cannot delete files and directory	Files and directories are deleted successfully
Mutant 17	Log out the login user	D	Logic errors	User can login and perform commands	User login to the server but immediately disconnected from the server
Mutant 18	Append File without permission	T & E	Incorrect access control	User cannot append file	File appended

					successfully
Mutant 19	User can subdirectories without permissions	I & E	Incorrect access control	User cannot subdirectories	User Successfully subdirectories
Mutant 20	a user with Force SSL checked can still log on without using SSL authentication	E	Incorrect access control	Cannot log in without SSL error message	User Successfully login
Mutant 21	Spoofs the first user created on server with any credentials except a blank username	S	Logic errors	Cannot login error message	User Successfully login
Mutant 22	Spoofs first user created on server with any credentials	S	Logic errors	Cannot login error message	User Successfully login
Mutant 23	No user or group settings are saved.	D	Business Logic errors	User successfully logs in	User Cannot login, error message
Mutant 26	Denial of Service for the QUIT command. Users can't log off using QUIT command.	D	Business Logic errors.	User is disconnected	Successful QUIT message but user is still connected to the server
Mutant 27	Denial of Service for the PASV command.	D	Logic errors.	User successfully enters passive mode.	Cannot enter passive mode, error message.
Mutant 29	QUIT command is issued a memory leak is caused.	D	Memory leak	The memory of the process will go down when QUIT command is issued.	The allocated memory of the process will stay the same when QUIT command is issued

Mutant 30	Denial of Service for the CWD command.	D	Logic errors.	User successfully changes directory	Cannot change directory, and the server disconnect the connection
Mutant 31	Format String Error that causes the server to crash when the LIST command.	D	Format String	The user gets a listing of files and directories.	Server Crashed
Mutant 33	No file log will be deleted using the configuration settings to set how big the collective files can get or to get rid of old log files by date.	D	Logic errors.	Some logs were deleted.	No logs were deleted
Mutant 34	Memory leak is created when the PASV.	D	Memory leak	The memory of the process will go down when a new PASV command is issued.	Memory of the process will goes up when a PASV command is issued
Mutant 35	The user can issue many consecutive commands causing the server to be slow to respond to other commands.	D	Business Logic errors.	Server builds up a wait time for your next login.	Server gets bogged down from multiple login or commands.
Mutant 36	Integer overflow causes an error that won't allow the user to change to passive mode.	D	Integer overflow	User successfully enters passive mode.	Cannot enter passive mode error message.

Mutant 37	The error causes the user to be disconnected from the server on any function that causes a transfer socket to be created.	D	Format String	User successfully uploads a file.	User is disconnected or server crashes
Mutant 38	Buffer overflow causes the server to crash after logins.	D	Buffer overflow	User successfully logs in	Server crashes

CHAPTER FIVE CASED STUDY II: GRANT PROPOSAL MANAGEMENT SYSTEM (GPMS)

GPMS application is a web-based workflow management application for replacing the manual approval process for grant proposal submission. The process of creating a research proposal, submitting and tracking the proposal status is time-consuming. During the process of submission until the final approval decision, many parties can become involved which raises a delegation of authority problem which required a change of security policies. GPMS provides a fine-grained control over the security policies by separating the workflow of the program from the access control policies.

The GPMS system has been developed as a research project that uses the XCAML standard language for specifying access control policies in software applications. It is implemented in Java language with, JSON API, JSP, web services API, and Mongo database. Currently, GPMS has 24,128 lines of java code and 84 classes.

5.1 Misuse Case Modeling

5.1.1 Defining GPMS Use Cases

GPMS documentations, comprehensive understanding of the source code, user experience, and team discussions represent the major sources of defining GPMS use cases. GPMS provides a different type of functionalities such as; create new proposal document, create new users, delete, submit, archive proposals based on the user permission, providing user notification, export to excel, administration services such as managing proposals, and managing users…etc. After defining the use cases, the textual description of each has been

created. Table 15 is an example of these use case textual descriptions. Figure 23 is an example of the use case diagram that has been created.

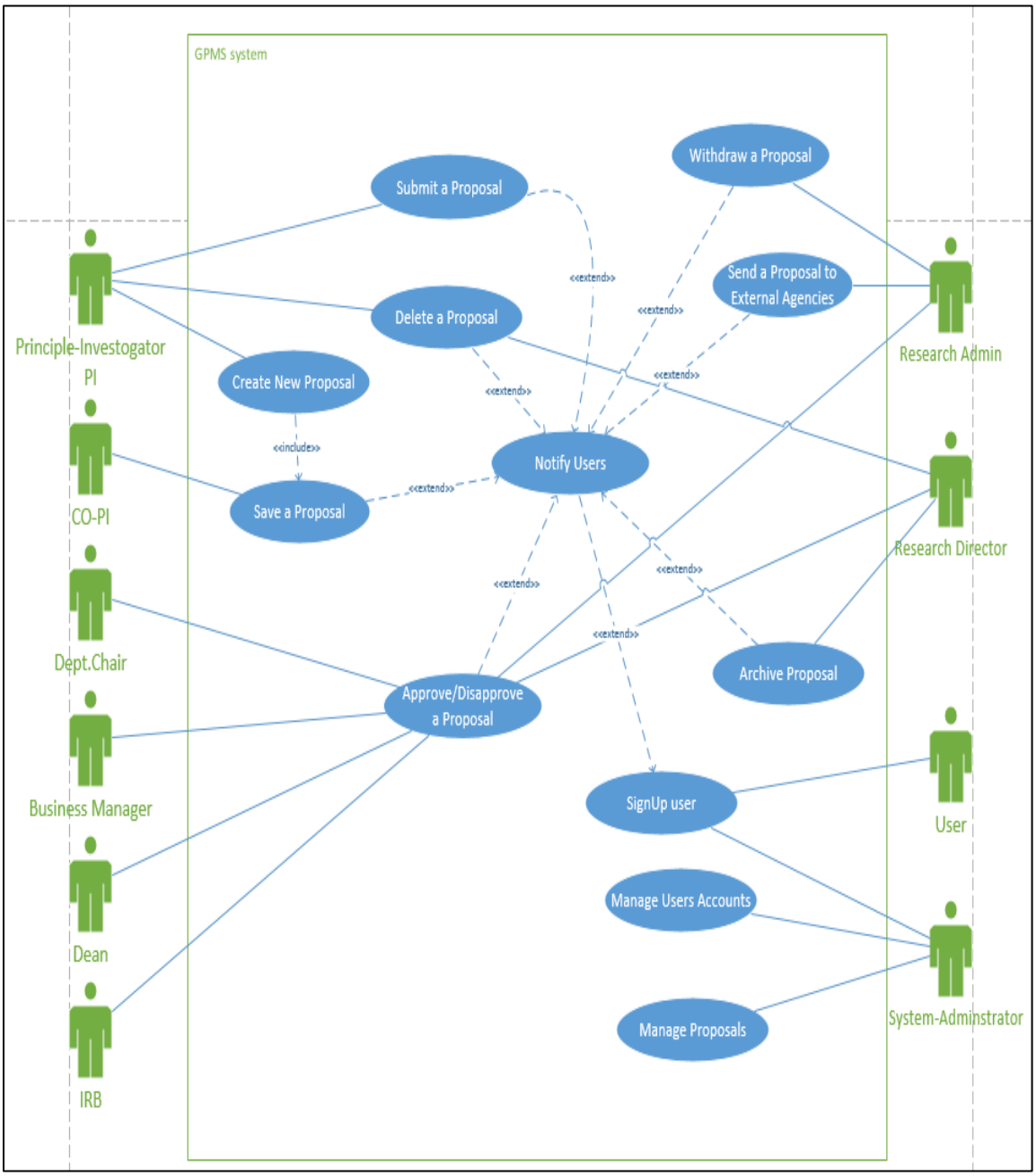

Figure 23: GPMS Use Case Diagram.

Table 15: Approve a Proposal by Business Manager Use Case Description.

Use case #	UC-3
Use case name	Approve a Proposal by Business Manager
Actor	Business Manager.
Goal	To approve the proposal by the business manager.
Preconditions	1. The proposal status is ready for business manager approval.
Main Flow	1. The actor is logged in. 2. The actor selects "My proposal" action. 3. The actor selects the proposal by selecting "edit" action. 4. The actor can edit the "Sponsor and Budget Information" section in the proposal. 5. The actor signs the proposal by filling the signature, date, and note fields. 6. The actor approves or disapproves the proposal by selecting approve/disapprove action. 7. **If** the actor approves the proposal, the system will send notifications to the PI, Co-PI, IRB and the Dean, **Else**, the system will send a notification to system sends an email to PI, Co-PI, Department chair, IRB, and all Business Managers. 8. The system updates the proposal status and saves it. 9. The system sends a confirmation message. 10. The system records that on the user audit log. 11. The system records in the system log.
Post-Condition	1. If the actor approved the proposal, the proposal status will change to ready for Dean's approval. 2. If the actor disapproves the proposal, the proposal status will change to not submitted.
Extension Points	1. Step 7, Notify Users use case.
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor opens check the notification tab 2. a.2 The actor selects the proposal. The use case continuous at *The actor approves/disapproves the proposal in MF.*
Exception flow	NONE
Recovery flow	NONE

5.1.2 Defining GPMS Misuse Cases

GPMS misuse use cases have been defined by applying STRIDE methodology and security goals to each step in the use case main flow and alternative flow. Table 16 illustrates the misuse cases types corresponding to STRIDE. After defining the misuse cases, the textual description has been created for each of the defined misuse cases. Table 17 is an example of misuse case textual description. Figure 24 demonstrates the use cases/misuse cases diagram that has been created.

Table 16: Misuse Cases STRIDE Classifications.

Misuse Case Type	S	T	R	I	D	E
Authentication & session management	*				*	
Cross-Site scripting (XSS) & Cross-Site request forgery (CSRF).		*		*	*	*
Injecting Malicious Code	*	*		*		*
Access Control Horizontal Privilege Escalation.		*				*
Access Control Abusing Workflow.					*	*
File Path Injection.				*	*	*
Upload Dangerous Content.		*				*
Overwrite Other Files.		*				*
Quota Overload Denial of Service DoS.					*	

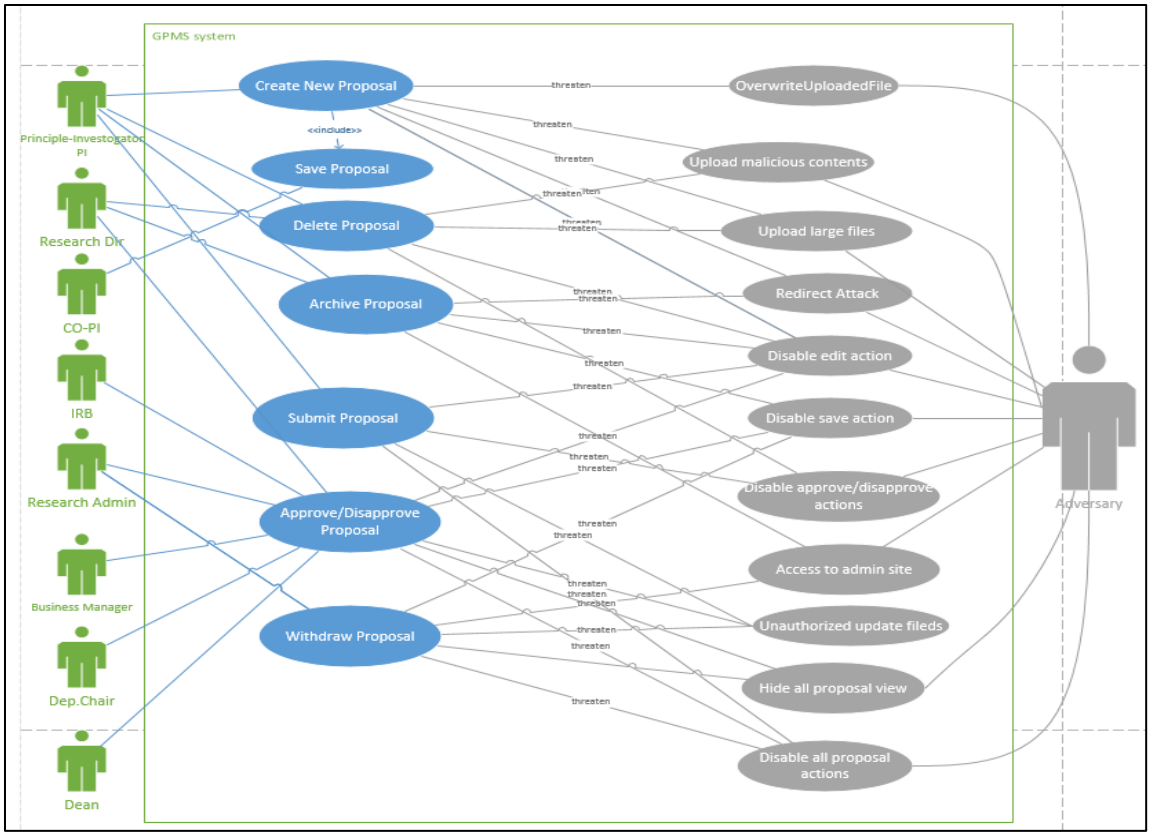

Figure 24: GPMS Use Case/Misuse Case Diagram

Table 17: Disable Submit Action Misuse Case Description

Misuse case #	**MUC-4**
Misuse case name	Disable Submit Action Attack.
Misuse case category	Information disclosure, DoS and elevation of privilege.
Goal	To disable the proposal submission functionality.
Actor	Adversary
Preconditions	1. The actor has an account on the system or has other user login information.
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor selects a proposal and opens it by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions.
Post-conditions	1. The disable action will be disabled. 2. The actor cannot submit a proposal.

Threat points	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 5. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 6. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 7. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

5.1.3 Defining GPMS Mitigation Use Cases

Mitigation use cases have been defined by STRIDE, Open Web Application Security Project (OWASP), and other security techniques. After defining the proper mitigation use cases, a textual description has been created for each of these mitigation use cases. Table 18 is an example of the GPMS mitigation use case textual description. Use cases/misuse cases/mitigation use cases diagram has also been created, shown in Figure 25.

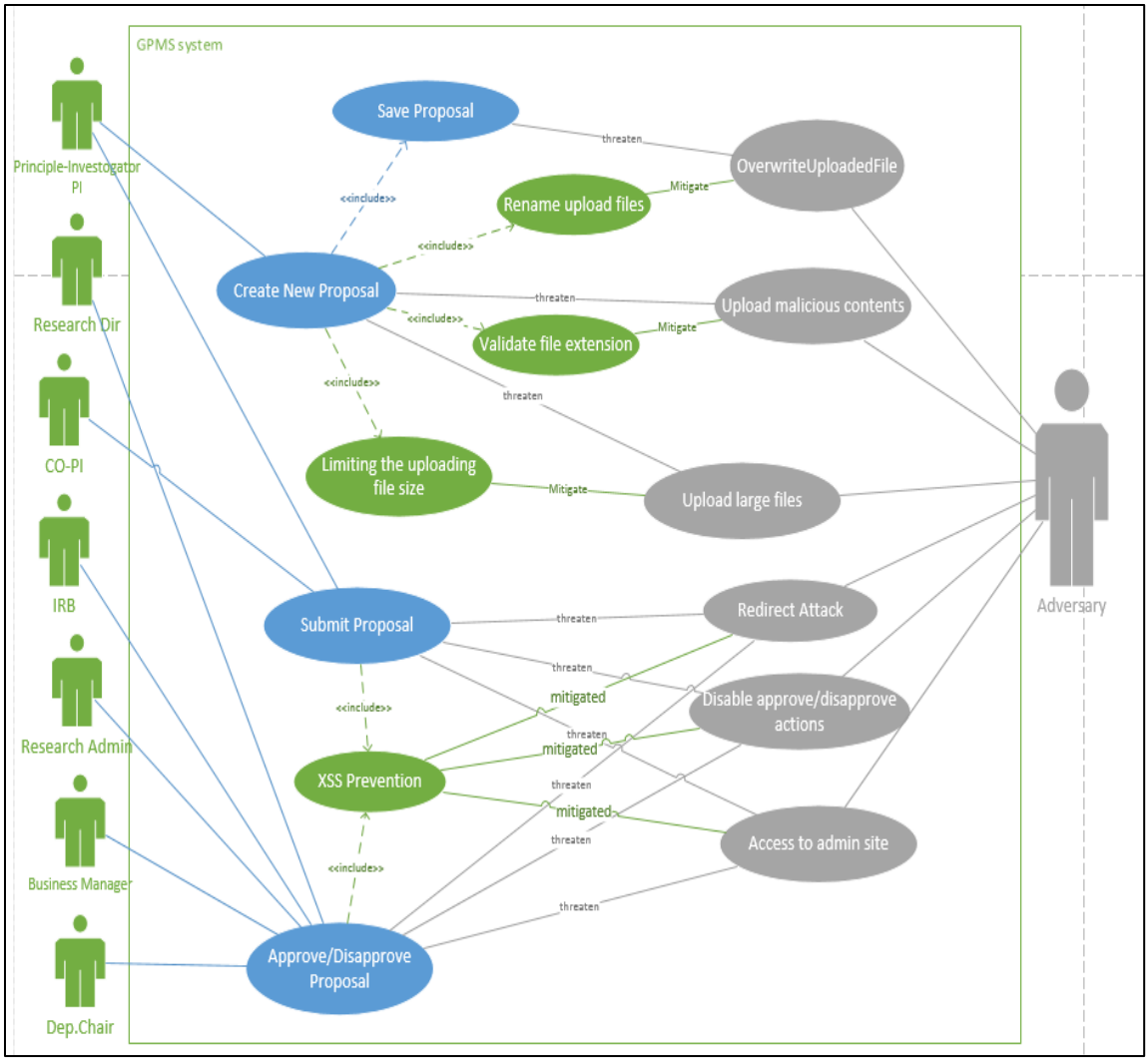

Figure 25: GPMS Use Case/Misuse Case/Mitigation Use Case Diagram

Table 18: GPMS Mitigation Use Case Textual Description

Mitigation use case #	**MITI-UC 2**
Mitigation use case name	Validating file extensions and contents.
Goal	To prevent uploading dangerous files to the system.
Precondition	1. The actor upload files to the system.
Main Flow	1. Validate the uploading file extension by using whitelist technique. 2. Validate the contents of the uploaded file by using antivirus.

Post Condition	1. The application prevents uploading files with a malicious extension such as. JSP. 2. The application prevents uploading files with malicious contents with the correct extension.
Priority	High

5.2 Create Security Test Model

After defining the GPMS misuse cases and proper mitigation use cases, security test models have been extracted by using the techniques discussed in chapter 3. Security test models have also been combined based on STRIDE and Use Case. Figure 26 is an example of a security test model based on STRIDE. Figure 27 is an example of security test model based on Use Case.

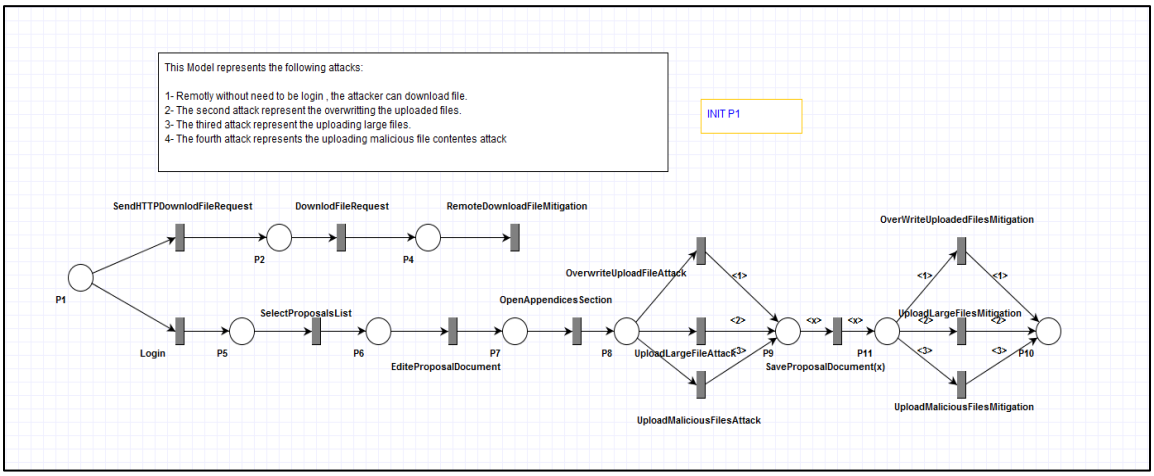

Figure 26: GPMS Security Test Model Based on STRIDE.

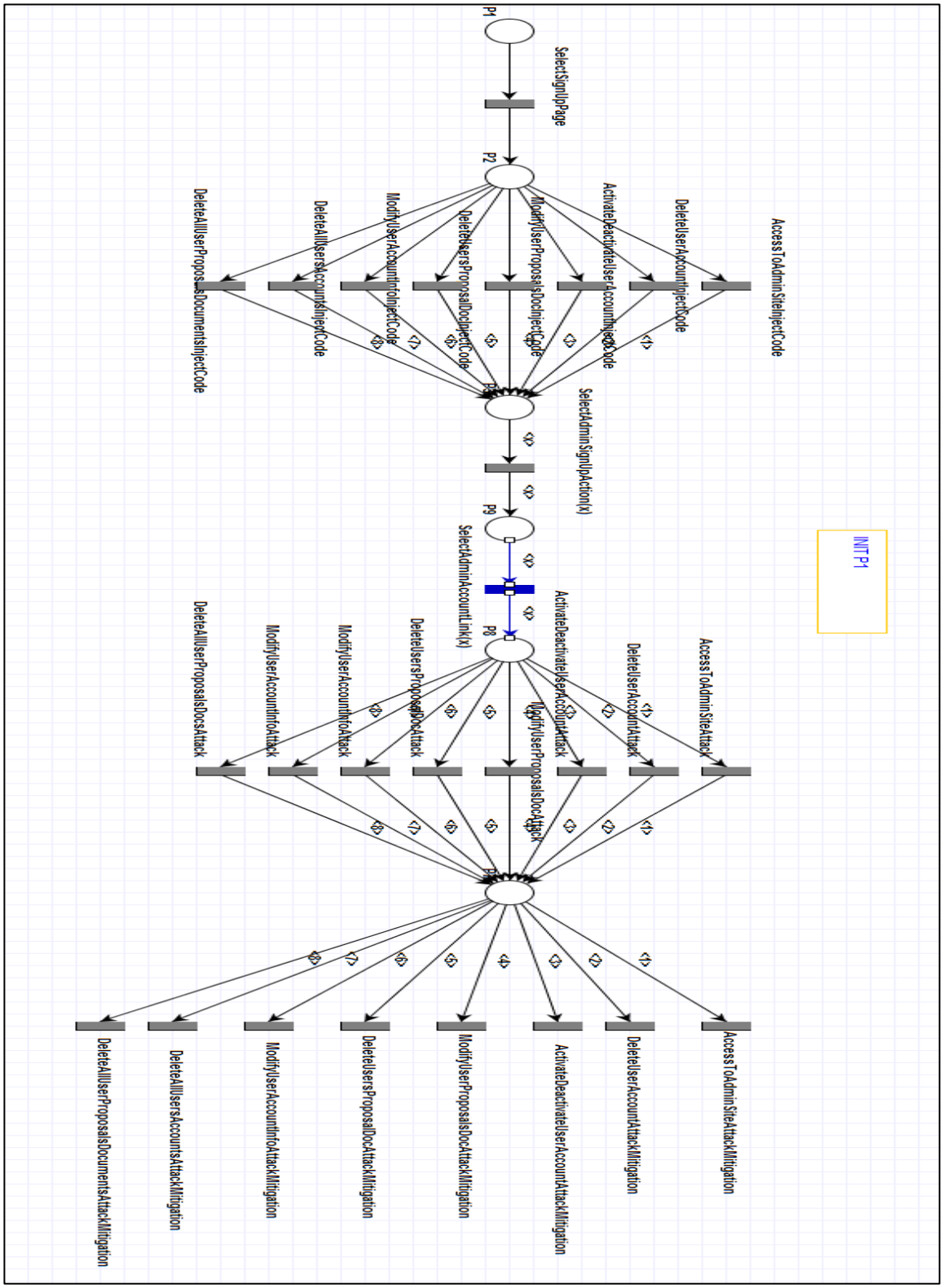

Figure 27: GPMS Security Test Model Based on Use Case.

5.3 Generate Security Test Cases

Security test cases have been generated based on the extracted security test model using MISTA tool. Security test cases have been generated in Java language with reachability tree coverage. Figure 28 presents an example of security test cases based on STRIDE security test model.

```
@Test
public void test1() throws Exception {
    System.out.println("Test case 1");
    SeleniumUtilities.userLoginToTheSystem(driver,"samer","samer123");
    SeleniumUtilities.createEmptyProposalDocument(driver);
    SeleniumUtilities.fillProjectInformationSectionFields(driver,"Redirect Attack MISTA tool test A");
    SeleniumUtilities.fillSponsorAndBudgetInformationSectionFields(driver);
    SeleniumUtilities.fillCostShareInformationSectionFields(driver);
    SeleniumUtilities.fillUniversityCommitmentsSectionFields(driver);
    SeleniumUtilities.fillconflictOfInterestAndCommitmentInformationSectionFields(driver);
    SeleniumUtilities.fillComplianceInformationSectionFields(driver);
    SeleniumUtilities.fillAdditionalInformationSectionFields(driver);
    SeleniumUtilities.fillCollaborationInformationSectionFields(driver);
    SeleniumUtilities.fillProprietaryAndConfidentialInformationSectionFields(driver);
    SeleniumUtilities.fillCertificationAndSignatureSectionFields(driver,GPMSAttacksSourceCode.getXXSScriptRedirectAttack());
    SeleniumUtilities.saveProposalDocument(driver);
    SeleniumUtilities.redirectAttackMitigation(driver);
}

@Test
public void test2() throws Exception {
    System.out.println("Test case 2");
    SeleniumUtilities.userLoginToTheSystem(driver,"samer","samer123");
    SeleniumUtilities.createEmptyProposalDocument(driver);
    SeleniumUtilities.fillProjectInformationSectionFields(driver,"Cannot Edit Attack MISTA tool test A");
    SeleniumUtilities.fillSponsorAndBudgetInformationSectionFields(driver);
    SeleniumUtilities.fillCostShareInformationSectionFields(driver);
    SeleniumUtilities.fillUniversityCommitmentsSectionFields(driver);
    SeleniumUtilities.fillconflictOfInterestAndCommitmentInformationSectionFields(driver);
    SeleniumUtilities.fillComplianceInformationSectionFields(driver);
    SeleniumUtilities.fillAdditionalInformationSectionFields(driver);
    SeleniumUtilities.fillCollaborationInformationSectionFields(driver);
    SeleniumUtilities.fillProprietaryAndConfidentialInformationSectionFields(driver);
    SeleniumUtilities.signProposalByPIByUsingJQuery(driver, GPMSAttacksSourceCode.getCloseProposalDocumentScript());
    SeleniumUtilities.saveProposalDocument(driver);
    SeleniumUtilities.logout(driver);
}
```

Figure 28: GPMS Security Test Case Example.

5.4 Evaluation of Security Test Cases.

As a result of applying our approach, 19 use cases were created to cover all the functionalities of GPMS. 30 distinct misuse cases have been identified by applying STRIDE and security goals that cover most of the attacks that might occur in a web-based application. All of the misuse cases have been applied to the GPMS use cases. Security test models have been extracted and combined based on STRIDE and Use Case, and the security test cases which have been generated based on the security test models.

To evaluate the effectiveness of the generated security test cases, all of the security test cases have been executed against each use case of the GPMS system. 147 security test cases successfully attacked the system and revealed security vulnerabilities that can be exploited by any system adversary such as Cross-Site scripting attacks, update data for other users, access and modify to other user data and DoS attacks…etc. Table 19 shows the result of the security test for GPMS system which demonstrates: misuse cases, corresponding use cases threaten by the misuse case, the result of execution attacks (i.e.; Pass means successful attack. Fail means failed attack), STRIDE category, the vulnerability type discovered, and attack description.

Table 19: GPMS Security Test Cases Results

Misuse Case ID	Misuse Case Name	Use Case Name	pass / fail	STRIDE	Vulnerability Type	Description
MUC-1	URL Redirection Attack	Create New Proposal	✓	D & E	XSS & CSRF	Redirect user to another website to steal information.
		Approve/ Disapprove by Dean	✓			
		Approve/ Disapprove by IRB.	✓			
		Approve/ Disapprove Proposal by Department Chair.	✓			
		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			
		Submit Proposal PI	✓			
		Withdraw Proposal by Research Admin.	✓			

M UC-2	Disable Edit Proposal Action Attack	Create New Proposal	✓	D & E	XSS Attack	Disable edit proposal action that prevents the user to edit or open the proposal document
		Approve/ Disapprove by Dean	✓			
		Approve/ Disapprove by IRB.	✓			
		Approve/ Disapprove Proposal by Department Chair.	✓			
		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			
		Submit Proposal by Principle Investigator PI	✓			
		Withdraw Proposal by Research Admin.	✓			

MUC-3	Disable Save Action Attack	Create New Proposal	✓	D & E	XSS & CSRF	Disable save action where the PI cannot save the proposal document.
MUC-4	Disable Approve/ disapprove actions Attack	Approve/ Disapprove by Dean	✓	D & E	XSS & CSRF	Disable approve and disapprove actions that prevent the users from using them
		Create New Proposal	✓			
		Approve/ Disapprove by IRB.	✓			
		Approve/ Disapprove Proposal by Department Chair.	✓			
		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			
		Withdraw Proposal by Research Admin.	✓			
MUC - 5	Display	Create New Proposal	✓	D & E	XSS Attack	Print out user

	User Information Attack	Approve/ Disapprove by Dean	✓			information on the proposal document by injecting XSS code.
		Approve/ Disapprove by IRB.	✓			
		Approve/ Disapprove Proposal by Department Chair.	✓			
		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			
		Submit Proposal PI	✓			
		Withdraw Proposal by Research Admin.	✓			
UC - 6	Destroy Proposal Document	Create New Proposal	✓	T & D	XSS Attack	Destroy the displaying of proposal document by injecting XSS code
		Approve/ Disapprove by Dean	✓			
		Approve/ Disapprove by IRB.	✓			

		Approve/ Disapprove Proposal by Department Chair.	✓			
		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			
		Submit Proposal PI	✓			
		Withdraw Proposal by Research Admin.	✓			
M UC - 7	Access to Admin Account	Create New Proposal	✓	I & E	XSS & CSRF	Access to the admin account by injecting code that can create a link to access to the admin dashboard.
		Approve/ Disapprove by Dean	✓			
		Approve/ Disapprove by IRB.	✓			
		Approve/ Disapprove Proposal by Department Chair.	✓			

		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			
		Submit Proposal PI	✓			
		Withdraw Proposal by Research Admin.	✓			
M UC - 8	D isable Proposal List View Attack	Create New Proposal	✓	D & E	XSS Attack	By injecting XSS code, the view that list all proposal documents for the users, will be hidden.
		Approve/ Disapprove by Dean	✓			
		Approve/ Disapprove by IRB.	✓			
		Approve/ Disapprove Proposal by Department Chair.	✓			
		Approve/ Disapprove Proposal by Business Manager	✓			

		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			
		Submit Proposal PI	✓			
		Withdraw Proposal by Research Admin.	✓			
M UC - 9	U nauthorized Update proposal fields	Create New Proposal	✓	E & T	Access Control Horizontal Privilege Escalation.	By executing XSS code, the attacker can change non-editable fields values and save the new values in system DB.
		Approve/ Disapprove by Dean	✓			
		Approve/ Disapprove by IRB.	✓			
		Approve/ Disapprove Proposal by Department Chair.	✓			
		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			

		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			
		Submit Proposal PI	✓			
		Withdraw Proposal by Research Admin.	✓			
M UC - 10	D isable All Proposal Actions	Create New Proposal	✓	E & D	XSS Attack	Disab le all proposal document action that disables the user to perform anything on the proposal document by injecting XSS code.
		Approve/ Disapprove by Dean	✓			
		Approve/ Disapprove by IRB.	✓			
		Approve/ Disapprove Proposal by Department Chair.	✓			
		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by	✓			

		Research Director.				
		Submit Proposal by CO-PI	✓			
		Submit Proposal PI	✓			
		Withdraw Proposal by Research Admin.	✓			
		Create New Proposal	✓			
		Approve/ Disapprove by Dean	✓			
		Approve/ Disapprove by IRB.	✓			
UC - 11	Disable Submit Action Attack	Approve/ Disapprove Proposal by Department Chair.	✓	D & I	XSS Attack	Inject XSS code that disables the submits action
		Approve/ Disapprove Proposal by Business Manager	✓			
		Approve/ Disapprove Proposal by Research Administrator	✓			
		Submit Proposal by Research Admin	✓			
		Approve/ Disapprove Proposal by Research Director.	✓			
		Submit Proposal by CO-PI	✓			

		Submit Proposal PI	✓			
		Withdraw Proposal by Research Admin.	✓			
M UC-12	Access to Admin Account from signup new user	Create New User Account	✓	S & E & I &T	XSS Attack	Access to admin account by injecting XSS code in the username field of creating new user account.
M UC-13	Unauthorized Delete Users Account Attack	Create New User Account	✓	S & E & I& T	XSS Attack	Inject XSS code that leads to access to the admin profile from creating new account and then delete user accounts
M UC-14	Unauthorized Activate/ Deactivate User Account Attack	Create New User Account.	✓	S & E & I& T	XSS Attack	Inject XSS code that leads to access to the admin profile from creating new account and then Activate or Deactivate user accounts
M UC-15	Unauthorized Access & modify user proposals Attack	Create New User account	✓	S & E & I& T	XSS Attack	Inject XSS code that leads to access to the admin profile from creating new account and then updates

						user proposals
M UC-16	Unauthorized Delete User Proposals	Create New User Account	✓	S & E & I& T	XSS Attack	Inject XSS code that leads to access to the admin profile from creating new account and then delete user proposals
M UC-17	Unauthorized Modify User Accounts Information Attack.	Create New User Account	✓	S & E & I& T	XSS Attack	Inject XSS code that leads to access to the admin profile from creating a new account and then updates user accounts information.
M UC-18	Unauthorized Delete All Users Accounts Attack	Create New Users	✓	S & E & I& T	XSS Attack	Inject XSS code that leads to access to the admin profile from creating a new account and then delete user accounts.
M UC-19	Unauthorized Delete All User Proposals Attack.	Create New User Account.	✓	S & E & I & T	XSS Attack and Code Injection	Inject XSS code that leads to access to the admin profile from creating a new account and then delete

						proposals document.
M UC-20	Unauthorized Download File Attack.	Create New Proposal Document	✓	E & I	File Path Injection.	File Path Injection, the attacker can craft path and then download the file.
M UC-21	Upload Dangerous Contents Attack	Create New Proposal Document	✓	D & E	Upload Malicious Contents	The attacker can upload any file type such as .JS or .exe and others that can lead to being executed and perform malicious attacks.
M UC-22	Upload Large Files	Create New Proposal Document	✓	D	Quota Overload Denial of Service DoS.	Upload to the system large files that lead to consuming the server disk space.
M UC-23	Overwrite Uploaded Files	Create New Proposal Document.		T & E	Overwrite Uploaded Files.	Upload files with the same name to overwrite the existence uploaded files.
M UC-24	Automatic Users Registration Attack	Create New User Account	✓	D & E	Authentication and Session management	Overwhelming the system by automatic user registration that leads to creating

						malicious accounts and make the server busy with creation request
M **UC- 25**	Username Harvesting Attack	User Login		S	Authentication and Session management	Analysis the error MSG by using the wrong password to validate the username
M **UC-26**	Unauthorized Proposal Disapprove / Disapprove by Co-PI	Sign Proposal by Co-PI		E	Access Control Abusing workflow	CO-PI submit proposal by skipping most of the workflow steps
M **UC-27**	Unauthorized Proposal Submission by research director	Approve/ disapprove proposal by research director		E	Access Control Abusing workflow	Research administrator submits proposal by skipping most of the workflow steps
M **UC-28**	Unauthorized Proposal Archive by BM	Approve/ disapprove proposal by BM		E	Access Control Abusing workflow	Dean archive proposal by skipping most of the workflow steps
M **UC-29**	Unauthorized Proposal Deletion	Approve/ disapprove a proposal by Dept. Chair		E	Access Control Abusing workflow	BM delete proposal by skipping most of the

	by Dept. Chair					workflow steps
M UC-30	U nauthorized Proposal Withdraw by PI.	submit a proposal by PI		E	Access Control Abusing workflow	Actor disapprove proposal by skipping most of the workflow steps

CHAPTER SIX CONCLUSIONS

6.1 Results Analysis

This thesis presented the approach of security testing with misuse case modeling by extracting security test cases from the misuse cases accompanying mitigation use cases and evaluates the effectiveness of the resulted security test cases in revealing software security vulnerabilities. The new structured mapping approach of a textual description of misuse cases and mitigation use cases to security test model presented in Predicate/Transition net has been implemented, and combination techniques of security test models by using STRIDE and Use Case have also been implemented as applied in two case studies with two completely different conditions. In FileZilla server case study, the approach has been applied after the development process had been completed. In GPMS system, the approach has been applied through to the development process.

In both case studies, a structured process has been used to identify the software system use cases based on the application's documentation and user experience. Misuse cases have been defined in a structured manner by applying STRIDE classifications and security goals against the software systems use cases. The misuse cases in both case studies covered the majority of security threats for each system function provided. The proper mitigation use cases have also been defined by applying STRIDE, OWASP, and other security techniques that can prevent the misuse cases for each case study.

In both studies, all the misuse cases accompanying mitigation use cases have been transformed successfully in a structured manner to the corresponding security test model

presented in Predicate/Transitions nets based on the mapping techniques presented in this approach. All of the generated security test models have been successfully converted to the executable code by specifying the MISTA MIME specifications.

Security test model combination techniques have been applied to all of the generated security test models. All of the resulted security test models were successfully combined by using STRIDE and Use Case techniques. All of the combined security test models have been successfully converted to executable code.

The generated security test cases from the combined security test models based on STRIDE technique are efficient for testing the security of a software system against specific security threat type (i.e., privilege elevation threat). In FileZilla FTP Server case study, the generated security test cases can kill 0.868 (33/38) of the security mutants and revealed two security vulnerabilities. The first vulnerability is overflow from the login table while the other is in retrieving the currently running server version that might give the attacker an idea of a possible attack by looking into the public figures.

Both case studies show that the generated security test cases from the combined security test models based on Use Case technique are effective for testing the security of a specific feature or service provided by a software system. In FileZilla FTP Server case study, 22 security test cases have killed 33/38 of the security mutants. In GPMS case study, 153 security test cases have been applied against all of the system functions. They have revealed 24 security vulnerabilities that can be exploited by a system adversary such as cross-site scripting attacks to include; injecting JavaScript code to disable all the proposal actions or other XCMAL attacks such as; without permissions, update proposal document

fields information, and other security vulnerabilities such as file path inactions and DoS of uploading large files (see Table 19).

Both studies show that security testing with misuse case modeling is particularly efficient and effective. The generated security test cases based on STRIDE and Use Case combination techniques are efficient and both combination techniques generate the same number of security test cases that killed the same number of mutants in FileZilla FTP Server case study.

Evidence of the effectiveness of security testing with misuse case modeling in both case studies is that the misuse cases and the resulting security test models are built as if the tester is an intelligent system adversary and the generated security test cases are direct to the target. i.e. the vulnerabilities that can be exploited by the system adversary.

6.2 Future Work

Developing a tool to automatically extract a security test model is a method superior to extracting the security test model from the textual description of misuse cases accompanying mitigation use cases by hand. For example, a tool that assists in managing and tracing the use cases, that are related to misuse cases and mitigation use case, then mapping the misuse cases and mitigation use case to security test model as represented in Predicate/Transition net is beneficial because of the intensive amount of detailed work that extracting and managing the security test models, use cases, misuse cases, and mitigation use cases requires. Any tool that could automate or semi-automate any of the approach steps would a be positive for such a system as that tool would save time and would eliminate unnecessary errors.

Another possible research scope is by including misuse case modeling through the agile software development process. Similar to use case, a user story is a high-level artifact that captures the requirements description and contains information such as what is the feature, why that feature is needed. Extending the user story to include the potential security threats, the consequences of these threats, and possible mitigation techniques for the system feature will help to address the software system security requirements in the early stages of the software development process and provide a reliable software system instead of installing patches at the end of software development.

REFERENCES

[1] Sindre G, Opdahl AL. Capturing security requirements through misuse cases. NIK 2001, NorskInformatikkonferanse 2001, http://www. nik. no/2001. 2001 Nov 26.

[2] Premkumar T. Devanbu and Stuart Stubblebine. 2000. Software engineering for security: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (ICSE '00). ACM, New York, NY, USA, 227-239. DOI=http://dx.doi.org/10.1145/336512.336559.

[3] Shujuan Ji and Xiaohong Bao, "Research on Software Hazard Classification", Procedia Engineering, vol. 80, pp. 407, 2014, ISSN 18777058.

[4] C. Rolland, C. B. Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N. Maiden, et al., "A Proposal for a Scenario Classification Framework", Requirements Engineering Journal, vol. 3, no. 1, 1998.

[5] Meier, J. D., Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla, and Anandha Murukan. "Improving web application security: threats and countermeasures." *Microsoft Corporation* 3 (2003).

[6] Alexander, "Initial industrial experience of misuse cases in trade-off analysis," Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on, 2002, pp. 61-68.

[7] Mæhre, Magne. "Industrial experiences with misuse cases." (2005). Mater Thesis, Department of Computer Science, Trondheim, Norway.

[8] Guttorm, S., & Andreas L, O. (2008). Misuse cases for identifying system dependability threats. *Journal of Information Privacy and Security*, *4*(2), 3-22.

[9] Breivik, G.F. Abstract Misuse Patterns – A new Approach to Security Requirements (2002). Master Thesis, Department of Computer Science, University of Bergen, Norway.

[10] I. A. Tøndel, J. Jensen and L. Røstad, "Combining Misuse Cases with Attack Trees and Security Activity Models," Availability, Reliability, and Security, 2010. ARES '10 International Conference on, Krakow, 2010, pp. 438-445.

[11] J. J. Pauli and D. Xu, "Misuse case-based design and analysis of secure software architecture," International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, 2005, pp. 398-403 Vol. 2.

[12] K. Saleh and M. Habil, "The Security Requirements Behavior Model for Trustworthy Software," *e-Technologies, 2008 International MCETECH Conference on*, Montreal, Que., 2008, pp. 235-238.

[13] Røstad, Lillian. "An extended misuse case notation: Including vulnerabilities and the insider threat." In XII Working Conference on Requirements Engineering: Foundation for Software Quality, Luxembourg. 2006.

[14] Dimitrakos, Theo, Brian Ritchie, Dimitris Raptis, Jan Øyvind Aagedal, Folker den Braber, Ketil Stølen, and Siv Hilde Houmb. "Integrating model-based security risk management into ebusiness systems development." In *Towards the Knowledge Society*, pp. 159-175. Springer US, 2003.

[15] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," Computer Security Applications Conference, 1999. (ACSAC '99) Proceedings. 15th Annual, Phoenix, AZ, 1999, pp. 55-64.

[16] J. Whittle, D. Wijesekera and M. Hartong, "Executable misuse cases for modeling security concerns," 2008 ACM/IEEE 30th International Conference on Software Engineering, Leipzig, 2008, pp. 121-130.

[17] http://cs.boisestate.edu/~dxu/research/MBT.html

[18] https://www.owasp.org/index.php/Main_Page

[19] https://filezilla-project.org/https://filezilla-project.org/

[20] https://www.ietf.org/rfc/rfc959.txt

[21] https://sourceforge.net/top

[22] R. Matulevicius, N. Mayer and P. Heymans, "Alignment of Misuse Cases with Security Risk Management," Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, Barcelona, 2008, pp. 1397-1404.

[23] D. Xu, M. Tu, M. Sanford, L. Thomas, D. Woodraska and W. Xu, "Automated Security Test Generation with Formal Threat Models," in IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 4, pp. 526-540, July-Aug. 2012.

[24] J. Pauli and Dianxiang Xu, "Integrating functional and security requirements with use case decomposition," 11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'06), Stanford, CA, 2006, pp. 10 pp.

[25] D. Xu and K. E. Nygard, "Threat-driven modeling and verification of secure software using aspect-oriented Petri nets," in IEEE Transactions on Software Engineering, vol. 32, no. 4, pp. 265-278, April 2006.

[26] M. El-Attar and I. Ahmad, "Improving Quality in Misuse Case Models: A Risk-Based Approach," Computer and Information Science (ICIS), 2011 IEEE/ACIS 10th International Conference on, Sanya, China, 2011, pp. 337-342.

[27] Schieferdecker, Ina, Juergen Grossmann, and Martin Schneider. "Model-based security testing." arXiv preprint arXiv:1202.6118 (2012).

[28] https://magento.com/

[29] Mead, Nancy R., and Ted Stehney. *Security quality requirements engineering (SQUARE) methodology*. Vol. 30, no. 4. ACM, 2005.

[30] Manico, Jim, and August Detlefsen. *Iron-clad Java: Building Secure Web Applications*. McGraw Hill Professional, 2014.

[31] Zech, Philipp. "Risk-based security testing in cloud computing environments." In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, pp. 411-414. IEEE, 2011.

APPENDIX A CASE STUDY 1: FILEZILLA SERVER MISUSE CASE MODELING DATA

A.1 FileZilla Server Use Cases

A.1.1 FileZilla FTP Server Uploading Files Use Case Description

Table 20: FileZilla FTP Server Uploading File Use Case.

Use case #	**UC1**
Use case name	Upload file.
Actors	User/System.
Goal	To upload a file from client machine to FTP server system.
Preconditions	1. The actor logged into the system.
Main Flow	1. The actor provides the "STOR" command with the remote destination folder and local source file paths by using command line tool. 2. The system receives the STOR command request. 3. The system validates destination folder path**.** 4. The system provides the actor a summary of file uploading. 5. The system records the request in the log file.
Post-condition	1. The file stored in the destination folder. 2. The file has the same name. 3. The file has the same contents.
Extension Points	NONE
Alternative Flow	**1a. The actor uses GUI client application.** 1.a.1. The actor drags the file from local source folder in the client machine. 1.a.2. The actor drops the source file into the remote system destination folder. The use case continuous at the System receives STOR command in the MF.
Exception Flow	**6a. The destination folder does not have enough space.** 6.a.1. The system sends an error message that informs the user cannot upload the file.
Recovery Flow	**4a. The remote destination file path does not exist.** 4.a.1. The system notifies the user that the destination folder does not exist. The use case continuous at The actor provides STOR command with existing remote destination folder in the MF.

A.1.2 FileZilla FTP Server Delete File/Directory Use Case Description

Table 21: FileZilla FTP Server Delete File/Directory Use Case.

Use case #	**UC2**
Use case name	Deleting files and/or Directories.
Actors	User, System.
Goal	To delete a file or/and directories from the client machine to FTP server.
Preconditions	1. The system is up and running. 2. The actor logged into the system.
Main Flow	1. The actor provides the "DELE" command with remote destination folder/directory path and its name by using command line tool. 2. The system receives the "DELE" command request. 3. The system validates the destination file path**.** 4. The system deletes the file or the directory. 5. The system provides the actor a summary of file/directory deleting process. 6. The system records the request in the log file.
Post-conditions	1. The file or directory is deleted successfully.
Extension Points	NONE
Alternative Flow	**1a. The actor uses GUI client application.** 1.a.1. The actor selects the file/directory that wishing to delete in the remote destination folder. 1.a.2. The actor deletes the file or directory from the available actions provided from file or directory properties. The use case continuous at System validates source file path in the MF.
Exception Flow	**3.a. The file name does not exist.** 3.a.2 The system throws error message telling the client that the file does not exist.
Recovery Flow	**4a. The destination path does not exist or invalid file/folder name.** 4. a.1 The system notifies the actor that the destination does not exist. The use case continuous at the actor provides "DELE" command with existing destination folder path or with valid file/folder name in the MF.

A.1.3 FileZilla FTP Server Create Directory Use Case Description

Table 22: FileZilla FTP Server Create Directory Use Case.

Use case #	**UC3**
Use case name	Create Directory.
Actors	User, System.
Goal	To create a directory from the client machine to the FTP server.
Precondition	1. The system is up and running. 2. The actor logged into the system.
Main Flow	1. The actor provides the "MKD" command with specifying the destination folder path with a folder name that wishing to create by using command line tool. 2. The system receives the "MKD" command request. **3.** The system validates the destination file path**.** **4.** The system validates the name of the folder. 5. The system creates the directory and saves it. 6. The system provides the actor a summary of the created directory. 7. The system records the request in the log file.
Post-conditions	1. The folder created in the correct destination. 2. The folder created by the name specified earlier.
Extension Points	NONE
Alternative Flow	**1. a. The actor uses GUI client application.** 1. a.1 The actor selects the destination folder that wishing to create the folder inside of it. 1. a.2 The actor creates a directory from the available actions provided from destination folder properties, the use case continuous at System receives the "MKD" command request in the MF.
Exception Flow	**6. a. The destination folder does not have enough space.** 6. a.1 The system disconnected and starting over again and never finishing creating the directory.
Recovery Flow	**4. a. The destination folder path does not exist.** 4.a.1 The system notifies the actor that the destination folder does not exist, the use case continuous at The actor provide "MKD" command with existing destination folder path in the MF.

	5. a. Invalid folder name. 5. a.1 The system notifies the actor that the folder name is invalid, the use case continuous at the actor provide "MKD" command with the valid folder name in the MF.

A.1.4 FileZilla FTP Server Download Files Use Case Description

Table 23: FileZilla FTP Server Download Files Server Use Case.

Use case #	**UC 4**
Use case name	Download files.
Actors	User, System.
Goal	To download a file from the system.
Precondition	1. The system is up and running. 2. The actor logged into the system.
Main Flow	1. The actor provides the "RETE" command with the remote source file and local destination folder paths by using command line tool. 2. The system receives the RETE command request. 3. The system validates source file path**.** 4. The system validates the source file name. 5. The system provides the actor a summary of downloading the file. 6. The system records the request in the log file.
Post-condition	1. The file downloaded in the destination folder. 2. The file has the same name. 3. The file has the same contents.
Extension Points	NONE
Alternative Flow	**1.a. The actor uses GUI client application.** 1.a.1 The actor drags the file from the remote source folder from the system machine. 1.a.2 The actor drops the remote source file into the local destination folder. The use case continuous at the System receives RETE command in the MF.
Exception Flow	**2.a. The file name does not exist.** 2.a.2 The system throws error message telling the client that the file does not exist.

Recovery Flow	**4.a. The destination path does not exist.** 4. a.1 The system notifies the actor that he/she/system the source file does not exist. The use case continuous at the actor provides "RETE" command with existing source file path in the MF. **5. a. Invalid file name.** 5. a.1 The system notifies the actor that he/she/system the source file name is invalid, the use case continuous at the actor provide "RETE" command with the valid file name in the MF.

A.1.5 FileZilla FTP Server Login Description

Table 24: FileZilla FTP Server Login Use Case

Use case #	**UC5**
Use case name	Log into the system.
Actors	User, System.
Goal	To log into the system by using username and password
Precondition	1- The system is up and running. 2- The actor already registered in the system.
Main Flow	1. The actor provides the "PASS" command with specifying username, password, and the system IP address by using command line tool. 2. The system validates the actor username and password. 3. The system validates the number of failed trials. 4. The system validates the actor enabling status. 5. The system validates the max number of connections. 6. The system validates the IP address. 7. The system changes to the actor home directory. 8. The system records the login request in the log file.
Post-condition	1. The actor successfully login to the system. 2. The system lists the files and folders in the home directory.
Extension Points	NONE
Alternative Flow	**1.a. The actor uses GUI client application.** 1.a.1 The actor provides the username, password and system IP address in the GUI fields. The use case continuous at The system receives the PASS command.
Exception Flow	**7.a. The home directory does not exist.** 7.a.1 The system informs the actor that cannot get the home directory. **4.a. No more login trails left.**

	4.a.1 The system suspends the user account. 4.a.2 The system informs the actor that the account has been suspended. **5.a. The actor account disabled.** 5.a.1 The system informs the actor that the account has been suspended. **6.a. Max number of available connection reached.** 6.a.1 The system inform cannot accept any connection because the max number of login actors has been reached. **7.a. The system refused any connection from this IP address.** 7.a.1 The system informs the actor that cannot accept any connection from this IP address.
Recovery Flow	**3.a invalid username and/or password and more trails left** 3.a.1 The system informs the actor that the username or password are not correct, the use case continuous at The system receive the PASS command with the correct username and/or password in the MF.

A.1.6 FileZilla FTP Server Rename Files/Directories Use Case Description

Table 25: FileZilla FTP Server Rename Files/Directories Use Case.

Use case #	**UC 6**
Use case name	Rename files or directories.
Actors	User, System.
Goal	To rename a file or a directory
Precondition	1. The system is up and running. 2. The actor logged into the system.
Main Flow	1. The actor provides the "RNTO/RNFR" command with the old and new name of the remote file or folder by using command line tool. The system receives the RNTO/ RNFR command request. 2. The system validates file/folder path**.** 3. The system validates the new name. 4. The system renames file/folder. 5. The system provides the actor a summary of renaming the file/folder. 6. The system records the request in the log file.

Post-condition	1. The file/folder renamed successfully. 2. The file/folder stay in the same destination.
Extension Points	NONE
Alternative Flow	**1.a The actor uses GUI client application.** 1.a.1 The actor selects the remote file/folder that wishing to rename. 1.a.2 The actor renames the file from the available actions provided from the file properties. The use case continuous at The actor provides the "RNTO/RNFR" Command in the MF.
Exception Flow	**2.a. The file/folder does not exist**. 2.a.1 The system notifies the actor that the file/folder does not exist.
Recovery Flow	**3.a. Invalid new file name/ the new name corresponds to already exist file/folder name.** 3.a.1 The system notifies the actor that the new name file is invalid or already exist, the use case continuous at the actor provide "RETE" command with the valid file name in the MF.

A.1.7 FileZilla FTP Server Append Files Use Case Description

Table 26: FileZilla FTP Server Append Files Use Case.

Use case #	**UC7**
Use case name	Append file.
Actors	User/System.
Goal	To append a file from client machine to the system.
Preconditions	1- The system is up and running. 2- The actor logged into the system.
Main Flow	1- The actor provides the "APPE" command with the remote destination folder and local source file paths by using command line tool. 2- The system validates remote destination folder path**.** 3- The system receives file data and stores it in the destination folder**.** 4- The system provides the actor a summary of the appending file. 5- The system records the request in the log file.
Post-condition	1- The file appended with same data being sent from the actor. 2- File data is not duplicated.
Extension Points	NONE

Alternative flow	NONE
Exception Flow	**3.a. The destination folder does not have enough space.** 3.a.1 The system disconnected and starting over again and never finishing appending the file.
Recovery Flow	**3.a. The remote destination file path does not exist.** 3.a.1 The system notifies the user that the destination folder does not exist. The use case continuous at The actor provides APPE command with existing remote destination folder in the MF.

A.1.8 FileZilla FTP Server List Directories Use Case Description

Table 27: FileZilla FTP Server List Directories Use Case.

Use case #	**UC8**
Use case name	List Directories.
Actors	User, System.
Goal	To list directories on the shared folder
Preconditions	1. The system is up and running. 2. The actor logged into the system.
Main Flow	1. The actor provides the "MLSD/LIST" command with remote destination folder/directory path and its name by using command line tool. 2. The system receives the "MLSD/LIST" command request. **3.** The system validates the directory path**.** 4. The system lists the files and the directories. 5. The system provides the actor a summary directory list process. 6. The system records the request in the log file.
Post-conditions	1. The directory contents listed successfully.
Extension Points	NONE
Alternative Flow	**1a. The actor uses GUI client application.** 1.a.1. The actor selects the directory that wishing to list in the remote destination folder. 1.a.2. The actor select list action from the available actions provided from directory properties. The use case continuous at System validates source file path in the MF.

Exception Flow	**3.a. The directory name does not exist.** 3.a.2 The system throws error message telling the client that the directory does not exist.
Recovery Flow	**3a. The directory path does not exist or invalid directory name.** 3. a.1 The system notifies the actor that the directory does not exist. The use case continuous at the actor provides "MLSD" command with existing directory path or with folder name in the MF.

A.1.9 FileZilla FTP Server List Subdirectory Use Case Description

Table 28: FileZilla FTP Server List Subdirectory Use Case.

Use case #	**UC8**
Use case name	List Subdirectory.
Actors	User, System.
Goal	To list directories on the shared folder
Preconditions	3. The system is up and running. 4. The actor logged into the system.
Main Flow	7. The actor provides the "NLIST" command with remote destination folder/directory path and its name by using command line tool. 8. The system receives the "NLIST" command request. **9.** The system validates the directory path**.** 10. The system lists the files and the directories. 11. The system provides the actor a summary directory list process. 12. The system records the request in the log file.
Post-conditions	2. The directory contents listed successfully.
Extension Points	NONE
Alternative Flow	**1a. The actor uses GUI client application.** 1.a.1. The actor selects the directory that wishing to list in the remote destination folder. 1.a.2. The actor select list action from the available actions provided from directory properties. The use case continuous at System validates source file path in the MF.
Exception Flow	**3.a. The Subdirectory name does not exist.** 3.a.2 The system throws error message telling the client that the subdirectory does not exist.

Recovery Flow	**3a. The Subdirectory path does not exist or invalid directory name.** 3. a.1 The system notifies the actor that the directory does not exist. The use case continuous at the actor provides "MLSD" command with existing directory path or with folder name in the MF.

A.1.10 FileZilla FTP Server Logout Use Case Description

Table 29: FileZilla FTP Server Logout Use Case

Use case #	**UC9**
Use case name	Logout from the system.
Actors	User/System.
Goal	To quit and close the connection from the system
Preconditions	1. The actor logged into the system.
Main Flow	1. The actor issues the "QUIT" command. 2. The system closes the connection to the client 3. The system records the logout process in the log file
Post-condition	The actor log out successfully from the system.
Extension Points	NONE
Alternative flow	**1a. The actor uses GUI client application.** 1.a.1 The client selects the close connection button.
Exception Flow	NONE.
Recovery Flow	NONE.

A.1.11 FileZilla FTP Server List System Features Use Case Description

Table 30: FileZilla FTP Server List System Features Use Case.

Use case #	**UC10**
Use case name	List system features.
Actors	User/System.

Goal	To list the system features.
Preconditions	1. The actor already logged into the system.
Main Flow	1. The actor issues the "FEAT" command. 2. The system response with feature list supported 3. The system records the actor request in the log file
Post-condition	The system list all features available in the system.
Extension Points	NONE
Alternative flow	NONE
Exception Flow	NONE.
Recovery Flow	NONE.

A.2 FileZilla Server Misuse Cases

A.2.1 Unauthorized Access/Modify Files Misuse Case Description

Table 31: Unauthorized Access/Modify Files Misuse Case.

Misuse case#	**MUC2**
Misuse case name	Unauthorized access/modifying files.
Misuse case category	Privilege Elevation, Tampering, Repudiation
Goal	To access and/or modify files/folders without permission validation.
Actor	Adversary
Preconditions	1. The system permits anonymous login. 2. The anonymous user does not have a permission to access and modify files.
Main Flow	1. The attacker or legitimate user login to the system anonymously or by using valid username and password. 2. The attacker upload files to the system 3. The attacker deletes files from the system. 4. The attacker renames and lists files of the system. 5. The attacker download files from the system.

Post-conditions	The attacker successfully tempers the system data.
Threat point	1. Uploading files, Main flow, Step 3 system validates user permission 2. Downloading files, Main flow, Step 3, the system validates user permission. 3. Creating directory, Main flow, Step 3, the system validates user permission. 4. Change work directory, Main flow, Step 3, the system validates user permission. 5. Append File, Main flow, Step 3, system validates user permission. 6. Delete folder/files, Main flow, Step 3, system validates user permission. 7. Rename Folder/file. Main flow, Step 3, system validates user permission.
Mitigation	Validate user permission mitigation use case.

A.2.2 Crack User Password Misuse Case Description

Table 32: Crack User Password Misuse Case.

Misuse case#	**MUC1**
Misuse case name	Crack user passwords
Misuse case category	Spoofing
Goal	To crack the user login information.
Actor	System, adversary.
Preconditions	1- The system enables anonymous login.
Main Flow	1- The attacker issues the login command with username and guessing the password. 2- The attacker keeps sending login command with username and different guessing password.

Post-conditions	The attacker cracks the user password.
Threat point	User login, Main flow, Step 1, the actor issues login command.
Mitigation	Auto ban IP address mitigation use case

A.2.3 Overflow Login Table Misuse Case Description

Table 33: Overflow Login Table Misuse Case.

Misuse case#	**MUC3**
Misuse case name	Overflow login table.
Misuse case category	Spoofing, Denial of service attack
Goal	To overflow the login table and crash the system.
Actor	Adversary
Preconditions	2. The attacker has an account on the system.
Main Flow	1. The attacker issues login command with the user id. 2. The attacker keeps issuing; 2.1 iterate the login command for many times. 3. The system login table cannot handle any new connection for the same user id.
Post-conditions	Overflow login table and the system will crash
Threat point	User login, Main flow, Step 1 The actor provides the "PASS" command
Mitigation	Auto ban IP address Mitigation use case

A.2.4 Injecting Malicious Code Misuse Case Description

Table 34: Injecting Malicious Code Misuse Case.

Misuse case#	**MUC4**
Misuse case name	Injecting malicious code
Misuse case category	Denial of service attack
Goal	To disturb the system services
Actor	Adversary, system

Preconditions	1. The system enables anonymous login.
Main Flow	1. The attacker anonymously log in to the system. 2. The attacker injecting malicious code LIST command. 3. The attacker injecting malicious code in the Download command. 4. The attacker injecting malicious code in the Rename command. 5. The attacker injecting malicious code in the Upload command
Post-conditions	The system will be crashed or suspended.
Threat point	1. Change work directory, Main flow, Step 2 The actor start sending file data. 2. Upload a file, Main flow, Step 1 the user sends the STOR command, step 2, the system receives the command and process it. 3. Download File, Main flow, Step 1, the user sends the RETE command, step 2, the system receive the command and process it. 4. Rename file/folders. Main flow, Step 1, the user sends the RENTO command, step 2, the system receives the command and process it. 5. Create Folder, Main flow, Step 1, the user sends the MKD command, step 2, the system receives the command and process it. 6. List directory, Main flow, Step 1, the user sends the LIST, LIST commands, step 2, and the system receive the command and process it.
Mitigation	Validate user input mitigation use case

A.3 FileZilla Server Mitigation Use Cases

A.3.1 Validate User Input Mitigation Use Case Description

Table 35: Validate User Input Mitigation Use Case.

Mitigation use case #	**MITI-UC 1**
Mitigation use case name	Validate user Input.
Goal	To validate user input.
Precondition	The actor issues a command with specific input.
Main Flow	1. Validate the user input against regular expressions, this allows to checks the syntax of the user input. 2. Validates the user input components, this allows the makes sure if the user input has malicious code injected. 3. Return a result of the user input validation to the system. 4. Based on the returned result the system will throw error or process user command.
Post Condition	Validate and prevent malicious code to be processed by the system.
Priority	High

A.3.2 Validate User Permission Mitigation Use Case Description

Table 36: Validate User Permission Mitigation Use Case.

Mitigation use case #	**MITI-UC 2**
Mitigation use case name	Validate user permission
Goal	To validate the user permission before access and/or modifying data.
Precondition	The actor login to the system as an anonymous or legitimate user.
Main Flow	1. Retrieve the actor permission. 2. Read the actor permission for the requesting command. 3. Return the right actor permission for the requesting command. 4. Based on the returned result the system will throw error or process the actor requesting command.
Post Condition	Read and validate the right the permission for the actor.

Priority	High

A.3.3 Auto Ban IP Address Mitigation Use Case Description

Table 37: Auto Ban IP Address Mitigation Use Case.

Mitigation use case#	**MITI-UC 3**
Mitigation Use Case Name	Auto Ban IP Address.
Goal	To Ban IP Address That Keep Sending Login Requests the System.
Precondition	1. Anonymous Login Made to The System.
Main Flow	1. Record The Number of Tries of the Login Request for Each IP Address. 2. Compare The Max Value of the Allowed IP Address for Issues Login Request with Recorded Number for the Number of tries 3. Block Incoming Login Request From The IP Address for A Specific Time.
Post Condition	Prevent Dos Attack and User Password Guessing.
Priority	High.

A.4 FileZilla Server Security Test Models Based on Use Case Technique

A.4.1 Uploading Files Use Case Security Test Model

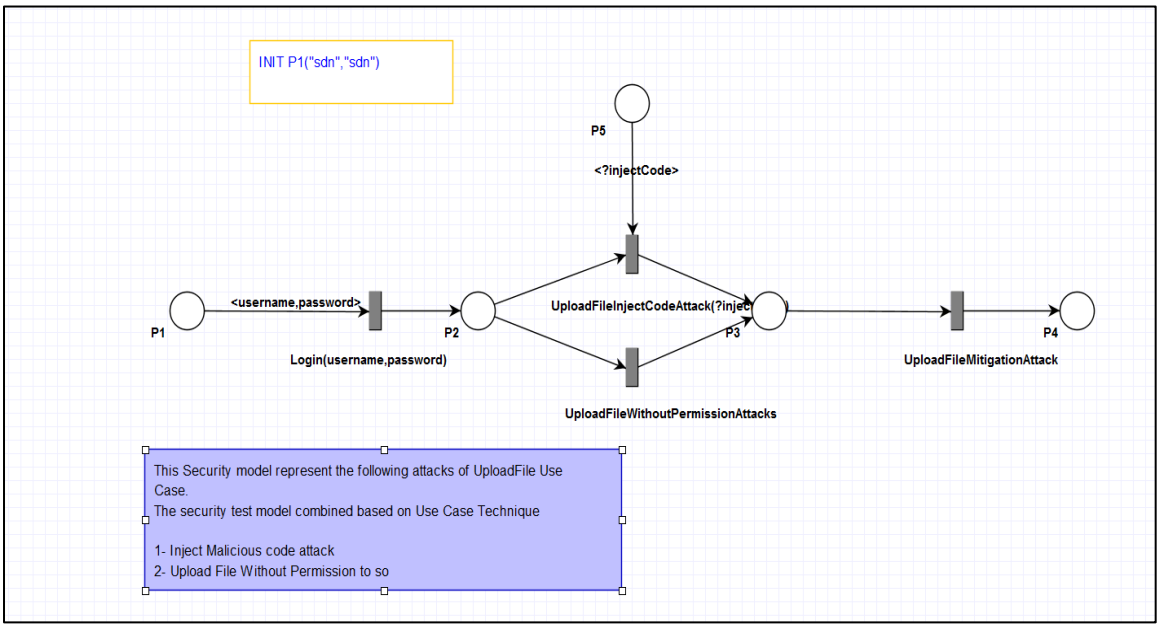

Figure 29: Upload Files Use Case Security Test Model.

A.4.2 Delete File\Directory Use Case Security Test Model

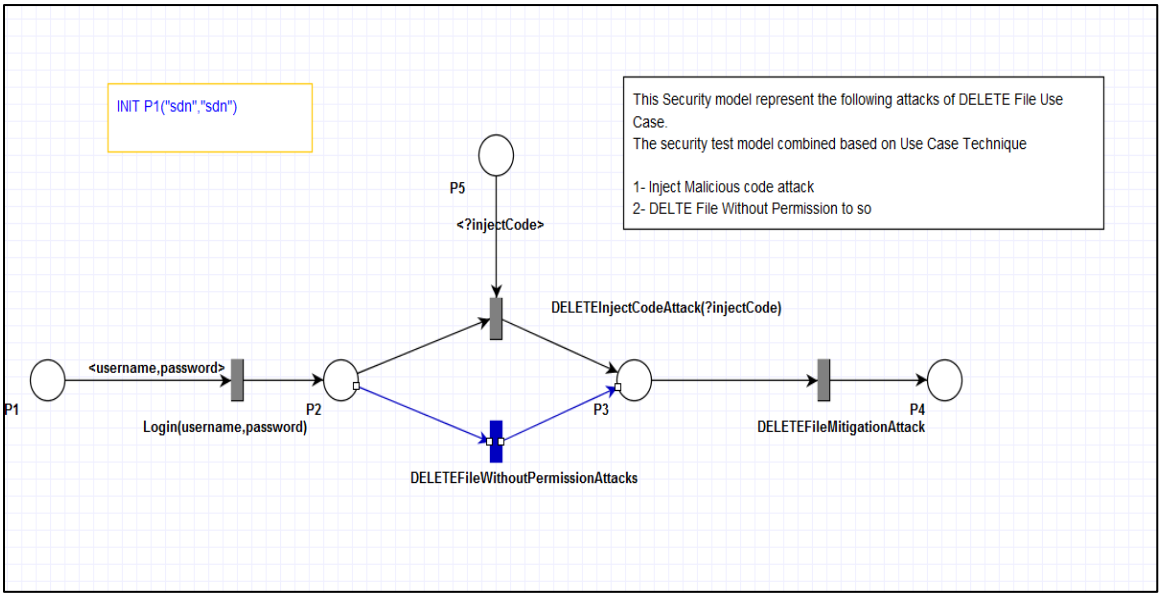

Figure 30: Delete File and Directory Use Case Security Test Model.

A.4.3 Create Directory Use Case Security Test Model

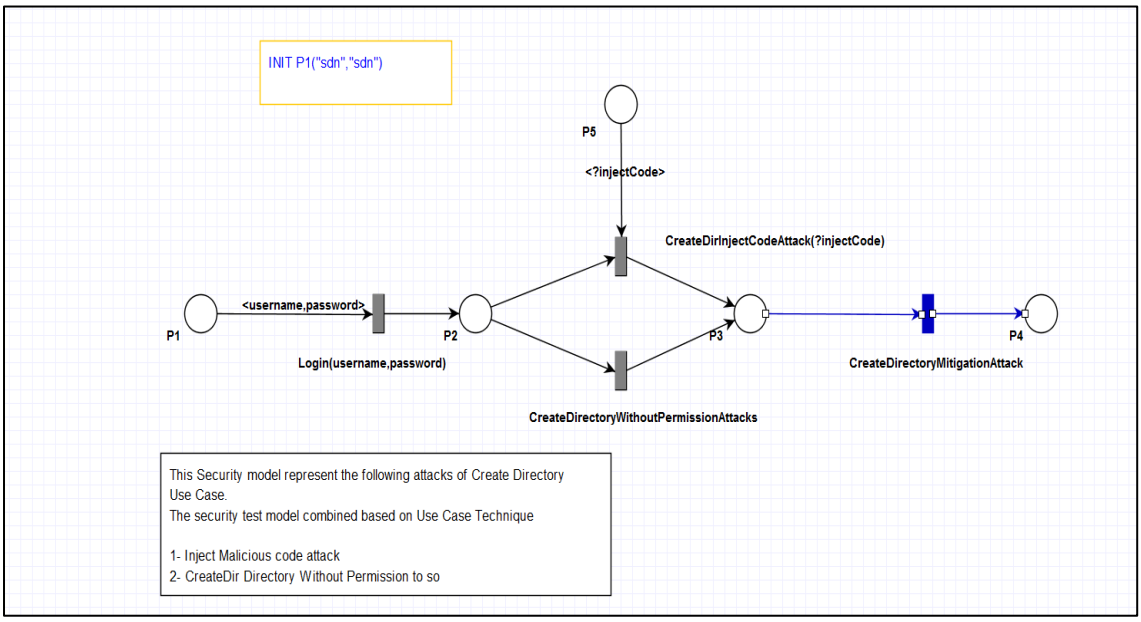

Figure 31: Create Directory Use Case Security Test Model.

A.4.4 Download Files Use Case Security Test Model

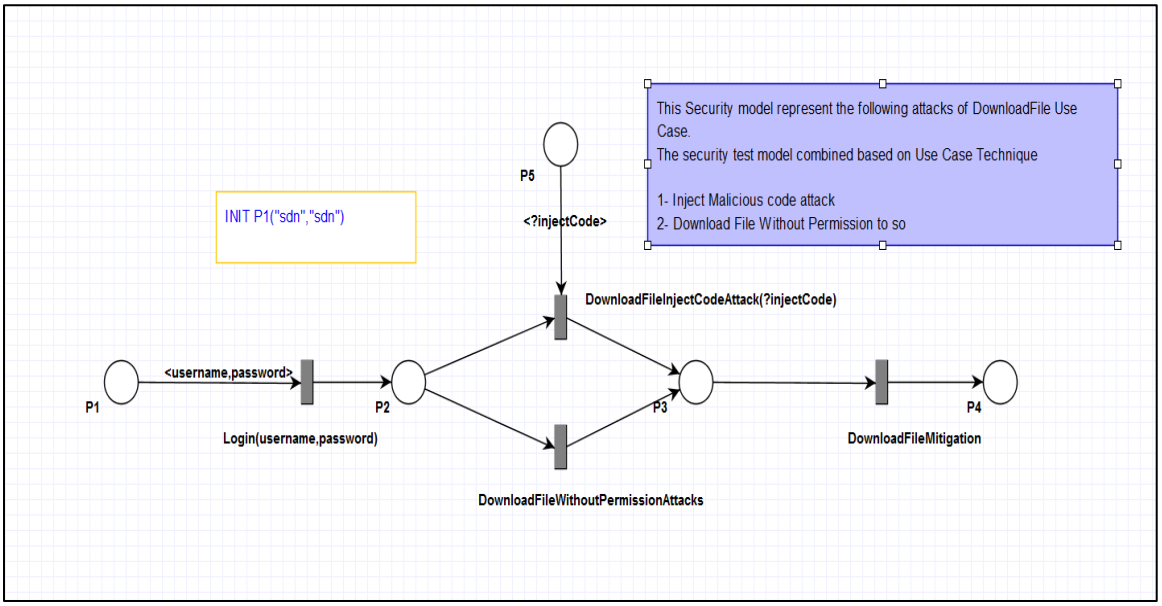

Figure 32: Download Files Use Case Security Test Model.

A.4.5 FileZilla FTP Server Login Use Case Security Test Model

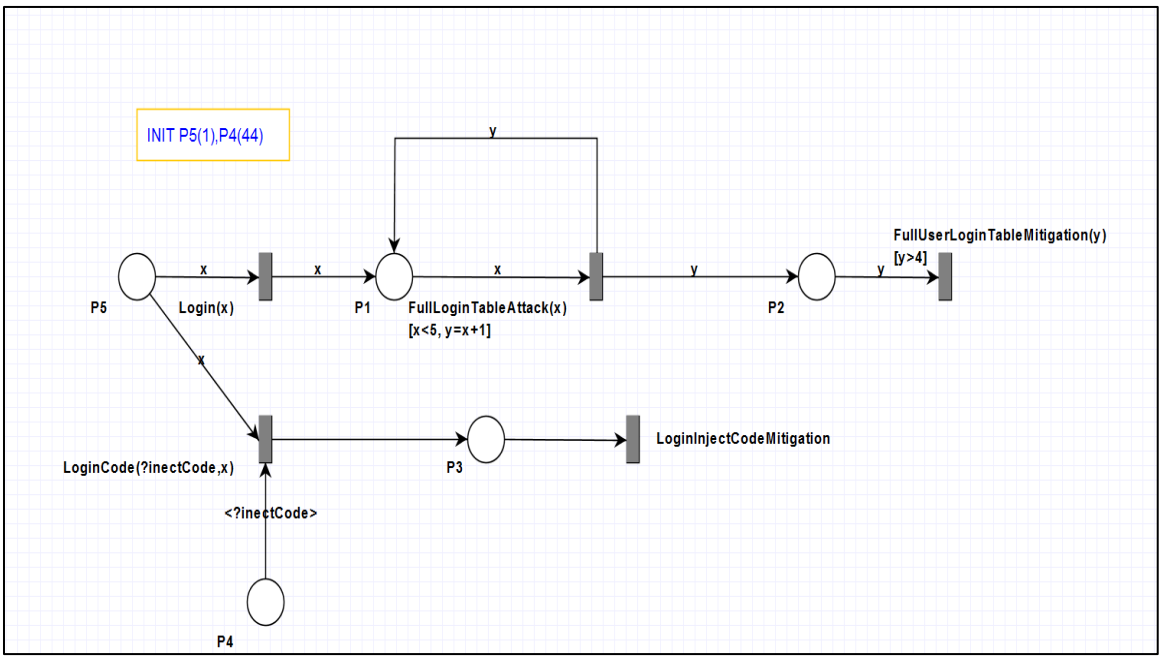

Figure 33: FileZilla FTP Server Login Use Case Security Test Model.

A.4.6 Rename Files\Directories Use Case Security Test Model

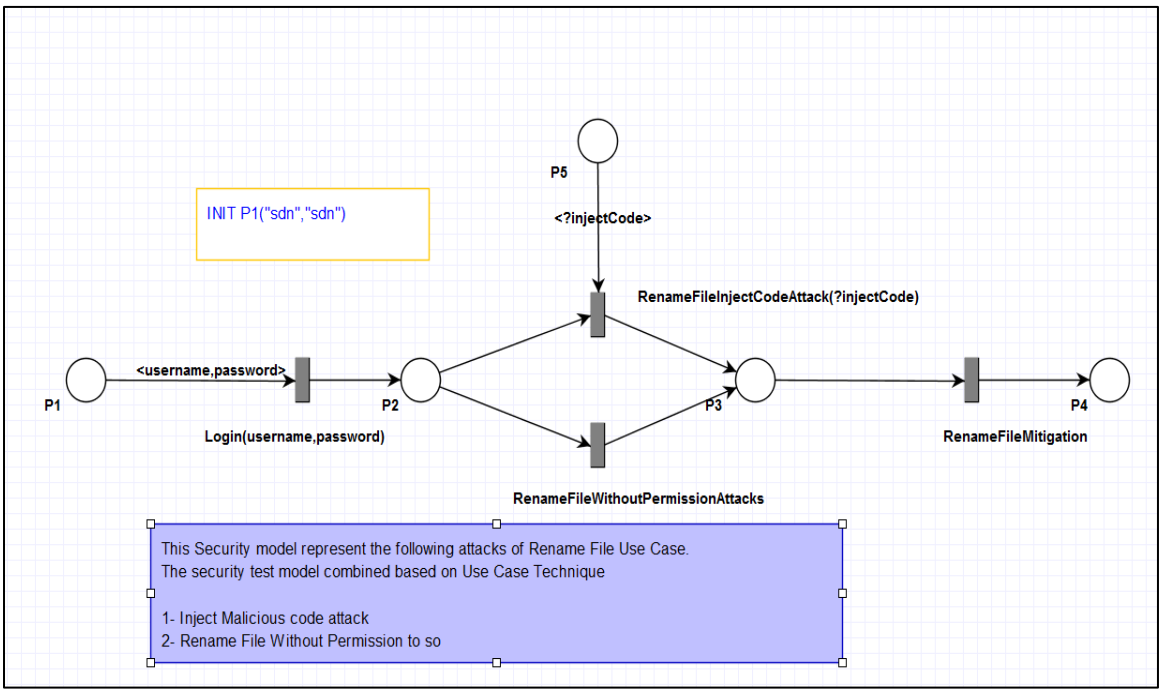

Figure 34: Rename Files/Directory Use Case Security Test Model.

A.4.7 Append Files Use Case Security Test Model

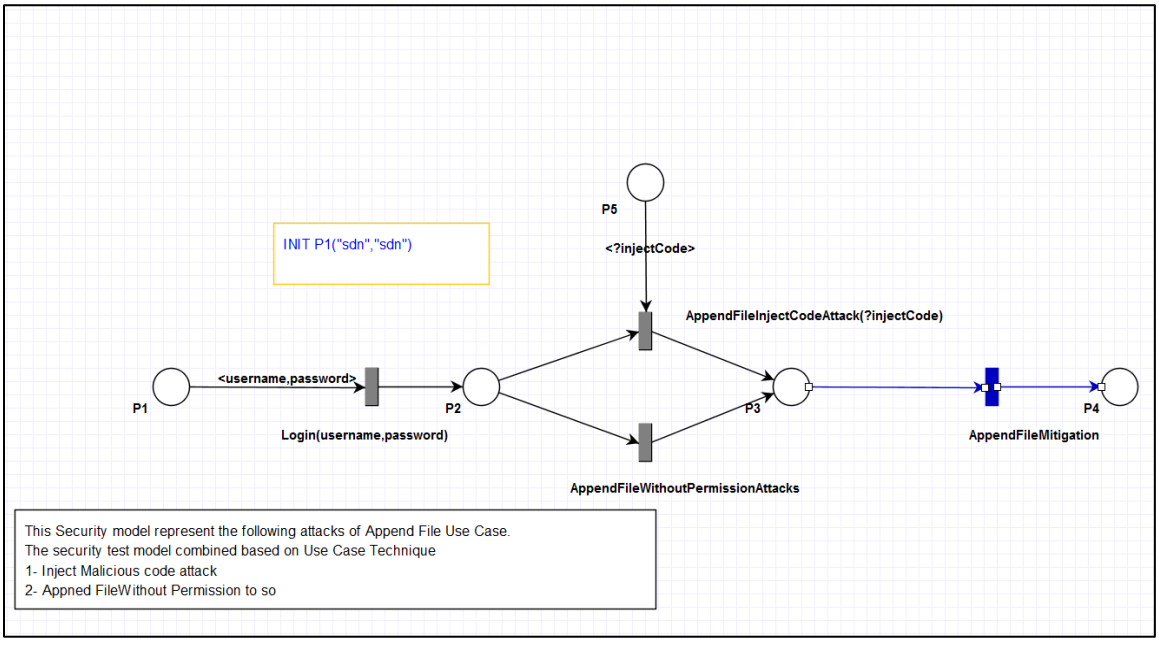

Figure 35: Append Files Use Case Security Test Model.

A.4.8 List Directories Use Case Security Test Model

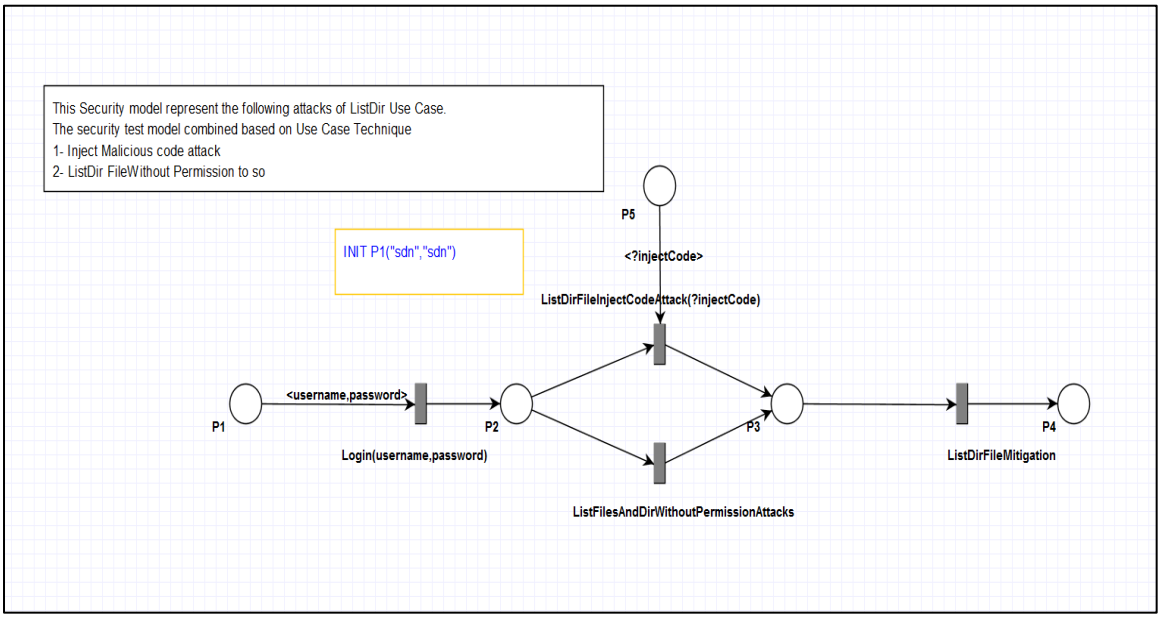

Figure 36: List Directories Use Case Security Test Model.

A.4.9 List Subdirectory Use Case Security Test Model

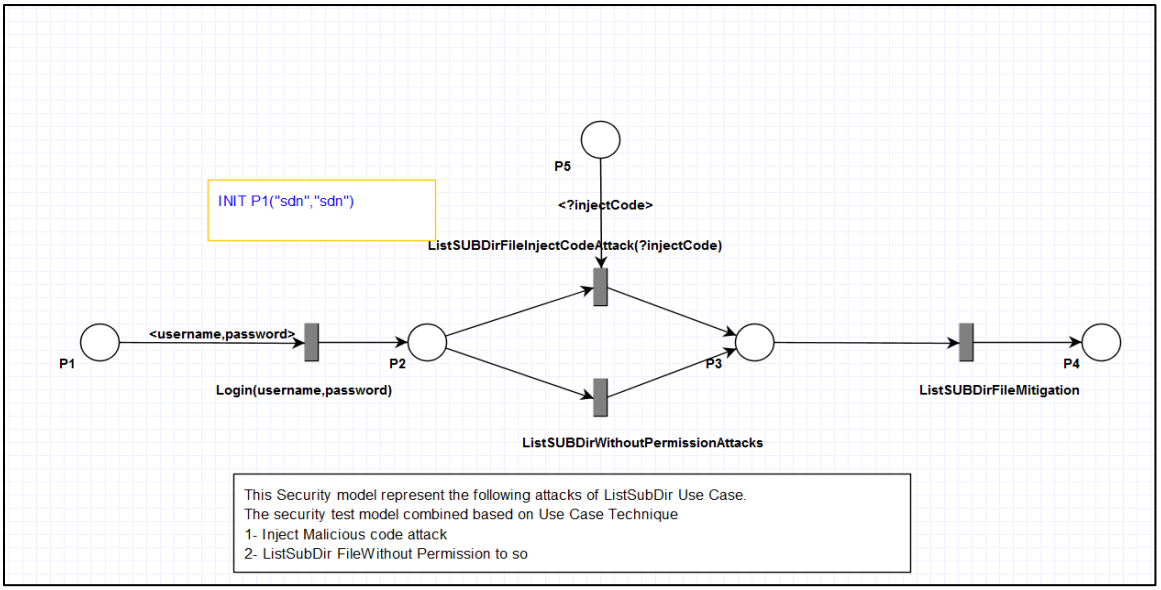

Figure 37: List Subdirectory Use Case Security Test Model.

A.4.10 FileZilla FTP Server Logout Use Case Security Test Model

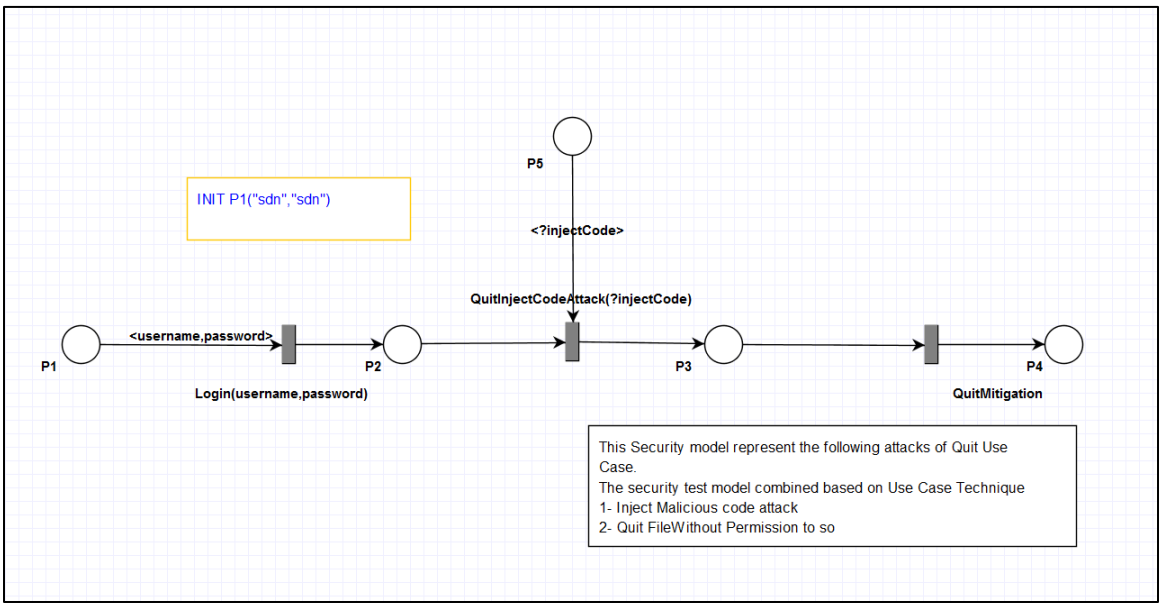

Figure 38: FileZilla FTP Server Logout Use Case Security Test Model.

A.4.11 List System Features Use Case Security Test Model

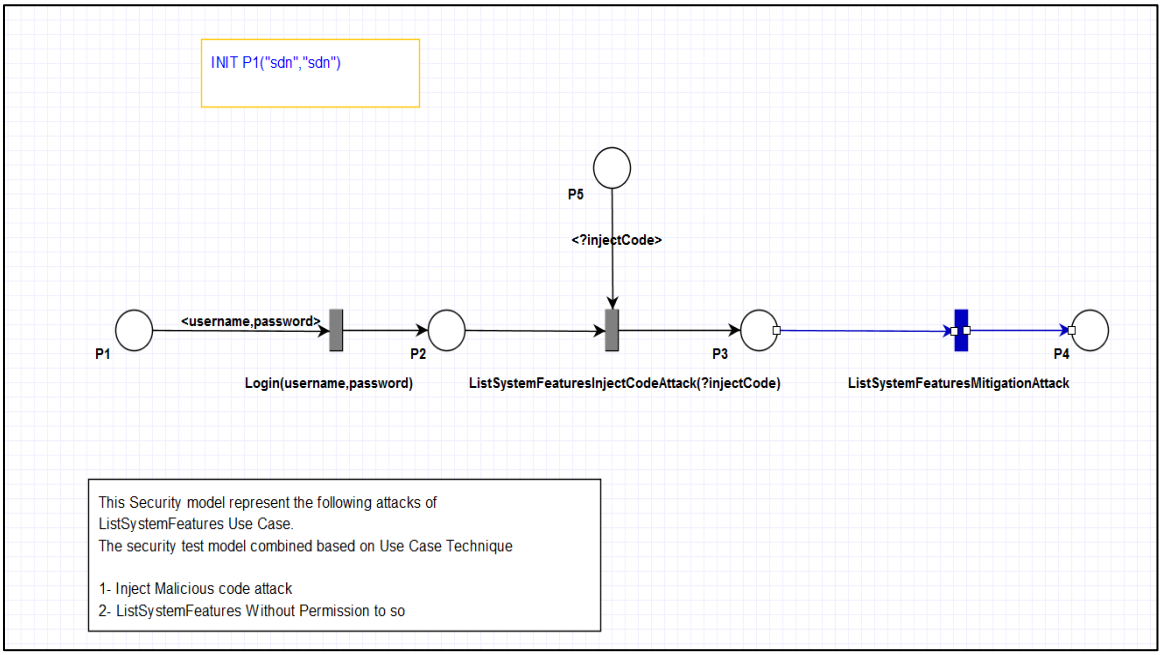

Figure 39: List System Features Use Case Security Test Model.

A.5 FileZilla Server Security Test Models Based on STRIDE Technique

A.5.1 Denial of Service Category Security Test Model

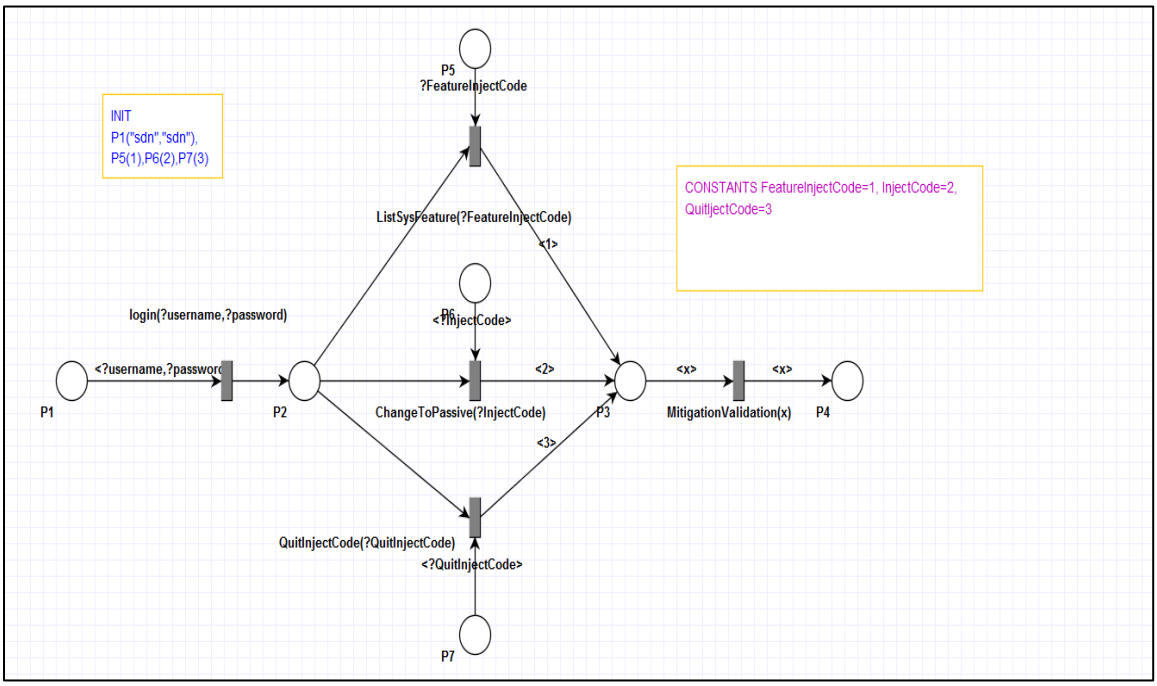

Figure 40: DoS Security Test Model Part 1.

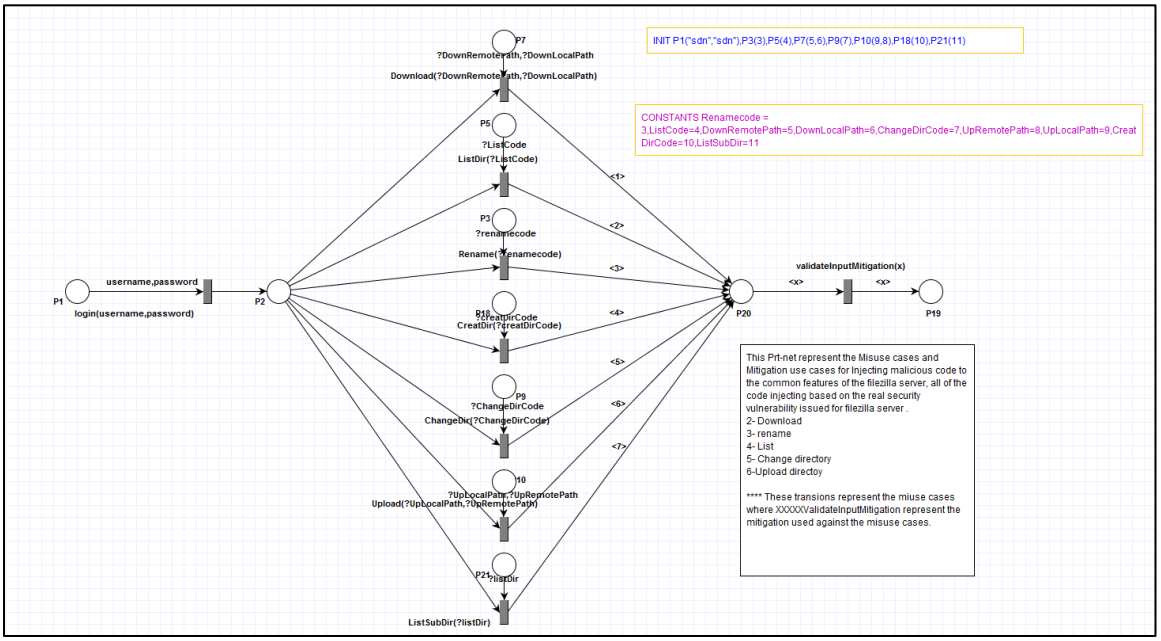

Figure 41: DoS Security Test Model Part 2

A.5.2 Privilege Elevation Category Security Test Model

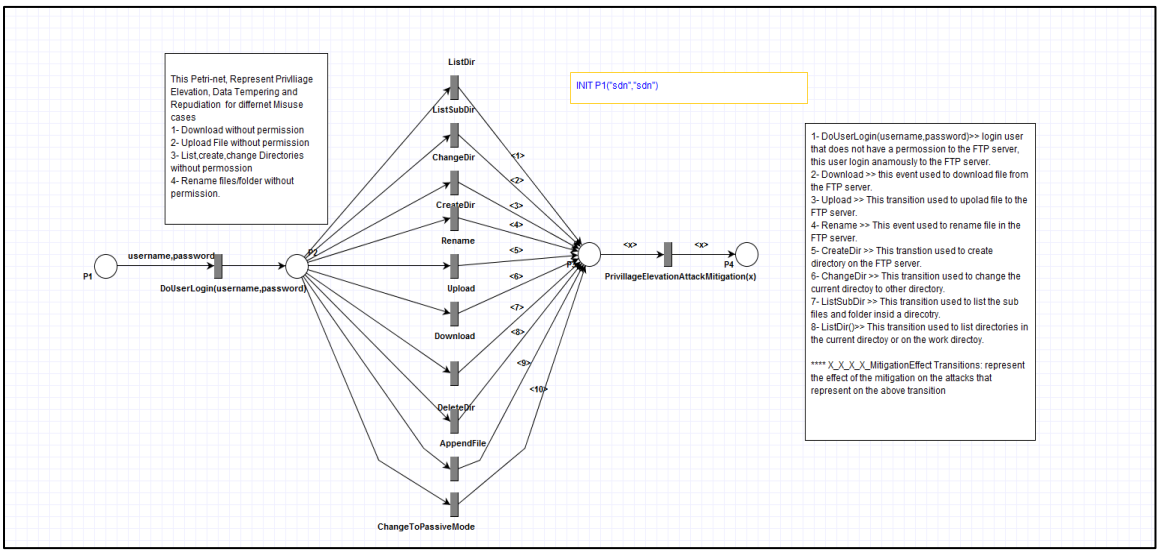

Figure 42: Privilege Elevation Security Test Model

A.5.3 Spoofing Category Security Test Model

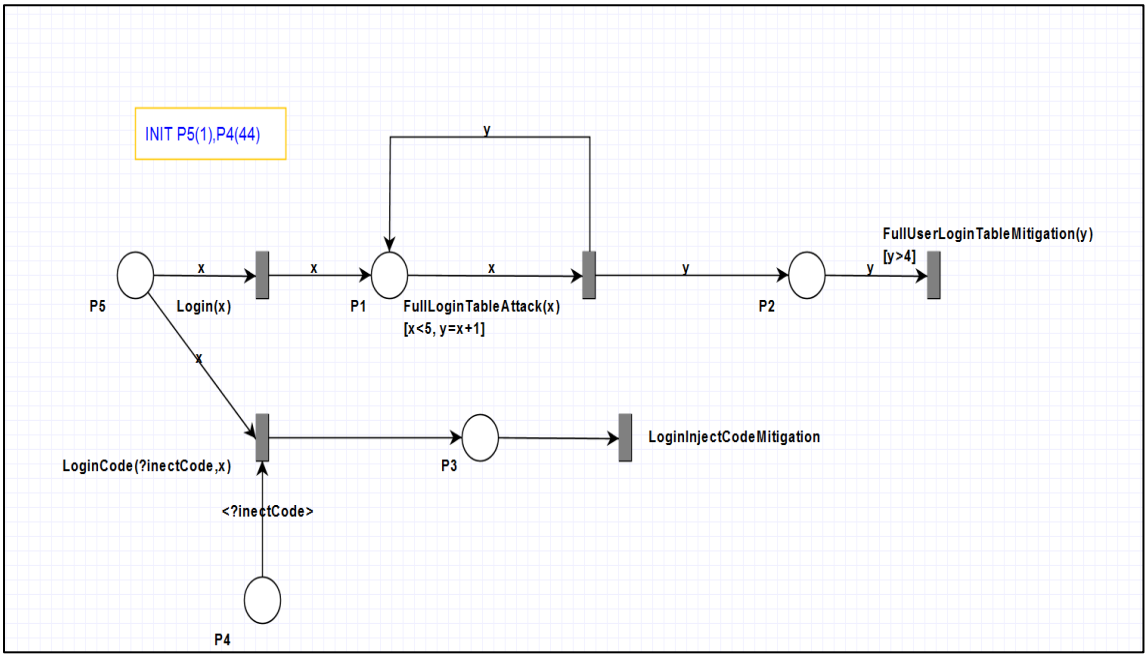

Figure 43: Spoofing Security Test Model.

APPENDIX B CASE STUDY II: GRANT PROPOSAL MANAGEMENT SYSTEM (GPMS) MISUSE CASE MODELING DATA

B.1 GPMS Use Cases

B.1.1 Signup a New User Use Case Description

Table 38: Signup a New User Use Case.

Use case #	**UC-1.**
Use case name	Create user account.
Actor	User, System admin.
Goal	To create a user account.
Preconditions	1. The user does not have an account before. 2. The system is up and running.
Main Flow	1. The actor selects "sign up" action. 2. The system redirects the actor to the signup page. 3. The actor fills the required fields and selects sign up action. 4. The system receives the sign up the request and sends a notification to the system admin and to the new user. 5. The actor selects "Manage Users" action. 6. The actor fills/verifies the User information. 7. The actor fills "User Position Details" by filling "User College", "Department", "Position Type", and "Position Title" information. 8. The system admin activates a new account. 9. The new user receives activation notification and can access his/her account.
Post-Condition	1. The user account is created successfully. 2. The user has access to the system.
Extension Points	Step 9, Notify Users Use Case
Alternative flow	1.a The **system admin Create user account.** 1.a.1 The system admin selects "Manage Users" action. 1.a.2 The System admin selects "Add new user" action. The use case continuous at *The System The system admin files/verifies the User information in the MF*
Exception flow	NONE.
Recovery flow	3.a **User selects username or e-mail address already exist.** 3.a.1 The system informs the user to use different Information.

B.1.2 Create a New Proposal Document Use Case Description

Table 39: Create a New Proposal Document Use Case.

Use case #	UC-2
Use case name	Create/ Add a Proposal Document.
Actor	The principal investigator (PI).
Goal	To create a new proposal.
Preconditions	1. The actor has an account on the system. 2. The actor job position should be tenured/ Non-tenure track faculty.
Main Flow	1. The actor login to the system. 2. The actor selects the "Add new Proposal" action. 3. The system receives the actor request and redirects the user to the new proposal page. 4. The actor fills the "Investigator Information" by filling the Co-PI and Senior Personal by selecting the "Add Co-PI" action and "Add Senior Personnel" action. 5. The actor fills the "Project Information" section. The actor fills the "Project Title, Project Type, Due Date, Project Period: From, TO Type of Request, and Location of Project" fields. 6. The actor fills the "Sponsor and Budget Information" by filling: "Name of Granting Agency, Direct Costs, Total Costs, F&A Costs, and F&A Rate" fields. 7. The actor fills "*Cost Share Information*" by filling: "*Is Institutional committed cost share included in the proposal? And Is Third Party committed cost share included in the proposal?*" fields. 8. The actor fills the "*University Commitments*" by filling: "*Will new or renovated space/facilities be required? Will rental space be required? and Does this project require institutional commitments beyond the end date of the project?*" fields. 9. The actor fills the "*Conflict of Interest and Commitment Information*" section by filling: "*Is there a financial conflict of interest related to this proposal? Has the financial conflict been disclosed? and Has there been a material change to your annual disclosure form?"* fields.

	10. The actor fills the "*Compliance Information*" section by filling: "*Does this project involve the use of Human Subjects? Does this project involve the use of Vertebrate Animals? Does this project involve Biosafety concerns? and Does this project have Environmental Health & Safety concerns?*" fields. 11. The actor fills the "*Additional Information*" section by filling: "*Do you anticipate payment(s) to foreign nationals or on behalf of foreign nationals? Do you anticipate course release time? and Are the proposed activities related to Center for Advanced Energy Studies?*" fields. 12. The actor fills the "*Collaboration Information*" section by filling: *Does this project involve non-funded collaborations?*" filed. 13. The actor fills the "*Proprietary/Confidential Information*" section by filling: "*Does this proposal contain any confidential information which is Proprietary that should not be publicly released? Will this project involve intellectual property in which the University may own or have an interest?*" fields. 14. The actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 15. The actor fills "*Appendices*" section by using the upload file action. 16. The actor selects the save action to save the proposal. 17. The system sends notifications to the Co-PI(s) and senior personal. 18. The system records the request in the user audit log
Post-Condition	1. The system saves the proposal with correct data submitted by the actor. 2. The actor can access the proposal.
Extension Points	1. Step 17, Notify Users Use Case
Alternative flow	NONE.
Exception flow	NONE.
Recovery flow	NONE.

B.1.3 Submit a Proposal by Principal Investigator (PI) Use Case Description

Table 40: Submit a Proposal by Principal Investigator (PI) Use Case.

Use case #	UC-3
Use case name	Submit a proposal by principal investigator (PI).
Actor	Principal Investigator (PI).
Goal	To submit the proposal to the department chair.
Preconditions	1. The PI created the proposal and signed it. 2. The Co-PI(s) signed the proposal. 3. The proposal status not submitted.
Main Flow	1. The actor login to account. 2. The actor selects "My proposals" action. 3. The actor selects the proposal by selecting the edit proposal action. 4. The system opens the proposal in edit mode. 5. The actor signs the proposal. 6. The actor selects the submit action. 7. The system sends a notification to the department chair, PI, Co-PI(s) and Senior Personnel. 8. The system records the request in the user audit log.
Post-Condition	1. The proposal status changed to waiting for chair approval. 2. The actor has read access to the proposal.
Extension Points	1. Step 7, Notify Users Use Case
Alternative flow	2.a The actor uses the research engine 2.a.1 The actor inserts the proposal information in the search fields. 2.a.2 The system returns the search result. 2.a.3 The actor selects the proposal. The use case continuous at *The actor selects the submit action in MF*
Exception flow	**4.a CO-PI(s) not signed the proposal** 4.a.1 The system shows an error message that CO-PIs are not signed on the proposal.
Recovery flow	NONE

B.1.4 Approve/Disapprove a Proposal by Dean Use Case Description

Table 41: Approve/Disapprove a Proposal by Dean Use Case.

Use case #	UC-4
Use case name	Approve/Disapprove a proposal by dean.
Actor	The Dean
Goal	To approve/disapprove proposal.
Preconditions	1. The proposal signed by all Business Manager. 2. The proposal approved by all Business Manager. 3. The proposal status is ready for dean approval.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor signs the proposal by filling the "Signature, Date and Note" fields. 4. The actor approves or disapproves the proposal by selecting approve/disapprove action. 5. **If** the actor approves the proposal, the system will send notifications to the PI, Co-PI, Senior Personnel and University Research Administrator, **Else**, the system will send a notification to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB. 6. The system updates the proposal status and saves it. 7. The system sends a confirmation message. 8. The system records that on the user audit log.
Post-Condition	1. If the actor approved the proposal, and the IRBs approved the proposal status changed to the ready for Research administrator approval else the status will stay ready for IRB approval. 2. If the actor disapproves the proposal, the proposal status changed to not submitted, and, clear all signatures.
Extension Points	1. Step 7, Notify Users Use Case
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2. a.2 The actor selects the proposal. The use case continuous at *The actor approves/disapproves the proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

B.1.5 Approve/Disapprove a Proposal by Research Director Use Case Description

Table 42: Approve/Disapprove a Proposal by Research Director Use Case.

Use case #	UC-5
Use case name	Approve/Disapprove a proposal by Research Director.
Actor	Research Director
Goal	To approve/disapprove the proposal.
Preconditions	1. The proposals signed by all research administrators. 2. The proposal approved by all research administrators. 3. The proposal status is ready for research director approval.
Main Flow	1. The actor login to his/her account. 2. The actor selects "My proposal" action. 3. The actor selects the proposal by selecting edit action. 4. The actor can update the "OSP" section fields in the proposal. 5. The actor signs the proposal by filling the "Signature, Date and Note" fields. 6. The actor approves or disapproves the proposal by selecting the approve/disapprove action. 7. **If** the actor approves the proposal, the system will send notifications to the PI, Co-PI, Senior Personnel, University Research Administrator, **Else**, the system will send a notification to System sends an email to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, IRB and all Deans. 8. The system updates the proposal status and saves it. 9. The system sends a confirmation message. 10. The system records that on the user audit log.
Post-Condition	1. If the actor approved the proposal, the proposal status changed to ready for search administrator submission. 2. If the actor disapproves the proposal, the proposal status changed to not submitted, and, clear all signatures.
Extension Points	1. Step 7, Notify Users Use Case
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor approve/disapprove the proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

B.1.6 Submit a Proposal by Research Administrator Use Case Description

Table 43: Submit a Proposal by Research Administrator Use Case.

Use case #	UC-6
Use case name	Submit a proposal by research administrator.
Actor	Research Administrator.
Goal	To submit the proposal.
Preconditions	1. The proposal approved by all research directories. 2. The proposals status ready for research administrator submission
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor can update the following sections of the proposal, such as "Investigator Information", "Project Information", Sponsor and Budget Information", "Cost share Information", "UniversityCommitments", "Conflict of Interest and Commitment Information"," Compliance Information", "Additional Information", "Collaboration Information", "Proprietary/Confidential Information", "Certification/Signatures", and "OSP Section". 4. The actor signs the proposal by filling the "Signature, Date and Note" fields. 5. The actor submits a proposal. 6. The system updates the proposal status and saves it. 7. The system sends the confirmation message. 8. The system sends a notification to the PI, Co-PI, Senior Personnel, Department Chair, Business manager, Dean, University Research Director and IRB. 9. The system records request on the user audit log. 10. The system records request on the system log.
Post-Condition	1. The proposal status changed to be submitted by research administrator.
Extension Points	1. Step 8, Notify Users Use Case
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects the notification tab. 2.a.2 The actor selects the proposal. The use case continuous at *The actor signs the proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

B.1.7 Withdraw a Proposal by Research Administrator Use Case Description

Table 44: Withdraw a Proposal by Research Administrator Use Case.

Use case #	UC-7
Use case name	Withdraw a proposal by Research Administrator.
Actor	Research Administrator
Goal	To withdraw the proposal.
Preconditions	1. The proposal status ready for research administrator approval.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor signs the proposal by filling the "Signature, Date and Note" fields. 4. The actor withdraws a proposal by selecting the withdraws action. 5. The system updates the proposal status and saves it. 6. The system sends the confirmation message. 7. The system sends a notification to the PI, Co-PI, Senior Personnel Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB The system records that on the user audit log. 8. The system records the request in the user audit log. 9. The system records in the system log.
Post-Condition	1. The proposal status changed to withdrawn. 2. The proposal cannot be updated by PI.
Extension Points	1. Step 7, Notify Users Use Case
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor withdraw proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

B.1.8 Delete a Proposal by Principal Investigator (PI) Uses Case Description

Table 45: Delete a Proposal by Principal Investigator (PI) Use Case.

Use case #	UC-8
Use case name	Delete a proposal by PI
Actor	the principal investigator (PI)
Goal	To delete the proposal document.
Preconditions	1. The proposal not submitted by PI.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor selects the "Delete" action. 4. The system processes the requests and deletes the proposal. 5. The system sends a confirmation message. 6. The system sends notification PI, Co-PI, Senior Personnel. 7. The system records that in user audit log 8. The system records that in the system log file.
Post-Condition	1. The system successfully deletes the proposal sheet. 2. The actor cannot find, open, and/or edit the proposal
Extension Points	1. Step 6, Notify Users Use Case
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The Actor Delete proposal in MF.*
Exception flow	NONE
Recovery flow	NONE

B.1.9 Delete a Proposal by Research Director Use Case Description

Table 46: Delete a Proposal by Research Director Use Case.

Use case #	UC-9
Use case name	Delete a proposal by Research Director
Actor	Research Director
Goal	To delete a proposal document.

Preconditions	1. The proposal status is ready for Research Director Approval.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor selects the "Delete" action. 4. The system processes the requests and deletes the proposal. 5. The system sends confirmation message. 6. The system sends notification to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB. 7. The system records that in user audit log 8. The system records that in the system log file.
Post-Condition	1. The system successfully deletes the proposal sheet. 2. The proposal status will change to deleted. 3. The PI cannot updates/edits the proposal. 4. The PI cannot have submitted again.
Extension Points	1. Step 6, Notify Users Use Case
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The Actor Delete proposal in MF.*
Exception flow	NONE
Recovery flow	NONE

B.1.10 Archive a Proposal by Research Director Use Case Description

Table 47: Archive a Proposal by Research Director Use Case.

Use case #	UC-10
Use case name	Archive a proposal by Research Director.
Actor	Research Director.
Goal	To archive the proposal.
Preconditions	1. The proposal approved by Research Administrator.
Main Flow	1. The actor login to his/her account.

	2. The actor selects the proposal by selecting edit action. 3. The actor selects the "Archive" action. 4. The system processes the requests and archives the proposal. 5. The system sends a confirmation message. 6. The system sends a notification to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB. 7. The system records that in user audit log. 8. The system records that in the system log file.
Post-Condition	1. The proposal status changed to archived 2. The proposal cannot be updated by any actor.
Extension Points	1. Step 6, Notify Users Use Case
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor selects Archive proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

B.1.11 Approve/Disapprove a Proposal by Department Chair Use Case Description

Table 48: Department Chair Approve/Disapprove a Proposal Use Case.

Use case #	UC-11.
Use case name	Approve/Disapprove a Proposal by Department Chair
Actors	The Department Chair.
Goal	Department Chair approve/disapprove proposal.
Preconditions	1. The proposal is signed by all Co-PI. 2. The proposal is signed by the PI. 3. The proposal is submitted by PI. 4. The proposal status is ready for Chair approval.
Main Flow	1. The actor logged into the system. 2. The actor selects "My proposal" action. 3. The actor selects the proposal by selecting "edit" action. 4. The actor signs the proposal by filling the signature, date, and note fields. 5. The actor approves or disapproves the proposal by selecting approve/disapprove action.

	6. **If** the actor approves the proposal, the system will send notifications to the PI, Co-PI, IRB and University Business Manager, **Else**, the system will send a notification to system sends an email to PI, Co-PI, and all Department Chairs. 7. The system updates the proposal status and saves it. 8. The system sends a confirmation message. 9. The system records that on the user audit log. 10. The system records in the system log.
Post-condition	1. If the actor approved the proposal, the proposal status will change to ready for Business Manager Approval and/or IRB. 2. If the actor disapproves the proposal, the proposal status will change to not submitted.
Extension Points	1. Step 7, Notify Users use case.
Alternative Flow	**2.a The actor uses the notification tab to select the proposal** 2.a.1 The actor opens check the notification tab 2. a.2 The actor selects the proposal. The use case continuous at *The actor approves/disapproves the proposal in MF.*
Exception Flow	NONE
Recovery Flow	NONE

B.1.12 Approve/Disapprove a Proposal by Business Manager Use Case Description

Table 49: Business Manager Approve/Disapprove a Proposal Use Case.

Use case #	**UC-12.**
Use case name	Business Manager approve/disapprove a proposal.
Actors	The Business Manager.
Goal	Business Manager Approve/Disapprove the proposal.
Preconditions	1. The proposal signed by all Department Chair. 2. The proposal approved by all Department Chair. 3. The proposal status is ready for Business Manager approval.

Main Flow	1. The actor is logged in. 2. The actor selects “My proposal” action. 3. The actor selects the proposal by selecting “edit” action. 4. The actor can edit the “Sponsor and Budget Information” section in the proposal. 5. The actor signs the proposal by filling the signature, date, and note fields. 6. The actor approves or disapproves the proposal by selecting approve/disapprove action. 7. **If** the actor approves the proposal, the system will send notifications to the PI, Co-PI, IRB and the Dean, **Else**, the system will send a notification to system sends an email to PI, Co-PI, Department chair, IRB, and all Business Managers. 8. The system updates the proposal status and saves it. 9. The system sends a confirmation message. 10. The system records that on the user audit log. 11. The system records in the system log.
Post-condition	1. If the actor approved the proposal, the proposal status will change to ready for Dean’s approval. 2. If the actor disapproves the proposal, the proposal status will change to not submitted.
Extension Points	1. Step 7, Notify Users use case.
Alternative Flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor opens check the notification tab 2. a.2 The actor selects the proposal. The use case continuous at *The actor approves/disapproves the proposal in MF.*
Exception Flow	NONE
Recovery Flow	NONE

B.1.13 Export to Excel Sheet Use Case Description

Table 50: Export to Excel Sheet Use Case.

Use case #	**UC-13.**
Use case name	Export proposals to excel sheet.
Actors	The User.
Goal	To Export to Excel.
Preconditions	1. The actor has proposals.
Main Flow	1. The actor login to the system. 2. The actor selects “My Proposal” action. 3. The actor selects “Export to Excel” action. 4. The excel file will start downloading.

	5. The system records that on the user audit log. 6. The system records in the system log.
Post-condition	1. The file exported to excel. 2. The excel file has the correct information of the proposals.
Extension Points	NONE
Alternative Flow	NONE.
Exception Flow	NONE.
Recovery Flow	NONE.

B.1.14 Update User Personal Information Use Case Description

Table 51: Update User Personal Information Use Case.

Use case #	**UC-14.**
Use case name	Update user personal information.
Actors	The user, the system Admin.
Goal	To update user personal information.
Preconditions	1. The actor has an account on the system. 2. The actor account is activated.
Main Flow	1. The actor login to the system. 2. The actor selects "Account Settings" action. 3. The system processes the request and redirects the user to user' account information page. 4. The actor selects the "General Information" tab and updates the fields for each section, such as "User Information", "Current Address", "Phone", and "E-mail Address" sections. 5. The actor selects "User Position Details" tab, and update the required fields. 6. The actor selects "User Login Credentials" tab, and update the required fields. 7. The actor selects "Audit Logs" tab, and check Actions and Audit logs. 8. The actor selects save action. 9. The system sends a confirmation message. 10. The system logs the user's request in the Audit Log.
Post-condition	1. The system saves the updated personal information.
Extension Points	NONE
Alternative Flow	**1.a The system admin Updates user personal information.** 1.a.1 The system admin selects "Manage Users" action. 1.a.2 The system admin selects "edit", The use case continues at *The system opens the user' account information page in the MF.* **2.a The Use Update user personal information.**

	2.a.1 The user selects “My Account” from the drop menu. The use case continues at *The system opens the user’ account information page in the MF.*
Exception Flow	NONE.
Recovery Flow	**3.a The user insert an exist/invalid email** 3.a.1 The system shows an error message. 3.a. 2 The actor inserts another valid email address.

B.1.15 Approve/Disapprove a Proposal by IRB Use Case Description

Table 52: IRB Approve/Disapprove a Proposal Use Case.

Use case #	**UC-15**
Use case name	IRB approve/disapprove proposal.
Actors	The IRB.
Goal	Business Manager approve/disapprove proposal.
Preconditions	1. The proposal status is ready for IRB approval. 2. The proposal has a compliance
Main Flow	1. The actor is logged in. 2. The actor selects “My proposal” action. 3. The actor selects the proposal by selecting “edit” action. 4. The actor signs the proposal by filling the signature, date, and note fields. 5. The actor approves or disapproves the proposal. 6. If the actor approves the proposal, the system will send notifications to the PI, Co-PI, Senior Personnel and Research Administrator, Else, the system will send notification to system sends an email to PI, Co-PI, and all Department chair. 7. The system updates the proposal status and saves it. 8. The system sends confirmation message. 9. The system records that on the user audit log.
Post-condition	1. If the actor approved the proposal and the Deans approved, the proposal status will change to the ready for Research Administrator’s approval else will remain ready for Dean approval. 2. If the actor disapproves the proposal, the proposal status will change to not submitted.
Extension Points	1. Step 6, Notify Users use case.
Alternative Flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor opens check the notification tab. 2. a.2 The actor selects the proposal. The use case continuous at *The actor approves/disapproves the proposal in MF.*
Exception Flow	NONE.
Recovery Flow	NONE.

B.1.16 Approve/Disapprove a Proposal by Research Administrator Use Case

Description

Table 53: Approve/Disapprove a Proposal by Research Administrator Use Case.

Use case #	**UC-16**
Use case name	Approve/Disapprove a proposal by Research Administrator.
Actor	The Research Administrator.
Goal	To approve/disapprove the proposal.
Preconditions	1- The proposal status is ready for Research Administrator.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor signs the proposal by filling the signature, date, and note fields. 4. The actor can update the following sections of the proposal, such as "Investigator Information", "Project Information", Sponsor and Budget Information", "Cost share Information", "UniversityCommitments", "Conflict of Interest and Commitment Information"," Compliance Information", "Additional Information", "Collaboration Information", "Proprietary/Confidential Information", "Certification/Signatures", and "OSP Section". 5. The actor approves or disapproves the proposal. 6. **If** the actor approves the proposal, the system will send notifications to the PI, Co-PI, Senior Personnel and University Research Director, **Else**, the system will send a notification to PI, Co-PI, Senior Personnel, Department Chair, Business Manager, Dean, University Research Administrator, University Research Director and IRB. 7. The system updates the proposal status and saves it. 8. The system sends a confirmation message. 9. The system records that on the user audit log. 10. The system records in the system log.
Post-Condition	1. If the actor approved the proposal, the proposal status changed to ready for Research Director approval. 2. If the actor disapproves the proposal, the proposal status changed to not submitted, and, clear all signatures.
Extension Points	1. Step 6, Notify Users use case.
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2. a.2 The actor selects the proposal. The use case continuous at The actor approve/disapprove the proposal in MF.
Exception flow	NONE.

Recovery flow	NONE.

B.1.17 GPMS User Login Use Case Description

Table 54: GPMS Login User Use Case.

Use case #	**UC-17**
Use case name	User Login.
Actor	User/ Admin.
Goal	To login to the system.
Preconditions	1. The actor has an account.
Main Flow	1. The actor selects the login page. 2. The system redirects to the login page. 3. The actor fills in the Email/Username field. 4. The actor fills in the Password field. 5. The actor selects the "Login" action. 6. The system redirects to actor's account page. 7. The system records that on the user audit log. 8. The system records in the system log.
Post-Condition	1. The actor successfully logs into the system. 2. The system redirects the user to the user home page.
Extension Points	NONE
Alternative flow	NONE.
Exception flow	NONE.
Recovery flow	NONE.

B.1.18 Sign a Proposal by CO-PI Use Case Description

Table 55: Sign a Proposal by CO-PI Use Case.

Use case #	**UC-18**
Use case name	CO-PI signs a proposal.
Actor	CO-PI.
Goal	CO-PI signs the proposal.
Preconditions	1. The CO-PI is added to the proposal by PI.
Main Flow	1. The actor login to his/her account. 2. The actor selects the proposal by selecting edit action. 3. The actor can update "Investigator Information" section in the proposal. 4. The actor signs the proposal by filling the signature, date, and note fields.

	5. The system updates the proposal status and saves it. 6. The system sends a confirmation message. 7. The send notification to the PIs and CO_PIs 8. The system records that on the user audit log. 9. The system records in the system log.
Post-Condition	1. The proposal status changed to ready to submit by PI.
Extension Points	2. Step 7, Notify Users use case.
Alternative flow	2.a The **actor uses the notification tab to select the proposal** 2.a.1 The actor selects check the notification tab 2.a.2 The actor selects the proposal. The use case continuous at *The actor signs the proposal in MF.*
Exception flow	NONE.
Recovery flow	NONE.

B.1.19 Notify Users Use Case Description

Table 56: Notify Users Use Case

Use case #	**UC-19**
Use case name	Notify Users
Actor	System
Goal	To notify users about the proposal document updates
Preconditions	1. A Proposal document changed to another status
Main Flow	1. The actor modifies to a proposal document. 2. The system receives the new proposal document changes. 3. The system sends notifications to the users according to on the notification policy.
Post-Condition	1. The actors receive the notification of the latest updates of a proposal document.
Extension Points	NONE
Alternative flow	NONE
Exception flow	NONE
Recovery flow	NONE

B.2 GPMS Misuse Cases

B.2.1 URL Redirection Misuse Case Description

Table 57: URL Redirection Misuse Case.

Misuse case #	**MUC-1**
Misuse case name	URL Redirection.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To redirect the user to other website and steal user information.
Actor	System, adversary.
Preconditions	2. The system is up and running.
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens/creates a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The system will redirect the user to the malicious site.
Threat point	2. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 3. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 4. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 5. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 6. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 7. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 8. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 9. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.2 Disable Edit a Proposal Document Action Misuse Case Description

Table 58: Disable Edit a Proposal Action Misuse Case.

Misuse case #	**MUC-2**
Misuse case name	Disable edit proposal document action.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To prevent the user from accessing to the proposal document.
Actor	Adversary, system
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens/creates a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the save, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The actor cannot open the proposal document.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention

B.2.3 Disable a Proposal Save Action Misuse Case Misuse Case Description

Table 59: Disable a Proposal Save Action Misuse Case.

Misuse case #	**MUC-3**
Misuse case name	Disable Save proposal action.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To prevent save the proposal.
Actor	Adversary
Preconditions	1. The actor has an account on the system or has other user login information.
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens/creates a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The user cannot save the proposal.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.4 Disable Approve /Disapprove Actions Misuse Case Description

Table 60: Disable Approve/Disapprove Actions Misuse Case.

Misuse case #	**MUC-4**
Misuse case name	Disable Approve/Disapprove proposal actions.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To prevent approve/disapprove the proposal.
Actor	Adversary.
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens/creates a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The user cannot approve/disapprove the proposal.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.5 Display/Print out User ID Misuse Case Description

Table 61: Display/Print out User ID Misuse Case.

Misuse case #	**MUC-5**
Misuse case name	Display user ID attack.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To display user ID.
Actor	Adversary.
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The used system ID printout on the signature section.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.6 Destroy a Proposal Document Web Page Misuse Case Description

Table 62: Destroy a Proposal Document Web Page Misuse Case.

Misuse case #	**MUC-6**
Misuse case name	Destroy the proposal document.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To prevent save the proposal.
Actor	Adversary.
Preconditions	1. To destroy content of the proposal document
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The proposal document layout destroyed.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.7 Access to Admin Site Misuse Case Description

Table 63: Access to Admin Site Misuse Case.

Misuse case #	**MUC-7**
Misuse case name	Access to the system admin account.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To access the admin site.
Actor	Adversary.
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. Access to the admin site.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.8 Disable Proposals View List Misuse Case Description

Table 64: Disable Proposals View List Misuse Case.

Misuse case #	**MUC-8**
Misuse case name	Hide/disable all proposal attack misuse cases
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To display user ID.
Actor	Adversary.
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The legitimate user proposal cannot be displayed.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.9 Unauthorized Update Proposal Fields Misuse Case Description

Table 65: Unauthorized Update Proposal Fields Misuse Case.

Misuse case #	**MUC-9**
Misuse case name	Unauthorized proposal updates section fields.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To update proposal without permission validation
Actor	Adversary.
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor is signed into the system. 2. The actor selects "My Proposals" action. 3. The actor selects the proposal by selecting "Edit" action. 4. The actor updates non-editable sections fields by executing javascript code. 5. The actor selects Approve, Disapprove or submits action.
Post-conditions	1. The updated section fields successfully updated and saved.
Threat point	1. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 2. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 3. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 4. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 5. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 6. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 7. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.10 Disable All Proposal Actions Misuse Case Description

Table 66: Disable All Proposal Actions Misuse Case.

Misuse case #	**MUC-11**
Misuse case name	Disable all proposal actions
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To disable the actions of proposal document
Actor	Adversary.
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The user cannot use any action from the proposal document.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.11 Disable Submit Proposal Action Misuse Case Description

Table 67: Disable Submit Proposal Action Misuse Case.

Misuse case #	**MUC-11**
Misuse case name	Disable "Submit" proposal actions.
Misuse case category	Information Disclosure, DoS and elevation of privilege.
Goal	To prevent the proposal submission.
Actor	Adversary.
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor login to the system. 2. The actor selects my proposal action. 3. The actor opens a proposal document by select edit action. 4. The actor injects the malicious code in the signature field. 5. The actor selects one of the saves, submit, approve or disapprove proposal actions according to the user privilege and proposal phase.
Post-conditions	1. The user cannot submit the proposal.
Threat point	1. Create/add a proposal, Main flow, Step 14, the actor fills the "*Certification/Signatures*" section by filling: *"Signature(s), Date and Note*" fields. 2. Approve/Disapprove proposal by Dean, Main flow, step 3, The actor sign the proposal. 3. Approve/Disapprove proposal by Research Director, Main flow, Step 4, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields. 4. Submit the proposal by Research Administrator, Main flow, Step 3, The actor sign the proposal by filling: *"Signature(s), Date and Note*" fields 5. Department Chair approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields. 6. Business Manager approves/disapprove the proposal, Main Flow, Step 5, The actor signs the proposal by filling the signature, date, and note fields. 7. Approve/Disapprove proposal by Research Administrator, Main Flow, Step 3, The actor selects the proposal by selecting edit action. 8. IRB approve/disapprove the proposal, Main Flow, Step 4, The actor signs the proposal by filling the signature, date, and note fields.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.12 Access to Admin Account by Using Signup Action Misuse Case Description

Table 68: Access to Admin Account by Using Signup Action Misuse Case.

Misuse case #	**MUC-12**
Misuse case name	Access to Admin Accounts by Using Signup Action
Misuse case category	Spoofing, DoS and Elevation of privilege.
Goal	To access the admin site
Actor	Adversary.
Preconditions	1. The actor has account on the system or has other user login information
Main Flow	1. The actor selects signup page. 2. The actor injects crafted malicious code in the username field. 3. The actor selects signup action. 4. The actor selects the link of the system admin web page. 5. The access to the Admin Account.
Post-conditions	1. Access the admin dashboard.
Threat point	1. Create User account/ signup new user, Main Flow, Step 3, The actor fills the required fields and selects sign up action.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.13 Unauthorized Delete User Account Misuse Case Description

Table 69: Unauthorized Delete User Account Misuse Case.

Misuse case #	**MUC-13**
Misuse case name	Unauthorized delete user account
Misuse case category	Spoofing, DoS and elevation of privilege.
Goal	To delete system users accounts.
Actor	Adversary.
Preconditions	1. The system is up and running.
Main Flow	1. The actor selects signup page. 2. The actor injects crafted malicious code in the username field. 3. The actor selects signup action. 4. The actor selects the link of the system admin web page. 5. The access to the Admin Account.
Post-conditions	1. To access the admin site.
Threat point	1. Create User account/ signup new user, Main Flow, Step 3, The actor fills the required fields and selects sign up action.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.14 Unauthorized Activate/Deactivate User Accounts Misuse Case Description

Table 70: Unauthorized Activate/Deactivate User Accounts Misuse Case.

Misuse case #	**MUC-14**
Misuse case name	Unauthorized activate/deactivate user accounts.
Misuse case category	Spoofing, DoS and elevation of privilege.
Goal	To activate/deactivate user accounts.
Actor	Adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor selects signup page. 2. The actor injects crafted malicious code in the username field. 3. The actor selects signup action. 4. The actor selects the link of the system admin web page. 5. The actor activates/deactivate users.
Post-conditions	1. The user account activated/deactivated.
Threat point	1. Create User account/ signup new user, Main Flow, Step 3, The actor fills the required fields and selects sign up action.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.15 Unauthorized Modification of User Proposals Misuse Case Description

Table 71: Unauthorized Modification Users Proposal Misuse Case

Misuse case #	**MUC-15**
Misuse case name	Unauthorized Modification Users Proposal fields.
Misuse case category	Spoofing, Privilege Elevation and information disclosure.
Goal	To update proposal without permission validation
Actors	System, adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor selects signup page. 2. The actor injects crafted malicious code in the username field. 3. The actor selects signup action. 4. The actor selects the link of the system admin web page 5. The actor modifies user's proposals.
Post-conditions	1. The user proposals updated successfully.
Threat point	1. Create User account/ signup new user, Main Flow, Step 3, The actor fills the required fields and selects sign up action.
Mitigation	Server side trusted data for driving access control decisions mitigation.

B.2.16 Unauthorized Delete User Proposals Misuse Case Description

Table 72: Unauthorized Delete User Proposals Misuse Case.

Misuse case #	**MUC-16**
Misuse case name	Unauthorized delete user proposals
Misuse case category	Spoofing, DoS and elevation of privilege.
Goal	To delete user proposals
Actor	Adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor selects signup page. 2. The actor injects crafted malicious code in the username field. 3. The actor selects signup action. 4. The actor selects the link of the system admin web page 5. The actor deletes user proposal document.
Post-conditions	1. The user account Activate/deactivate.
Threat point	1. Create User account/ signup new user, Main Flow, Step 3, The actor fills the required fields and selects sign up action.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.17 Unauthorized Modifying User Accounts Misuse Case Description

Table 73: Unauthorized Modifying User Accounts Information Misuse Case.

Misuse case #	**MUC-17**
Misuse case name	Unauthorized Modifying User Accounts Information.
Misuse case category	Spoofing, DoS and elevation of privilege.
Goal	To modifying user accounts information
Actor	Adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor selects signup page. 2. The actor injects crafted malicious code in the username field. 3. The actor selects signup action. 4. The actor selects the link of the system admin web page 5. The actor modifies user's information.
Post-conditions	1. The user account information updated successfully.
Threat point	1. Create User account/ signup new user, Main Flow, Step 3, The actor fills the required fields and selects sign up action.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.18 Unauthorized Delete All Users Accounts Misuse Case Description

Table 74: Unauthorized Delete All Users Accounts Misuse Case.

Misuse case #	**MUC-18**
Misuse case name	Unauthorized delete All User Accounts.
Misuse case category	Spoofing, DoS and elevation of privilege.
Goal	To delete all user accounts
Actor	Adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor selects signup page. 2. The actor injects crafted malicious code in the username field. 3. The actor selects signup action. 4. The actor selects the link of the system admin web page 5. The actor deletes all system user's actions.
Post-conditions	1. The user's accounts deleted successfully.
Threat point	1. Create User account/ signup new user, Main Flow, Step 3, The actor fills the required fields and selects sign up action.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.19 Unauthorized Delete All Proposals Misuse Case Description

Table 75: Unauthorized Delete All Proposals Misuse Case.

Misuse case #	**MUC-19**
Misuse case name	Unauthorized delete all user's proposals documents
Misuse case category	Spoofing, DoS and elevation of privilege.
Goal	To delete all user proposals document.
Actor	Adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor selects signup page. 2. The actor injects crafted malicious code in the username field. 3. The actor selects signup action. 4. The actor selects the link of the system admin web page. 5. The actor deletes all proposal documents.
Post-conditions	1. The all user proposal deleted successfully.
Threat point	1. Create User account/ signup new user, Main Flow, Step 3, The actor fills the required fields and selects sign up action.
Mitigation	Cross-site scripting (XSS) prevention.

B.2.20 Unauthorized Download File Misuse Case Description

Table 76: Unauthorized Download File Misuse Case.

Misuse case #	**MUC-20**
Misuse case name	Unauthorized download files.
Misuse case category	DoS, Information Disclosure
Goal	To download proposals document files.
Actor	Adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor craft the HTTP header request by adding the name of the file to be downloaded. 2. The actor sends the HTTP request to the system to download the file.
Post-conditions	1. The file downloaded successfully.
Threat point	1. Add/Create a proposal, Main Flow, Step 15, The actor fills "Appendices" section by using the upload file action.
Mitigation	Prevent file path injection mitigation use case

B.2.21 Upload Dangerous Contents Misuse Case Description

Table 77: Upload Dangerous Contents to the System Misuse Case.

Misuse case #	**MUC-21**
Misuse case name	Upload malicious files to the system
Misuse case category	DoS.
Goal	To upload malicious files.
Actor	Adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor login to the system. 2. The select the "My Proposals" actions. 3. The actor opens proposal document by selecting edit proposal action 4. The actor selects the "*Appendices"* section 5. The actor uploads the malicious file by selecting the upload action. 6. The actor selects the save action.
Post-conditions	1. The file uploaded successfully.
Threat point	1. Add/Create a proposal, Main Flow, Step 15, The actor fills "Appendices" section by using the upload file action.
Mitigation	Validate file extension and content mitigation use case.

B.2.22 Upload Large Files DoS Misuse Case Description

Table 78: Upload Large Files DoS Misuse Case.

Misuse case #	**MUC-22**
Misuse case name	Upload large files to the system
Misuse case category	Denial of service (DoS).
Goal	To upload large files that consume the I/O disk space.
Actor	Adversary.
Preconditions	1. The system is up and running
Main Flow	1. The actor login to the system. 2. The select the "My Proposals" actions. 3. The actor opens proposal document by selecting edit proposal action 4. The actor selects the "*Appendices"* section 5. The actor uploads the large file by selecting the upload action. 6. The actor selects the save action.
Post-conditions	1. The file uploaded successfully.
Threat point	1. Add/Create a proposal, Main Flow, Step 15, The actor fills "Appendices" section by using the upload file action.
Mitigation	Limiting the uploading file size per user mitigation use case.

B.2.23 Overwrite Uploaded Files Misuse Case Description

Table 79: Overwrite Uploaded Files Misuse Case.

Misuse case #	**MUC-23**
Misuse case name	Overwrite uploaded files.
Misuse case category	Denial of service (DoS).
Goal	To overwrite uploaded files with other files
Actors	Adversary.
Preconditions	1. The proposal status is not submitted by PI. 2. The actor upload files with the same name of the files that already uploaded. 3. The actor has access to the system.
Main Flow	1. The actor login to the system. 2. The select the "My Proposals" actions. 3. The actor opens proposal document by selecting edit proposal action 4. The actor selects the "*Appendices"* section 5. The actor uploads the already exists file by selecting the upload action.

	6. The actor selects the save action.
Post-conditions	1. The file being uploaded, uploaded to the system and overwrite the existence one.
Threat point	1. Add/Create a proposal, Main Flow, Step 15, The actor fills "Appendices" section by using the upload file action.
Mitigation	Rename uploaded file to the system mitigation use case

B.2.24 Automatic Users Registration Misuse Case Description

Table 80: Automatic Users Registration Misuse Case.

Misuse case #	**MUC-24**
Misuse case name	Automatic register user accounts
Misuse case category	Tempering, Denial of service attack (DoS).
Goal	To automatically create malicious user accounts and/or crash the service.
Actor	System, adversary.
Preconditions	1. The system is up and running.
Main Flow	1. The actor access to the system login page. 2. The actor automatically and repeatedly does: 2.1 The actor selects the SingUp page link. 2.2 The actor fills the required fields. 2.3 The actor selects SignUp action.
Post-conditions	1. Create malicious users accounts. 2. The signup process will be suspended or crashed.
Threat point	1. Create user account, Main flow, Step 4, The system receives the signup request.
Mitigation	Honey Token Form component mitigation use case

B.2.25 Username Harvesting Misuse Case Misuse Case Description

Table 81: Username Harvesting Misuse Case.

Misuse case #	**MUC-25**
Misuse case name	Username harvesting
Misuse case category	Spoofing
Goal	To validate the correctness of the username
Actors	System, adversary.
Preconditions	1. The system is up and running.

Main Flow	1. The actor access to the login page. 2. The actor inserts the username and wrong password. 3. The actor validates the error message that issued by the system.
Post-conditions	1. Validate the correctness of the username.
Threat point	1. Login, Main Flow, Step2, User insert the username and password.
Mitigation	Prevent harvesting the username and/or password mitigation use case.

B.2.26 Unauthorized Proposal Approve/Disapprove by CO-PI Misuse Case Description

Table 82: Unauthorized proposal approve/ disapprove by CO-PI Misuse Case.

Misuse case #	**MUC-26**
Misuse case name	Unauthorized proposal approve/ disapprove by CO-PI.
Misuse case category	Privilege elevation attack, DoS.
Goal	To approve/disapprove the proposal without permission validation.
Actors	System, adversary.
Preconditions	1. The proposal is ready for CO-PI submission.
Main Flow	1. The actor is signed in as CO-PI. 2. The actor selects “My Proposal” action. 3. The actor selects the proposal by selecting “Edit” action. 4. The actor signs the proposal. 5. The actor selects “Approve” action
Post-conditions	1. The Proposal status changed to be approved by CO_PI.
Threat point	1. Submit proposal by CO_PI, Main flow, Step 5, The actor submits the proposal.
Mitigation	Server side trusted data for driving access control decisions mitigation use case.

B.2.27 Unauthorized Submission by Research Director Misuse Case Description

Table 83: Unauthorized a Proposal Submission by Research Director Misuse Case.

Misuse case #	**MUC-27**
Misuse case name	Unauthorized Submission by Research Director

Misuse case category	Privilege elevation, DoS.
Goal	To submit a proposal without authorization.
Actors	System, adversary.
Preconditions	1. The proposal status is waiting for research director approval.
Main Flow	1- The actor is signed in as Research Director. 2- The actor selects “My Proposals” action. 3- The actor selects the proposal by selecting “Edit” action. 4- The actor signs the proposal. 5- The actor selects “submit” section.
Post-conditions	1- The proposal status changed to be submitted by research director.
Threat point	1. Approve/Disapprove proposal by Research Director, Main flow, Step 5, The actor approve or disapprove the proposal by selecting approve/disapprove action.
Mitigation	Server side trusted data for driving access control decisions mitigation

B.2.28 Unauthorized Archive a Proposal by Business Manager Misuse Case Description

Table 84: Unauthorized Archive a Proposal by BM Misuse Case.

Misuse case #	**MUC-28**
Misuse case name	Unauthorized archive a proposal by BM.
Misuse case category	Privilege elevation, DoS
Goal	To archive proposal without authorization.
Actors	System, adversary.
Preconditions	1. The proposal status is ready for BM approval.
Main Flow	1. The actor is signed in as BM. 2. The actor selects “My Proposals” action. 3. The actor selects the proposal by selecting “Edit” action. 4. The actor signs the proposal. 5. The actor selects “archive” action.
Post-conditions	1. The proposal status changed to be archive by BM.
Threat point	1. Approve/disapprove a proposal by BM, Main flow, Step 5, The actor selects the approve/disapprove action.
Mitigation	Server side trusted data for driving access control decisions mitigation u

B.2.29 Unauthorized Delete a Proposal by Department Chair Misuse Case Description

Table 85: Unauthorized Delete a Proposal by Dept. Chair Misuse Case.

Misuse case #	**MUC-29**
Misuse case name	Unauthorized Delete of a proposal by department Chair.
Misuse case category	Privilege elevation attack, DoS.
Goal	To delete proposal without authorization.
Actors	System, adversary.
Preconditions	1. The proposal status is waiting for Chair's approval.
Main Flow	1- The actor is signed in as Chair. 2- The actor selects "My Proposals" action. 3- The actor selects the proposal by selecting "Edit" action. 4- The actor signs the proposal. 5- The actor selects "Delete" action.
Post-conditions	1- The proposal status changed to be deleted. 2- The proposal document cannot be submitted by PI one more time.
Threat point	1. Approve/Disapprove proposal by Department Chair, Main flow, Step 5, The actor approve or disapprove the proposal by selecting approve/disapprove action.
Mitigation	Server side trusted data for driving access control decisions mitigation.

B.2.30 Unauthorized Withdraw a Proposal by PI Misuse Case Description

Table 86: Unauthorized Withdraw a Proposal by PI Misuse Case

Misuse case #	**MUC-30**
Misuse case name	Unauthorized withdraw of a proposal by PI.
Misuse case category	Privilege elevation attack, DoS.
Goal	To submit a proposal without authorization.
Actors	System, adversary.
Preconditions	1. The proposal status is waiting for research director approval.
Main Flow	1. The actor is signed in as PI. 2. The actor selects "My Proposals" action. 3. The actor selects the proposal by selecting "Edit" action. 4. The actor signs the proposal. 5. The actor selects "withdraw" section.
Post-conditions	1. The proposal status changed to be withdrawn by PI.
Threat point	2. submit a proposal by PI, Main flow, Step 5, The actor submits or proposal.

Mitigation	Server side trusted data for driving access control decisions mitigation.

B.3 GPMS Mitigation Use Cases

B.3.1 Cross-Site Scripting Attack Mitigation Use Case Description

Table 87: Cross-Site Scripting Attack Mitigation Use Case.

Mitigation use case #	**MITI-UC 1**
Mitigation use case name	Cross-site scripting (XSS) prevention.
Goal	To prevent the XSS attack
Precondition	1. The actor injects malicious script code. 2. The actor uploads malicious script code.
Main Flow	1. Validate user input by using whitelist technique. 2. Use Output Escaping technique to ensure any JavaScript code is converted to safe display. 3. Use HTML escape JSON values and decode the JSON values and safely parse it.
Post Condition	1. The XSS attack malicious code will not be executed. 2. The application will render the web pages safely and appropriately. 3. The application protects the user data.
Priority	High

B.3.2 Validate File Extension and Content Mitigation Use Case Description

Table 88: Validate File Extension and Content Mitigation Use Case.

Mitigation use case #	**MITI-UC 2**
Mitigation use case name	Validating file extensions and contents.
Goal	To prevent uploading dangerous files to the system.
Precondition	1. The actor upload files to the system.
Main Flow	1. Validate the uploading file extension by using whitelist technique. 2. Validate the contents of the uploaded file by using antivirus.
Post Condition	1. The application prevents uploading files with a malicious extension such as. JSP. 2. The application prevents uploading files with malicious contents with the correct extension.
Priority	High

B.3.3 Limiting the Uploading File Size Mitigation Use Case Description

Table 89: Limiting the Uploading File Size Mitigation Use Case.

Mitigation use case #	**MITI-UC 3**
Mitigation use case name	Limiting the uploading file size per user.
Goal	To prevent uploading large file size to the system.
Precondition	1. The actor upload files to the system.
Main Flow	1. Validating the file size with the available system disk space. 2. Compare the file size with remaining disk space for the user. 3. Validate the number of uploading files for the user.
Post Condition	1. Prevent the actor from an uploading large number of files. 2. Prevent the actor from uploading large file size.
Priority	High.

B.3.4 Rename Uploaded File to the System Mitigation Use Case Description

Table 90: Rename Uploaded File to the System Mitigation Use Case.

Mitigation use case #	**MITI-UC 4**
Mitigation use case name	Rename uploaded files.
Goal	To prevent files, overwrite attack.
Precondition	1. The actor uploads the file to the system.
Main Flow	Use random function to generate new file name Concatenate the generated random name with characters. Rename the file being uploaded with new random name.
Post Condition	1. The uploaded file has the new random name.
Priority	Medium

B.3.5 Prevent File Path Injection Mitigation Use Case Description

Table 91: Prevent File Path Injection Mitigation Use Case.

Mitigation use case #	**MITI-UC 5**
Mitigation use case name	Prevent file path injection

Goal	To prevent read resources inside and outside the web application folder.
Precondition	1. The actor injects malicious parameter in the request header.
Main Flow	1. Probably canonical and validating any given file path. 2. Validate the user session if it still alive or not. 3. Validate the user permission before accessing to any resource. 4. Use the minimum privilege for accessing the resources.
Post Condition	1. Prevent accessing to the system resources. 2. Prevent downloading resource without user validation.
Priority	High

B.3.6 Honey Token Form Component Mitigation Use Case Description

Table 92: Honey Token Form Component Mitigation Use Case.

Mitigation use case #	**MITI-UC 6**
Mitigation use case name	Honeytoken form component.
Goal	To prevent automatic user registration.
Precondition	1. The actor executes automatic user registration script.
Main Flow	1. Add a field to the user sign up sensitive form. 2. Hide this field with CSS and make it invisible to the normal users and visible to the automation registration scripts. 3. The value of the hidden file will empty for the normal registration and non-empty for the automatic one. 4. Validate the hidden field value for each sign-up request on the server side. 5. Throw a message if the field values not empty and stop the sign up the process.
Post Condition	1. Discover the automatic registration request from normal ones. 2. Prevent the automatic registration request.
Priority	High.

B.3.7 Prevent Harvesting the Username Mitigation Use Case Description

Table 93: Prevent Harvesting The Username Mitigation Use Case.

Mitigation use case #	**MITI-UC 7**
Mitigation use case name	Prevent username and/or password harvesting
Goal	To prevent harvesting the username or password from the login error message.

Precondition	The actor keeps using different username or password
Main Flow	1. The validate the username and password for each login request. 2. The system throws the same error message in both cases if the username or password are incorrect.
Post Condition	1. Prevent the actor to validate the correctness of the username or password.
Priority	Medium.

B.3.8 Server side for Driving ACL Mitigation Use Case Description

Table 94: Server Side Trusted Data for Driving ACL Mitigation Use Case.

Mitigation use case #	**MITI-UC 8**
Mitigation use case name	Server side trusted to drive access control decision.
Goal	To prevent using client request data to make access control decision.
Precondition	1. The actor updates unauthorized proposal section. 2. The actor requests unauthorized proposal actions. 3. The actor requests modifying and accessing to unauthorized application resources.
Main Flow	1. Using trusted session to verify and retrieve user identity. 2. Using server side trusted resources to retrieve the policy information and user's roles. 3. Validate the updated proposal sections data and requested actions based on access control decisions and uses roles. 4. Reject saving the updated proposal document that violates the user permission.
Post Condition	1. The application prevents accessing to unauthorized resources. 2. The application prevents unauthorized proposal data. 3. The application discovers and prevents processing the tampered client requests.
Priority	High.

B.4 GPMS Security Test Models Based on Use Case Technique

B.4.1 Create a New Proposal Document by PI Security Test Model

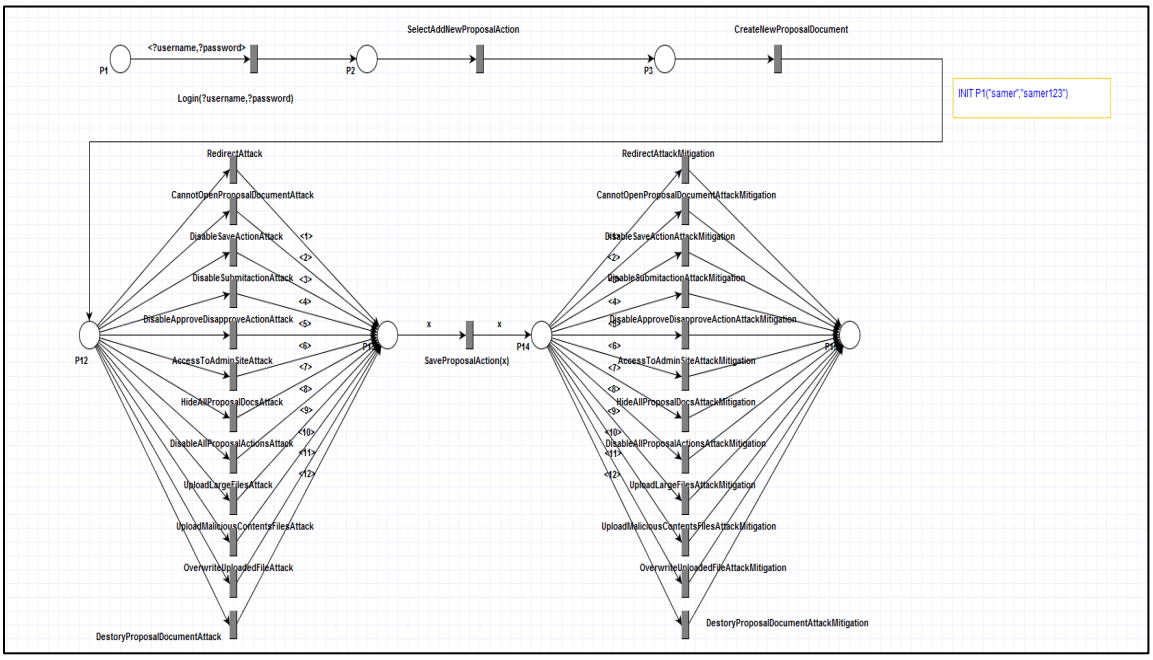

Figure 44: Create a New Proposal Document by PI Security Test Model.

B.4.2 Submit a Proposal by PI Security Test Model

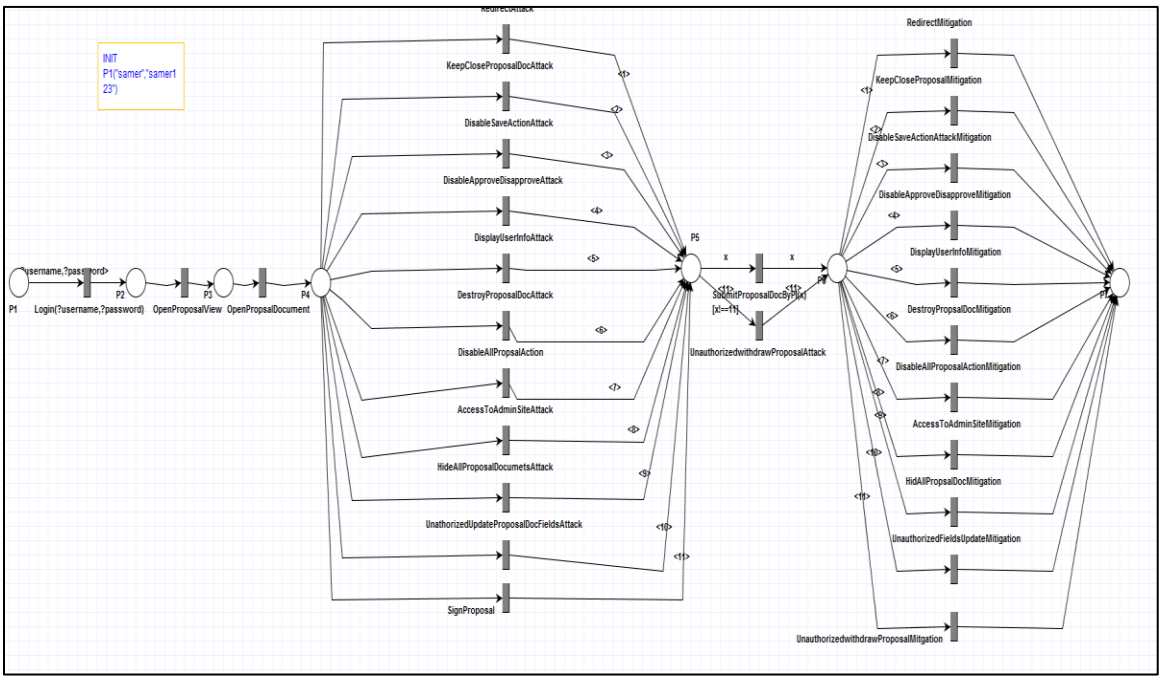

Figure 45: Submit a Proposal by PI Security Test Model.

B.4.3 Sign a Proposal by CO-PI Security Test Model

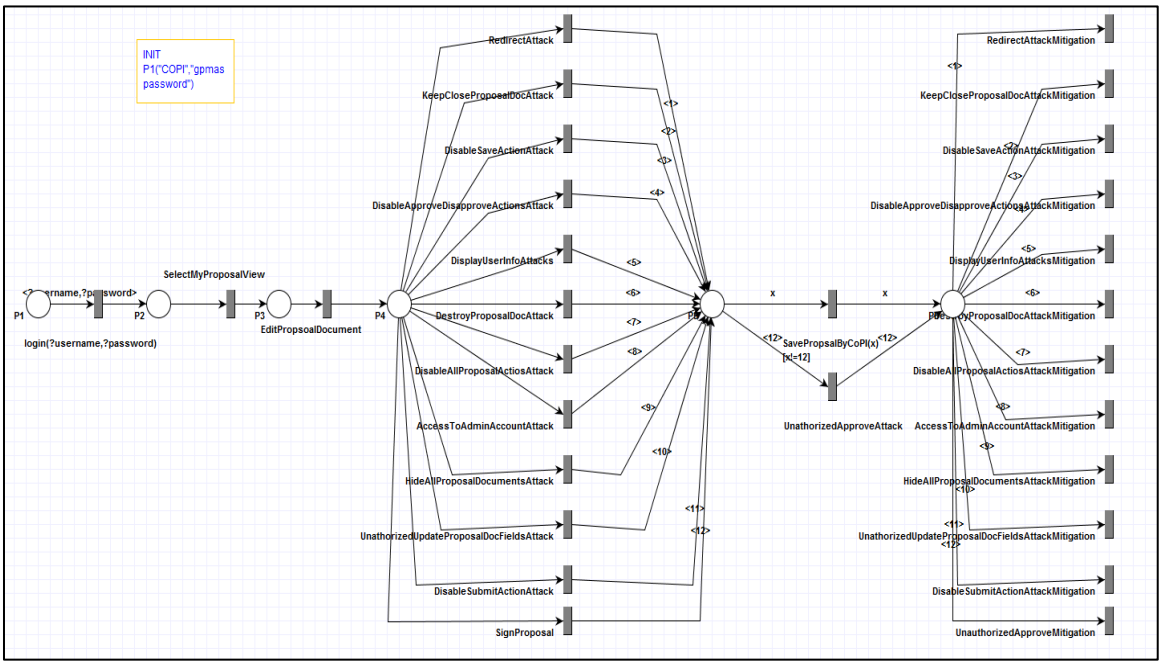

Figure 46: Sign a Proposal by CO-PI Security Test Model.

B.4.4 Approve/Disapprove a Proposal by Department Chair Security Test Model

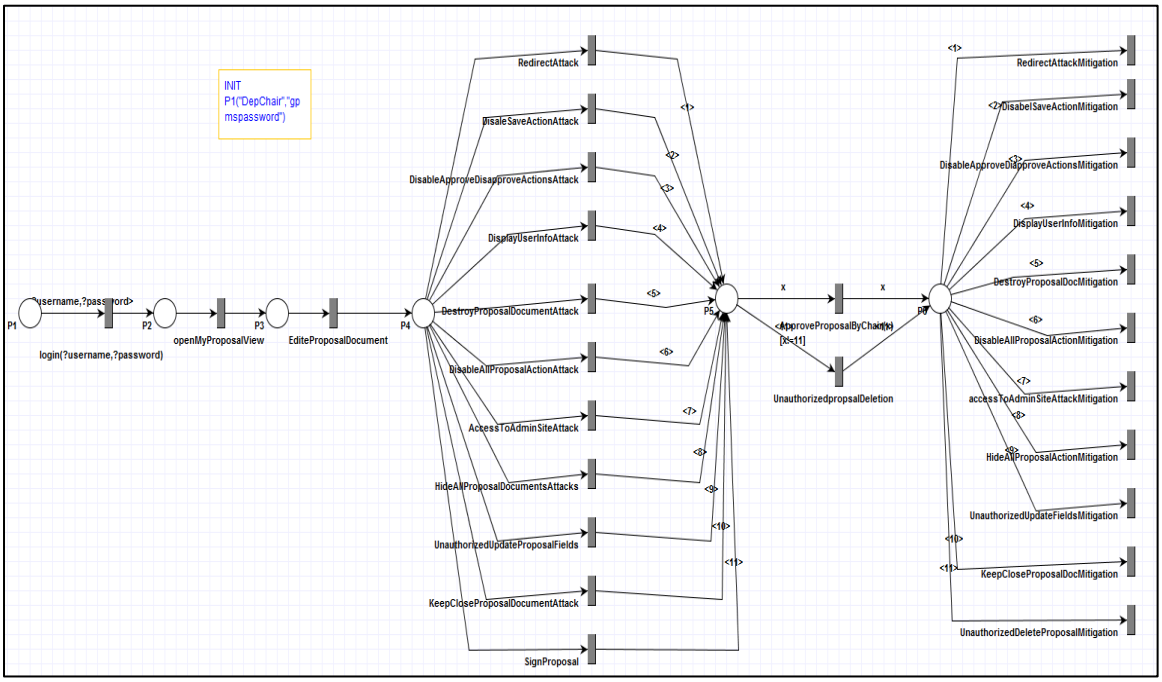

Figure 47: Approve/Disapprove a Proposal by Department Chair Security Test Model.

B.4.5 Approve/Disapprove a Proposal by Research Admin Security Test Model

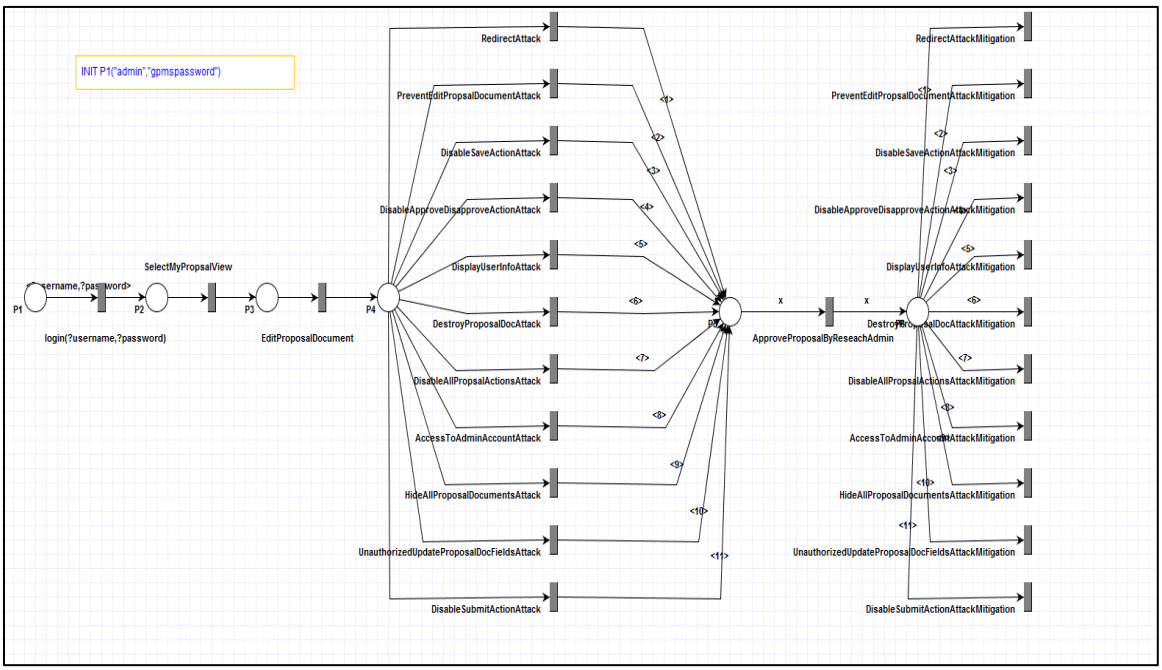

Figure 48: Approve by Approve/Disapprove Admin Security Test Model.

B.4.6 Approve/Disapprove a Proposal by IRB Security Test Model

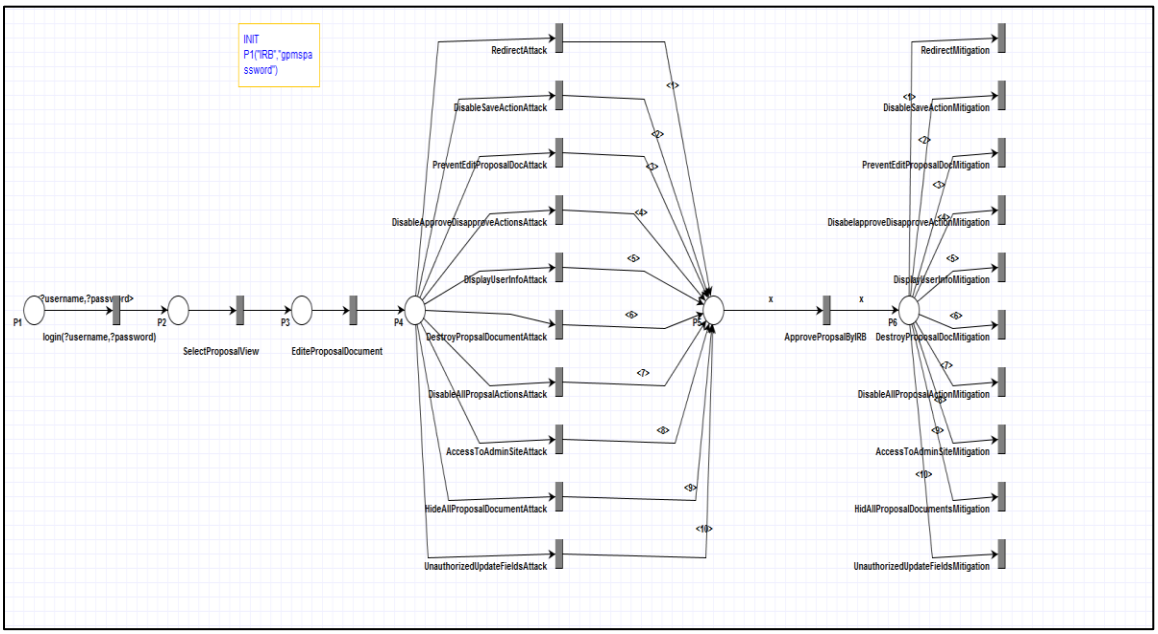

Figure 49: Approve/Disapprove a Proposal by IRB Security Test Model.

B.4.7 Approve/Disapprove a Proposal by Research Director Security Test Model

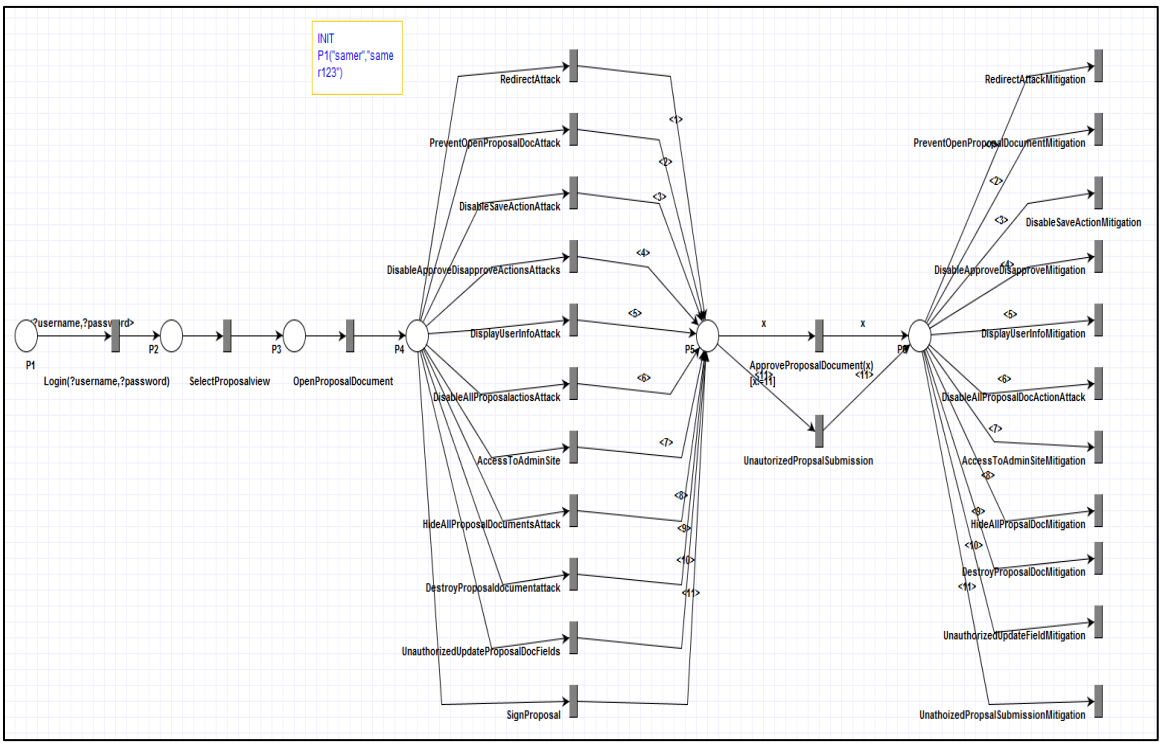

Figure 50: Approve/Disapprove by Research Director Security Test Model.

B.4.8 Approve/Disapprove a Proposal by Dean Security Test Model

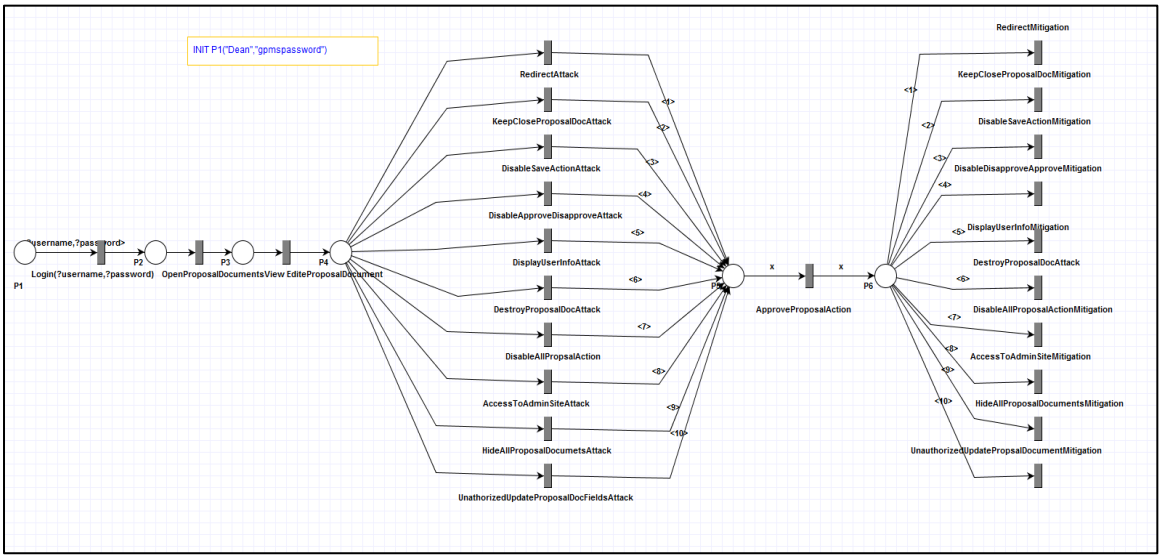

Figure 51: Approve/Disapprove by Dean Security Test Model.

B.4.9 Approve/Disapprove a Proposal by Business Manager Security Test Model

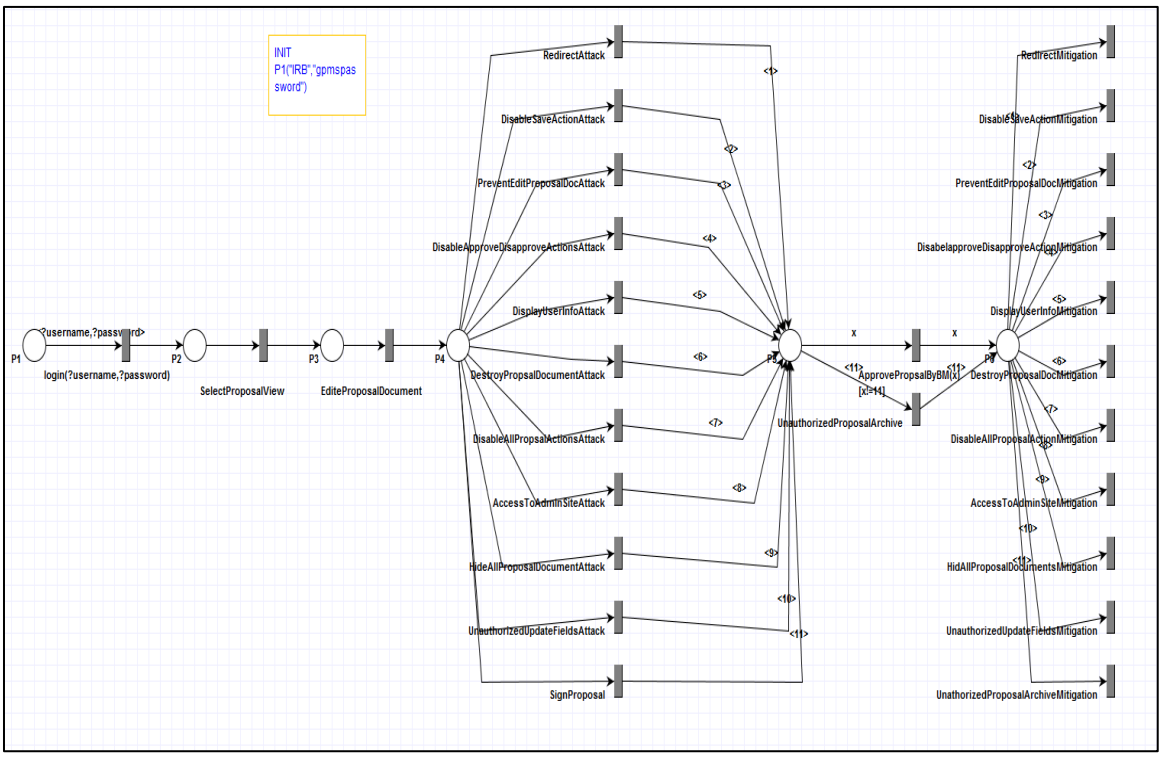

Figure 52: Approve/Disapprove by Business Manager Security Test Model.

B.4.10 Signup New User Attack Security Test Model

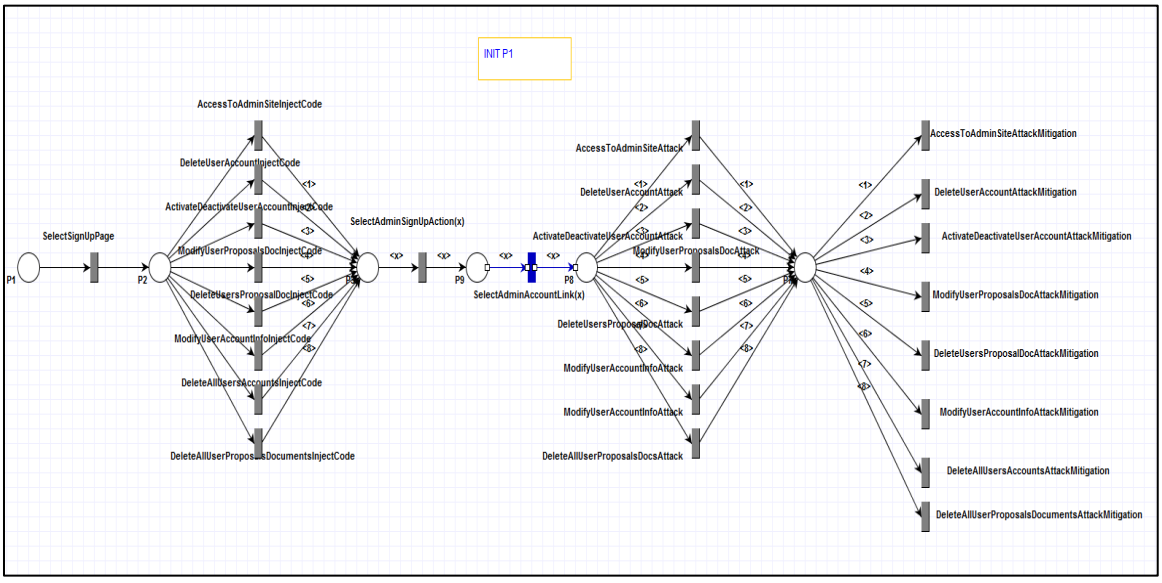

Figure 53: Signup New User Attack Security Test Model.

B.4.11 Submit a Proposal by Research Admin Security Test Model

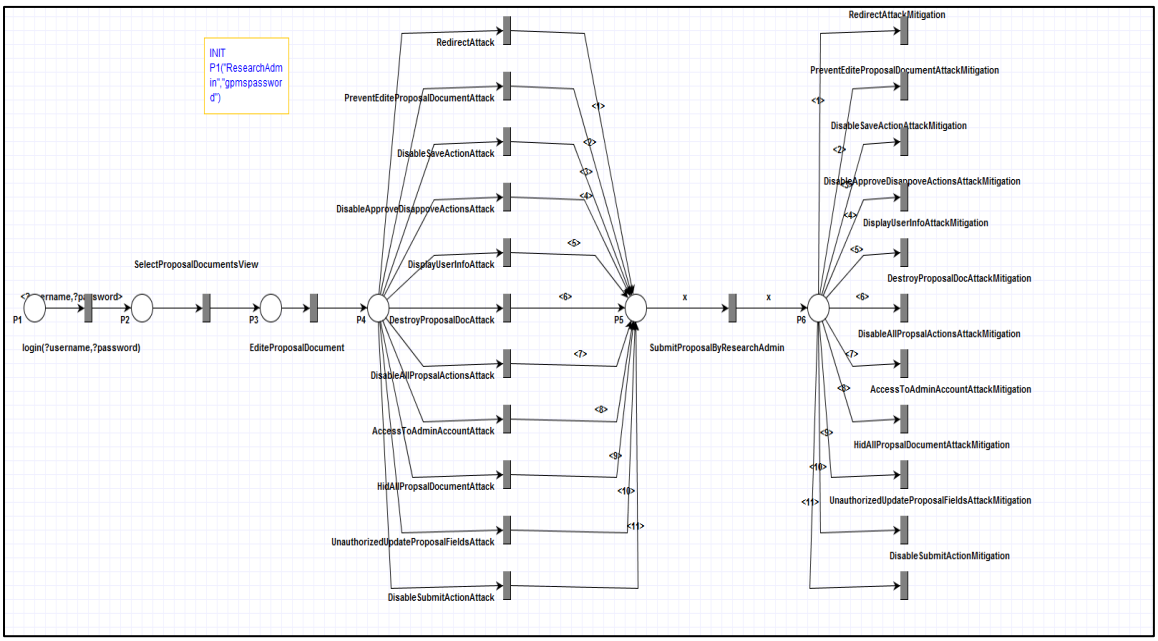

Figure 54: Submit a Proposal by Research Admin Security Test Model.

B.4.12 Withdraw a Proposal by Research Admin Security Test Model

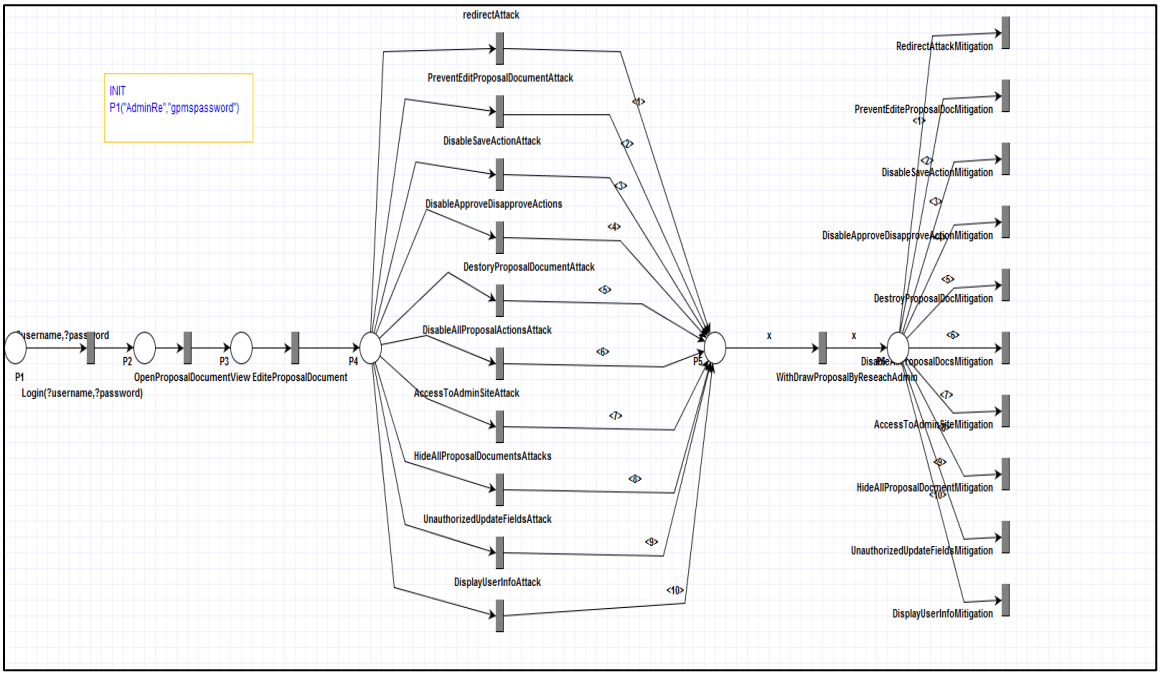

Figure 55: Withdraw a Proposal by Research Admin Security Test Model.

B.4.13 Login Use Case Harvest Attack Security Test Model

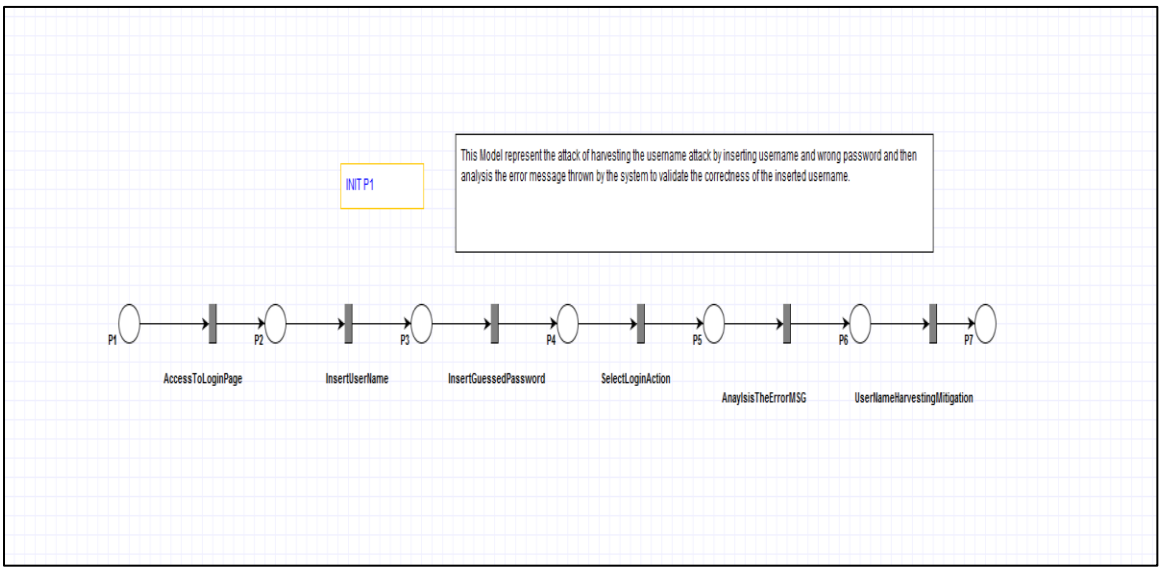

Figure 56: Login Use Case Harvest Attack Security Test Model.

B.5 GPMS Security Test Models Based on STRIDE Technique

B.5.1 Denial of Service, Information Disclosure, and Privilege Elevation Security Test Model

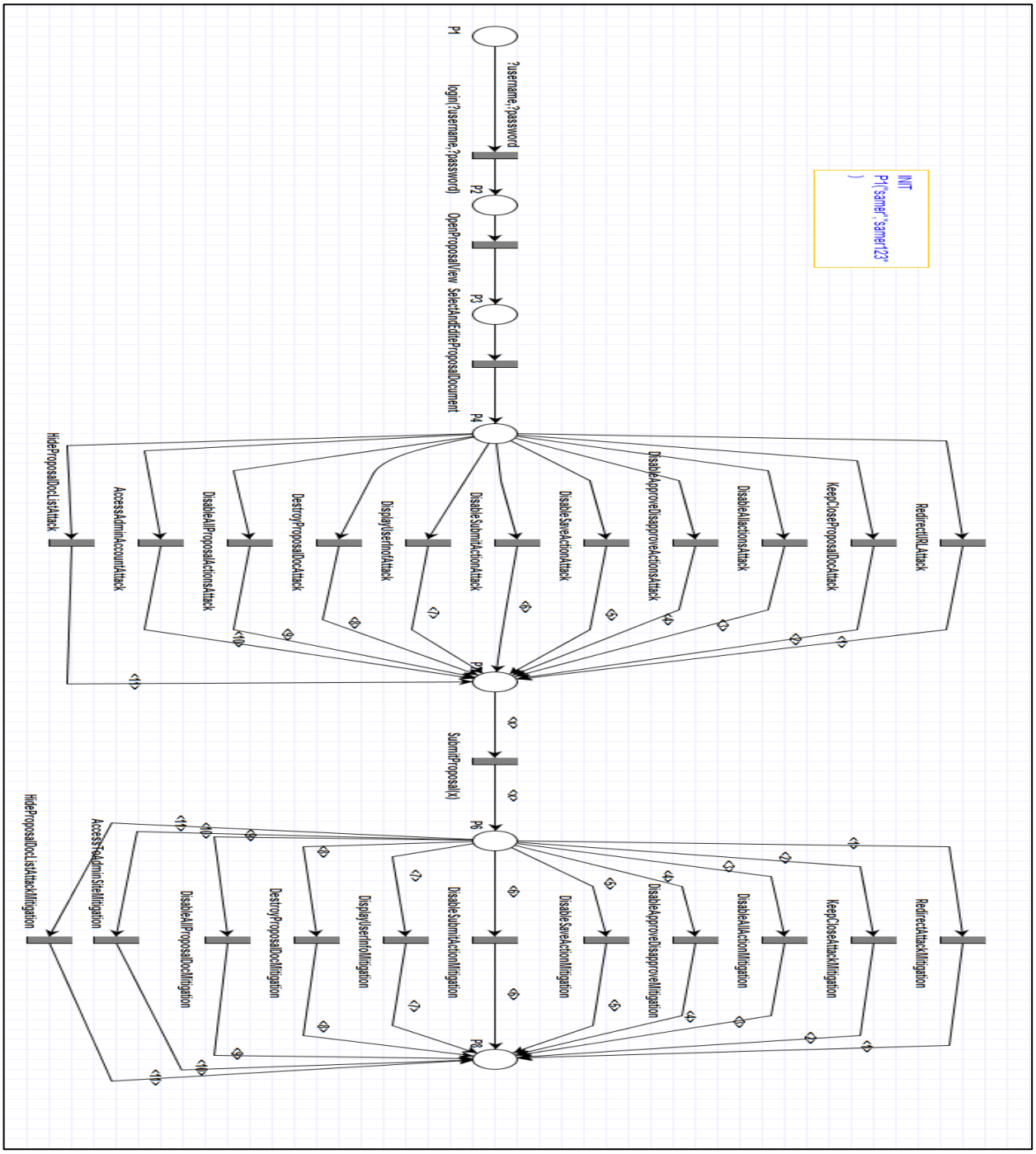

Figure 57: Information Disclosure, Denial of Service, and Privilege Elevation Categories Security Test Model.

B.5.2 Spoofing, Privilege Elevation and Denial of Service Security Test Model

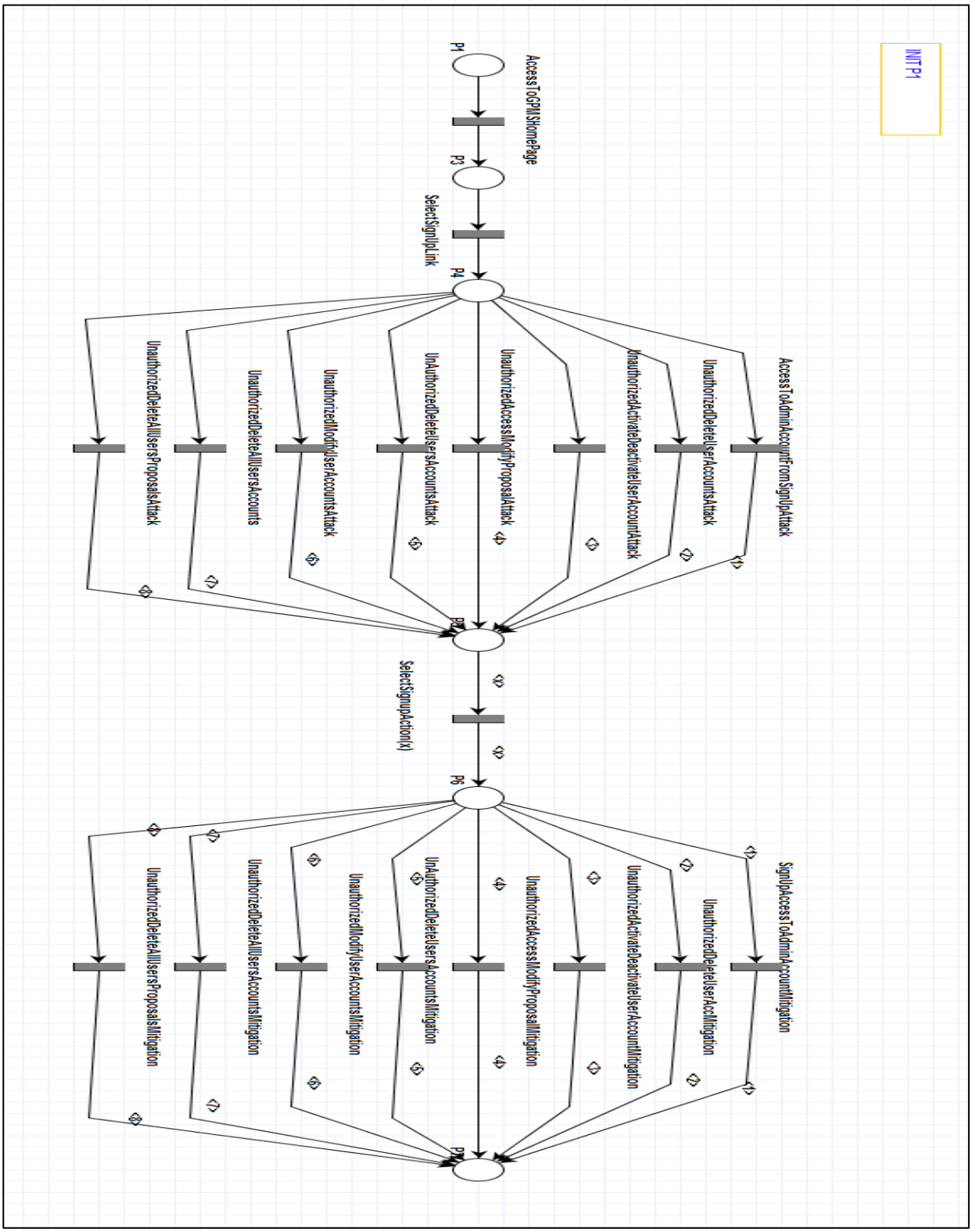

Figure 58: Privilege Elevation and Denial of Service Security Test Model

B.5.3 Denial of Service Category Security Test Model

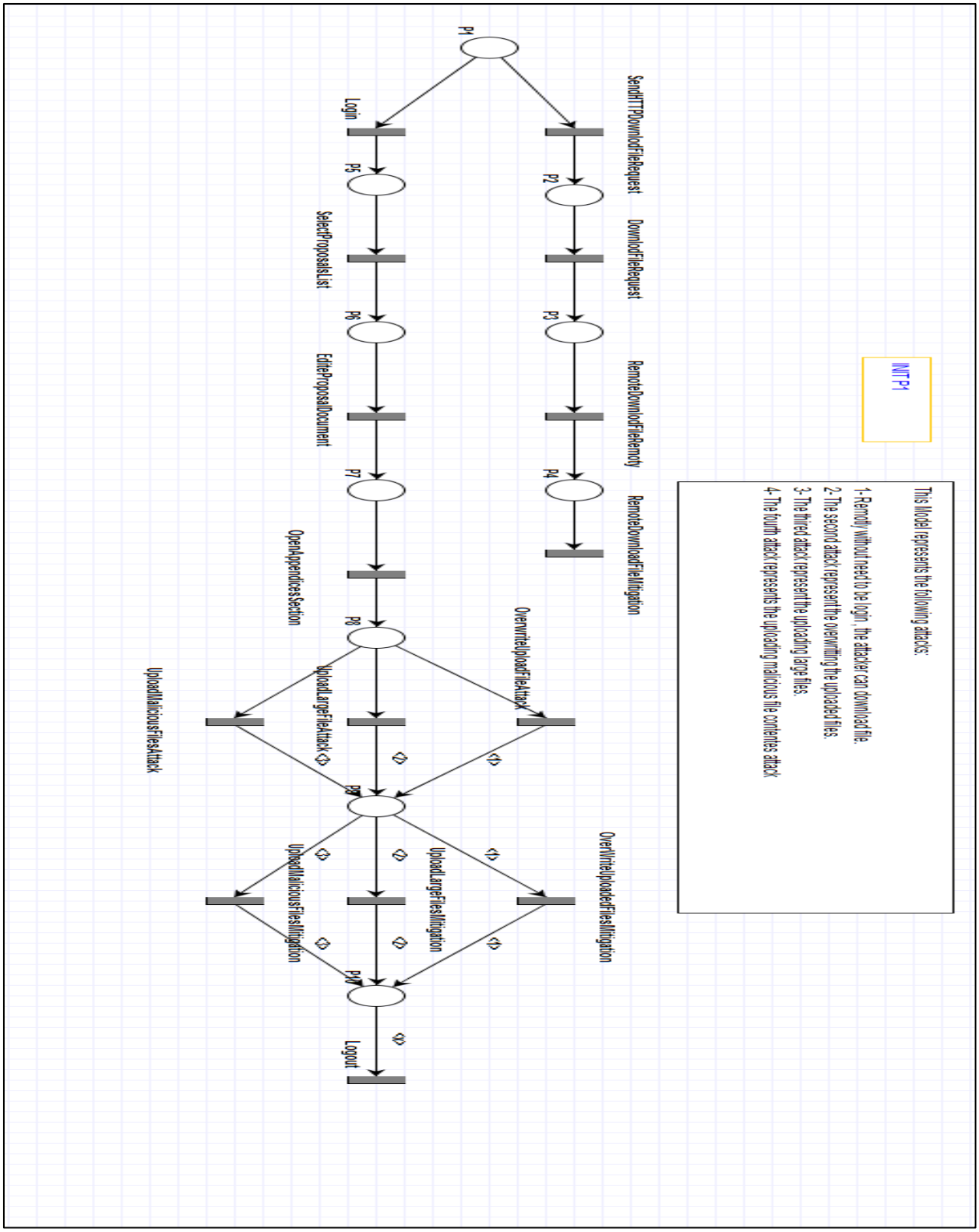

Figure 59: Denial of Service Category Security Test Model.

B.5.4 Denial of Service and Tampering Categories Security Test Model

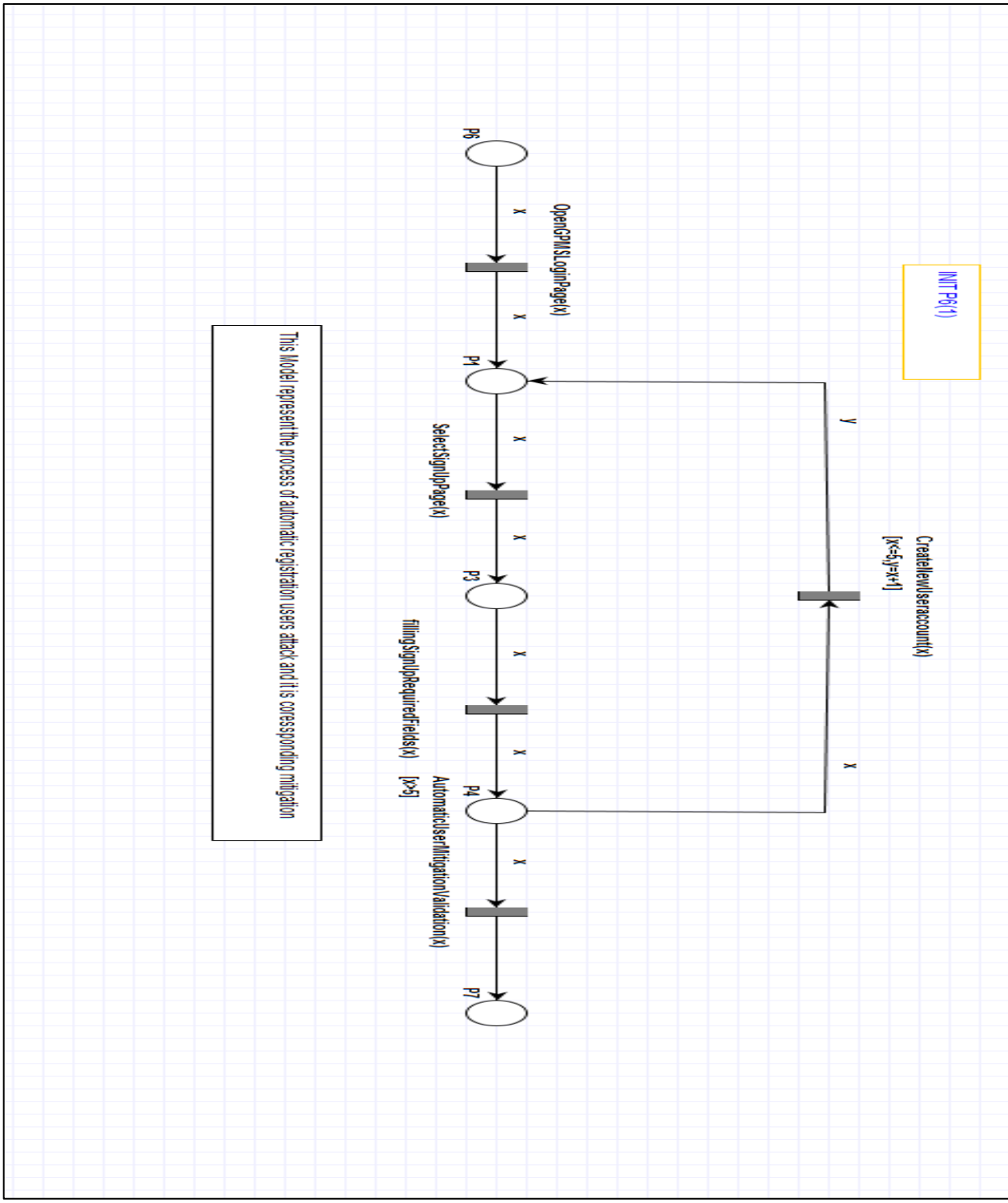

Figure 60: Denial of Service/Tampering Categories Security Test Model.