Modeling Security Requirements of Target of Evaluation and Vulnerabilities in UML

Kamran Yaqub
Acknowledgement

This work is carried out at the Division of System Sciences, Department of Business Administration and Social Sciences. Luleå University of Technology, Sweden.

I wish to express my gratitude to my supervisor Chen Yinru, her constant support and valuable comments in the course of my thesis.

My heartfelt acknowledgement goes to my parents, brothers, sister and loving cousin Nadia Latif and her parents for their encouragement.

I am also grateful to my supervisor Jörgen Nilsson, for his value comments and discussions. I am very thankful to AnnaCarin Larsson international office coordinator, David Carr program coordinator for Master of Science in Computer Science and Engineering, James Leblanc, Head of Division Signal Processing for supporting me in my research work.

Thanks to all the colleagues at the Division of System Sciences for their help to improve the contents of this thesis.
Modeling Security Requirements of Target of Evaluation and Vulnerabilities in UML

Kamran Yaqub
Division of System Sciences
Department Business Administration and Social Sciences
Luleå University of Technology

Keywords
Common Criteria, Protection Profile, Application Level Firewall, Security Requirement Engineering

Abstract
The Common Criteria (CC) provides Protection Profile (PP) for any organization or user to express their security requirements without considering implementation. PP is a template for specifying security features for different products.

However, the problems arise when user or organization develops the security requirement for Target of Evaluation (TOE) because Common Criteria (CC) expresses the security requirements in text. It is difficult for the PP developer to provide security measures without understanding the behavior of threats and threat agents. Therefore, there is a need to develop tools or methods for describing security requirements of the TOE graphically.

The objective of the thesis is to provide graphic description of the TOE security requirements. The corresponding research questions are to model Security Requirements of TOE focusing on assumptions and threats and vulnerabilities that are foundations of attacks.

In order to fulfill the object, the Unified Modeling Language (UML) is chosen as the research tool to capture the behavior of different threats in the operational environment. Application Firewall is used as a case study to show the connection among the assumptions of the TOE and how threat agents explore different vulnerabilities and access different assets.

It is expected that the research results will help any user to develop PP.
Table of Contents

Acknowledgement                                      i
Abstract                                             ii
Chapters                                             iii
List of Acronyms                                     vi
List of Figures                                      vii
List of Tables                                       viii

Chapters

1. Introduction

1.1 Background                                       1
1.2 Research objective                               2
1.3 Research Questions and Approaches                2
1.4 Research Method                                  3
1.5 Research Limitation                              3
1.6 Organization of the thesis                       3

2. Introduction to Common Criteria

2.1 Common Criteria                                  5
2.2 Key concepts in the Common Criteria               5
2.3 Users of the Common Criteria                      6
    2.3.1 Consumers                                    6
    2.3.2 Developers and product vendors               6
    2.3.3 Evaluators                                  6
2.4 Protection Profile                               6
2.5 Relationship between Security Target and Protection Profile 7
2.6 Security Target and Protection Profile formal evaluation and Security environment 7
2.7 TOE Security Environment                         8
    2.7.1 Assets                                       8
    2.7.2 Entry Points                                 8
    2.7.3 Assumptions                                  8
    2.7.4 Physical Aspect                              8
    2.7.5 Personnel Aspect                             8
2.8 Threats                                          8
2.9 Identification and characterization of Vulnerabilities 9
2.10 Difference between threats and vulnerabilities   9
2.11 TOE Relation with Threat, vulnerabilities        10
3. Introduction to Unified Modeling Language (UML)

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Model</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>Visual presentation in UML</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>UML Diagrams</td>
<td>12</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Use/Misuse Case diagram</td>
<td>12</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Actor</td>
<td>13</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Relationship between Use Cases</td>
<td>13</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Extends Relationship</td>
<td>14</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Uses Relationship</td>
<td>15</td>
</tr>
<tr>
<td>3.3.6</td>
<td>Use Case in Security Context</td>
<td>15</td>
</tr>
<tr>
<td>3.3.7</td>
<td>Use/Misuse Case Security Requirements</td>
<td>15</td>
</tr>
<tr>
<td>3.4</td>
<td>Sequence diagram</td>
<td>16</td>
</tr>
<tr>
<td>3.5</td>
<td>Class diagram</td>
<td>16</td>
</tr>
<tr>
<td>3.6</td>
<td>Object diagram</td>
<td>16</td>
</tr>
<tr>
<td>3.7</td>
<td>State diagram</td>
<td>16</td>
</tr>
<tr>
<td>3.8</td>
<td>Activity diagram</td>
<td>17</td>
</tr>
<tr>
<td>3.9</td>
<td>Collaboration diagram</td>
<td>17</td>
</tr>
<tr>
<td>3.10</td>
<td>Component diagram</td>
<td>17</td>
</tr>
<tr>
<td>3.11</td>
<td>Deployment diagram</td>
<td>18</td>
</tr>
<tr>
<td>3.12</td>
<td>Role of modeling in security engineering</td>
<td>18</td>
</tr>
<tr>
<td>3.13</td>
<td>Modeling TOE Security Environment and Protection Profile</td>
<td>18</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>What Are Firewalls</td>
<td>20</td>
</tr>
<tr>
<td>4.2</td>
<td>The Firewall role of protection</td>
<td>20</td>
</tr>
<tr>
<td>4.3</td>
<td>Application Level Firewall</td>
<td>21</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Proxy Servers</td>
<td>21</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Application Level Proxy</td>
<td>21</td>
</tr>
<tr>
<td>4.4</td>
<td>Application level Firewall in Protocol Context</td>
<td>22</td>
</tr>
<tr>
<td>4.5</td>
<td>User Connection with FTP or Telnet</td>
<td>23</td>
</tr>
<tr>
<td>4.6</td>
<td>Application Level Firewall in UML</td>
<td>23</td>
</tr>
<tr>
<td>4.7</td>
<td>Identification of Security Threats</td>
<td>27</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Sniffer programs</td>
<td>27</td>
</tr>
<tr>
<td>4.7.1.1</td>
<td>How Does a Sniffer Work?</td>
<td>27</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Cracker programs</td>
<td>29</td>
</tr>
<tr>
<td>4.7.2.1</td>
<td>Brute force attack</td>
<td>29</td>
</tr>
<tr>
<td>4.7.2.2</td>
<td>Dictionary Attack</td>
<td>29</td>
</tr>
<tr>
<td>4.7.2.3</td>
<td>Ruled Based Attack</td>
<td>29</td>
</tr>
<tr>
<td>4.7.3</td>
<td>Port Scanners</td>
<td>31</td>
</tr>
</tbody>
</table>
List of Acronyms

EAL Evaluation Assurance Level
IT Information Technology
PP Protection Profile
SF Security Function
SFP Security Function Policy
SOF Strength of Function
ST Security Target
TOE Target of Evaluation
TSC TSF Scope of Control
TSF TOE Security Functions
TSFI TSF Interface
TSP TOE Security Policy
UML Unified Modeling Language
List of Figures

Fig 2.1 Developer concepts and relationships.................................................................11
Fig 3.1 Threat model........................................................................................................16
Fig 4.1 shows how proxy works.......................................................................................21
Fig 4.2 Show the relationship between network layers..................................................22
Fig 4.3 Relation between Attacker, Application Firewall and proxy...............................24
Fig 4.4 Shows the activity of Application level Firewall...............................................25
Fig 4.5 Sequence diagram of Application level firewall...............................................26
Fig 4.6 How sniffer programs work in the environment.................................................28
Fig 4.7 shows relations between victim and attacker using Abuse/Use Case...............30
Fig 4.8 Sequence Diagram Shows Crack program.......................................................31
Fig 4.9 (a) TCP connect() scan....................................................................................33
Fig 4.9 (b) TCP connect() scan....................................................................................34
Fig 4.10 Sequence diagram of SYN|ACK Scanning.....................................................35
Fig 4.11 Sequence diagram of Fin Scanning.................................................................36
Fig 4.12 Use Case of UDP Scanning threaten by the attacker........................................37
Fig 4.13 Use case of Assumptions of Application level firewall....................................40
Fig 5.1 Sequence diagram of Software Vulnerabilities, and Trojan............................44
Fig 5.2 Sequence diagram of Bounce Scan attacker....................................................46
List of Tables

Table 1 kinds of use case relationship..........................................................14
This Thesis is dedicated

To my

Parents, Uncle Aslam Rehmat and his family
Chapter 1

Introduction

This chapter outlines the background, objectives, research questions and approaches, limitation and structure of the thesis. The basic concept and other issues surrounding the Common Criteria and Firewall are also overviewed.

1.1 Background

There are different kinds of threats which cause damage to assets. Security mechanisms protect assets against potential threats. There are different security mechanisms, which consist of different services i.e. Firewalls, Intrusion Detection System are used to protect assets from threats. These security services are employed to fulfill security requirement such as authentication. All security mechanisms provide very isolated functionality. They might not be sufficient to protect assets against potential threat from different emerging environments like ad hoc environment and mobile network [54].

The information security is becoming increasingly growing need for all organizations. Different organizations are using different methods to store information. On other hand threat agents are using different techniques to access information. To protect those assets different strategies are used, Which protect assets against threats [17,18]. The gap among the security requirements and their formal specification is filled by Common Criteria. Common Criteria is used to create two type of documents “Protection Profile” and “Security Target”. To create these documents one should go through analysis of security environment. Security environment consist of threat, assumptions and policies. During development of protection profiles there are assumptions and threats, which have been set up. Protection Profile set up list of security requirements, these security requirements are selected to achieve objectives. There are two types of security requirements functional and non functional requirements. Functional requirements captures the system of the system where as non functional requirements define constraints. Gathering functional and non functional requirements is complex task during the development of Protection Profiles. Common Criteria explains functional requirements in textual form. The textual form is very difficult for the developer to understand the behavior of the system. There is a need to express security requirements graphically. The graphical presentation show the behavior of threats and vulnerabilities in operating environment [19,37].

Threats are presumed attack method or any vulnerability that are the foundation of attack [28].This research work focus on gathering functional and non functional security requirement using Unified Modeling Language. The graphical presentation shows how threat agents explore different vulnerabilities and gain access to different assets. An Application Level Firewall is selected as case study for this research work. Application level Firewalls are considered to be the most secure type of Firewall’s but they incur important performance penalty [27]. Different threats are explored, which can bypass firewall and harm assets. Threats agents use different techniques and tools to
bypass firewalls and get access to assets. It is important for the developer to know the behavior of the threats because it would give guideline to design security mechanism. Moreover different FTP vulnerabilities are exploited by the threat agents. Vulnerabilities are important issues to be address because negligence of vulnerabilities would weaken the TOE. This research work also shows how the attacker can exploit different vulnerabilities.

UML defines various ways to develop models, and while all maybe useful the most commonly used models are Use Case and Sequence diagrams and Component models. It is useful to decompose the application into its core components in order to understand how it works. Unified Modeling Language (or UML) can be an exceptionally useful tool to help visually model applications [33].

1.2 Research objective

The aim of this research is to explore the TOE security environment threats and assumptions and capturing functional and non-functional requirements using Unified Modeling Language. Modeling of TOE security environment captures the functional and non-functional requirements using Use/Abuse case diagrams. Security engineering of TOE security environment refines the security objective into a set of security requirement. Identification of vulnerabilities strengthens the TOE and make effective.

1.3 Research Questions and Approaches

The research questions and the proposed approaches can be formulated as

1. How to model Security Requirements, mainly assumptions and threats without considering policies for Security Target?

   (a) Analysis of Application Level Firewall, different assumptions are made to describe the working of Firewall using Use Case methodology.

   (b) Perform threat modeling using Use Case/Abuse for security requirement.

   (c) Develop different threats functions TOE face from threat agents.

2. How Sequence diagram help to identify vulnerabilities that are foundation of attacks?

   (a) Identify vulnerabilities for firewall.

   (b) The Sequence diagram of different vulnerabilities represents the Sequence which represents how objects interact with each other and focus on message sequences.
1.4 Research Method

To address the problems Application Level Firewall is selected as case study. In first step the behavior of the application level firewall is examined using use case and sequence diagrams. It is very important to address the behavior of the application level firewall to capture the security requirement. The activity diagram is needed to examine the activities to operation environment of application level firewall. Use cases are used to define the mitigation approach, which measures neutralization of security threats. Use case modeling is a proven method for the elicitation of communication and functional requirement. Use case diagrams and sequence diagrams help to understand the scope of protection which PP developer must understand. Scope of protection defines the behavior of threat agents in operational environment. The integral development of use case and misuse case provides a systematic way for the elicitation of various system requirement, functional and non functional [50,55].

The Use/Abuse defines threat mitigation and shows how threat agents can violate the firewall and harm assets. Identification of different threats and vulnerabilities is performed using sequence diagrams. The sequence diagrams are exploited to describe the decision making process an attacker would go through to comprise or misuse the system.

1.5 Research Limitations

The aim of this research is to perform visual presentation of TOE security as there are different threats in environment that can cause damage to the assets. TOE security environment describe different assumption. Security environment consists of threats, assumptions and policies. To develop effective TOE it is necessary to analyze the TOE security environment. Threat agent use different methods to break the security mechanism and cause damage to assets. Analyzing security environment of a distributed system is a very difficult task, in a distributed environment there are different entry point for threat agents which can cause damage through active or passive attacks. New methods used by the threat agents in distributed environment or new methods to exploit vulnerabilities are out of scope of this thesis.

1.6 Organization of the Thesis

Chapter 1 Introduces the concept and other issues surrounding the Common Criteria and Firewall in addition to the outline of the thesis.

Chapter 2 Introduction of Common Criteria, Protection Profile and Unified modeling language and discuss different security modeling issues regarding Common Criteria and UML. This chapter describes the background of Common Criteria role of security environment and users of common criteria. TOE security environment describes the different security aspect of the environment in which the TOE is intended to be used. Moreover it describes the role of modeling for TOE Security Environment.
Chapter 3 This chapter explains the concept of application level firewall and present working using UML diagrams. It also explains different threats in TOE security environment Sequence diagrams and Use/Abuse Case methodology that are used to show how it works in the environment. This explains how UML diagrams can be used in security modeling. It also shows how to perform visual presentation of TOE assumptions and threats using Use Case methodology.

Chapter 4 This chapter explains how firewall works. It explores behavior of the firewall in the environment using UML diagrams. It also explores different threats in the environment which cause damage to the assets.

Chapter 5 This chapter explains some FTP vulnerabilities which can be exploited by the threat agents. It very important to address the vulnerabilities because ignorance of vulnerabilities effect the TOE.

Chapter 6 This chapter discusses security view of UML diagrams.

Chapter 7 This chapter contains some conclusion and suggestion for further research.
Chapter 2

Introduction to Common Criteria

This chapter describes concepts of Common Criteria, Protection Profile and discusses different security modeling issues regarding Common Criteria. This chapter also discusses TOE security environment, which consist of assets, threats and assumptions. Moreover it addresses vulnerabilities issues. It describes the relation between threats and vulnerabilities and relates how it is important for the TOE to address vulnerabilities.

2.1 Common Criteria

The Common Criteria is established by an international team to replace the security criteria. The team develops the set of requirements for the security functionality in order to protect organization assets. The CC is useful for the development of products with IT security functionality. The CC defines the set of security requirements, which protects assets from unauthorized disclosure, or loss of data. CC is applicable to protection of assets from different threats being raised from different threat agents. The CC specifies meta-criteria (i.e criteria for creating security criteria). CC defines the procedures for evaluating products against security criteria. CC uncouples security features from security assurance, which are strongly related to TCSEC [56,37].

2.2 Key concepts in the Common Criteria

Important parts of Common Criteria are as follow:

- **Target of Evaluation (TOE)**

  This section describes security requirement and define the product type. It defines general IT features [57].

- **Target of Evaluation (TOE) Security Environment**

  This section describes the security aspect of the environment in which the TOE is intended to be used and define assumptions, threats and security policies [57].

- **Security Objectives**

  Security objective define how to protect different asset in the environment. It emphasizes on technologies which explain how to achieve security objectives [57].

- **TOE Security Requirements**
This section defines the functional and assurance security requirements that the TOE need to satisfy in order to meet the security objectives for TOE [57].

2.3 Users of the Common Criteria

There are three main users of the common criteria

- Consumers
- Developers and product vendors
- Evaluators

2.3.1 Consumers

Common Criteria plays an important role in developing IT security requirements for consumers to satisfy their organization needs. It satisfies the need of consumer because it provides the fundamental purpose and justification of evaluation process. Consumers can use the results of evaluation to decide whether an evaluated product or system fulfill the needs. The evaluation results can be use to compare different products or systems. The Common Criteria define protection profile in consumers express their security measurement in the TOE [37].

2.3.2 Developers and product vendors

The Common Criteria assist developers in the evaluation of their products or systems. The Common Criteria define the security function that a developer can include in the TOE. The common criteria define the actions and responsibilities which is necessary to support the evaluation of the TOE [37].

2.3.3 Evaluators

The CC is used by the evaluators to perform the conformance of TOE to their security requirement. The CC describes the set of general actions the evaluator is to carry and the security function on which to perform these actions [37].

2.4 Protection Profile

The Common Criteria define the Protection Profile (PP), which allows consumers or developers to create standardized set of security requirements which meet their requirements [29].

“Implementation independent statement of security requirements for a category of TOE (target of evaluation) that meet specific customer needs to address a specified security environment.” [29]

A protection profile describes a set of requirements that are specified with the aim of countering specified threats in a specified environment. The protection profile may not
describe the correct solution countering threats, but its relevant information effectively addresses that problem space. It is expected that a Protection Profile may be written by a user community, government, industry group or insurance firm [29].

A PP would be appropriate in the following cases:

- A consumer group wishes to specify security requirements for an application type [29].
- A government wishes to specify security requirements for a class of security products [29].
- An organization wishes to purchase an IT system to address its security requirements [29].

2.5 Relationship between Security Target and Protection Profile

The Common Criteria is used to create two kinds of documents, a “Protection Profile” or a “Security Target” (ST). A Security Target is a document that identifies what a product actually does, or a subset of it, that is security relevant. A Security Target doesn’t need to meet the requirements of any particular Protection Profile, but ST could meet the requirements of one or more PP. A “Protection Profile” is created by different groups of consumers or different organizations that identifies the desired security properties of a product [29,30].

2.6 Security Target and Protection profile Formal evaluation and Security environment

An evaluation is a process which ensures that PP simply meets various documentation and judgment checks. A Security Target not only involves examining the Security Target document, but it also evaluate the actual system called the “Target of evaluation” or TOE. The purpose of an ST evaluation is to ensure the level of the assurance requirements specified by the ST, the actual product (the TOE) meets the Security Target security functional requirements. Comparison help consumers to determine if the products meet their requirements and if not what are the limitations [31]. To create a PP and ST, you go through a process of identifying the security environment, namely, your assumption, threats and relevant organizational security polices. From the security environment, you drive the security objectives for the product or product type. The security requirements are selected so that they meet the objectives. There are two types of security requirements [31].

- Functional requirements
- Assurance requirements

Development of PP or ST is a complex task. In that situation CC is very useful in identifying security environment, objectives and requirements. The CC text is used to define standardized functional requirement and assurance requirements [31].
2.7 TOE Security Environment

The statement of TOE security environment describes the different security aspects of the environment in which the TOE is intended to be used. TOE security environment consist of assumptions made on the operational environment. It defines different threats that the product is designed to counter, and the organizational security policies with which the product is designed to comply. The TOE security environment description includes a statement of assumptions that are to be met by the TOE environment in order for the TOE to be considered secure.

2.7.1 Assets

Assets are resources, component or system that a threat agent might try to modify, steal or otherwise access or manipulate. Assets can be tangible such as a process token, or more abstract, such as data consistency [45].

2.7.2 Entry Points

Entry points shows all the places where the threat agent can attack the system including transfer points such as open sockets, remote procedure call (RPC) interface, web services, and data being read from the file system [45].

2.7.3 Assumptions

Assumptions may include aspects of the intended application, limitations of use, etc. There may be several kinds of assumptions.

2.7.4 Physical Aspect

It is assumed that TOE hardware and software including network and peripheral cabling is approved for the transmittal of the most sensitive data held by the system. These media must be prevented from unauthorized physical access [28,37].

2.7.5 Personnel Aspect

- One or more competent individuals manage the TOE and the security of the information it contains [28].
- The administrators are not careless, willfully negligent, or hostile, and will follow and abide by the TOE documented instructions [28].

2.8 Threats

The CC characterizes a threat in terms of a threat agent, a presumed attack method, any vulnerability that forms the foundation for the attack, and identification of the asset under attack. The threat consists of a threat agent, an asset (either in the operational or in the development environment) and an adversely action (adversely actions are the actions performed by a threat agent on an asset) of that threat agent on that asset [37,28, 41]. The CC addresses protection of assets in operating environment against threat agents. There
are different types of threat which are address by the CC. Following list of threats address unauthorized access that access or destroy assets.

- A threat agent may inappropriately read the data stored in different locations. In case of firewall the threat agent use sniffer programs to grab information [28].
- An unauthorized user attempt to gain entry using port scanning techniques [28].
- A hacker remotely copying confidential file from the company network [37].
- A worm, Trojan horse and spam that allowed privilege access to the server seriously degrading the performance of the wide area network [37].
- Violation of user privacy by a system administration [37].
- Stealing of designing source code [37].

2.9 Identification and characterization of Vulnerabilities

Vulnerability is “A flaw or weakness in a system’s design, implantation, or operation and management that could be exploited to violate the system’s security policy [47]”. According to this definition.

- Vulnerabilities are not just bugs, glitches or operator mistakes, but all elements (deliberate or accidental) that could cause the harm to the security objectives or privacy [46].
- Vulnerabilities can occur to all system assets software, hardware, services and information [46].
- Vulnerabilities are related to the whole life cycle of a system, from the specification of requirement to its implementation and configuration, to its use and maintenance [46].
- Vulnerabilities are not “incorrect” in system they are because
  1. They are exposed the system to the materialization of an accidental failure.
  2. They can be dangerously manipulated by malicious threat agents [46].

2.10 Difference between Threat and Vulnerabilities

A threat is simply a goal an adversary might try to achieve to abuse an asset in the system. Vulnerability is specific way that a threat can be exploited through a unmitigated attack path [45].
2.11 TOE Relation with Threat, vulnerabilities

Threat agent gain access to assets by exploring vulnerabilities and wish to abuse. Threats agents use different methods to exploit vulnerability, these vulnerability arise from the development environment of the TOE. The problem in the development environment such as accidental errors made during development or the intentional addition of malicious code may lead to TOE with exploitable vulnerabilities. The development environment has threat agent’s i.e development staffs that can intentionally include malicious code that cause the vulnerability to appear in the TOE [37]. Different methodologies are used to improve the security strength of the systems and remove the weakness of the system. There are different methodologies that exist mainly to improve the security strength of the system, because vulnerabilities make security system weak, which can be easily exploited by the threat agents.

![Diagram of Developer concepts and relationships [37]](image)

In above diagram its shows the relation between threat, vulnerabilities, threat agents and TOE. It show the threat agent wish to abuse or damage the assets, similarly vulnerabilities are caused by threats and vulnerabilities in TOE can easily be explored by the threat agents.
Chapter 3

Introduction to Unified Modeling Language (UML)

This chapter describes modeling concept and define different UML diagrams. How modeling presents the visual presentation of environment and captures the behavior of the system. It discusses the role of modeling in security engineering and discuss how Use/Misuse case captures functional and Non-functional requirements.

3.1 Model

A model captures the important aspects of the things being modeled from a certain point of view and simplifies the rest. It defines the essential aspects of a system and ignores some of the aspect which is not necessary. A model is also a description of the generic structure and meaning of a system. A model is considered to express things which are convenient for working [4]. A model makes it easier to break up a complex application or a huge system into simple, discrete pieces that can be individually studied. We can focus more easily on the smaller parts of a system and then understand the "big picture."[3]

Models have two major aspects.

- **Semantic information**

  The semantic information captures the logical aspect of the application such as use case, classes and messages. Model describes the semantic of the information. Models are artifacts in a computer system and they are used within the larger context that gives them their full meaning. The models are used in the larger context so they can be used to present visual overall system process of the organization [4].

- **Visual presentation**

  The semantic information is presented in visual form that can be easily browsed, and edited by humans. Presentation elements carry the visual presentation of the model that is they show in a form directly apprehensible by human. The visual presentation guide human to understand the meaning of the model. The visual presentations consist of different assumptions, internal organization of the model, a set of default, and a relationship to the environment in which they are used [4].

3.2 Visual presentation in UML

The unified modeling language is modeling language to construct, specify, visualize, and document the artifact of a software system. It defines the decision and understanding about system that must be constructed. It is used to understand, design, browse, configure, maintain and control information about system [4]. UML includes semantic concept, notation, and guidelines. It has static, dynamic, environment, and organization
parts. It is intended to be supported by interactive visual modeling tools that have code generators. The UML captures information about the static structure and dynamic behavior of a system. The UML also contains organizational constructs for arranging model into packages that permit developers to partition large systems into workable pieces, to understand and control dependencies among the packages, and to manage the versioning of model units in a complex development environment [4].

3.3 UML Diagrams

UML defines different tags, and stereotypes, tagged values and constraints. UML has a finite and fixed set of stereotypes associated with it. A stereotype has variables. A tag is an instantiation of a variable. Each stereotype has tags, threat specifications and constraints associated with it [53].

Constraints specify security requirements and threats specifications model actions. Different threats scenarios can be specified threat agent strength. Threat agent strength can be used in different scenarios using stereotypes. Therefore tags and constraints that specify security requirements can directly make statement. In UML, security engineering has been encapsulated into stereotypes so that they can be used in system modeling [53].

The basic idea of UML is that no one diagram can confine the different elements of a system in its entirety. Hence, UML is made up of nine diagrams that can be used to model a system at different points of time in the software life cycle of a system. The nine UML diagrams are:

3.3.1 Use/Misuse Case diagram:

Jacobson defines a use case in his 1992 book as “a behaviorally related sequence of transactions in a dialogue with the system” [5]. A more recent definition for the Rational Unified Process describes use case as “a description of a set or sequence of actions, including variants, that a system performs that yields an observable result of value to a particular actor” [6].

Use case has following characteristics.

- Use Cases are very useful if system consists of many people in requirement gathering and analysis because it captures the specification of the system [3,4].
- It breaks the system, subsystem, or class using primary elements termed as “actors” and the processes are called “Use Case”. It captures the behavior of the system and shows which actors interact with each use case [3,4].
- As use cases breaks a system into subsystems, or classes. An internal use case represents behavior that a part of the system presents to the rest of the system [3,4].
There are different approaches to analyze the security requirement. Misuse case is recent approach used to address different threat. Misuse case are used to analyze and specify security threats, misuse case concentrate on interaction between the application and its misuse cases i.e. threat agent who violate the security. Misuse case captures the specification and shows how threat agents threaten the security.

3.3.2 Actor

An actor present the interaction with the system, it shows how some someone uses the system. Actors show the interaction with the system and define how actor sends or receive messages to and from the system or exchanges information with the system. An actor can be human being or another system [7].

Characteristics of Actor;

- An actor is a type not an instance [7].
- The actor represents role, not an individual user of the system [7].
- It communicates with system by sending and receiving messages [7].
- A primary actor uses the system primary functions such as the main functionality [7].
- A secondary actor is one that uses the secondary functions of the system those functions maintain the system such as databases, communications and other administration task [7].
- Actor can be active or passive [7].
- Active actor is one that initiates use cases [7].
- Passive actor never initiates a use case but only participates in one or more use case [7].

3.3.3 Relationship between Use Cases

There are three types of relationships between use cases extend, uses, and grouping. This extends and uses relationship different forms of inheritance. Grouping is a way of placing related use cases together in a package. The relationship definitions are as follows [7].
<table>
<thead>
<tr>
<th>Relationship</th>
<th>Function</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association</td>
<td>The communication path between an actor and a use case that it participates in</td>
<td></td>
</tr>
<tr>
<td>Extend</td>
<td>The insertion of additional behavior into a base use case that does not know it</td>
<td>&lt;&lt;Extend&gt;&gt;</td>
</tr>
<tr>
<td>Use case generalize</td>
<td>A relationship between a general use case and a more specific use case that inherits and adds features to it</td>
<td></td>
</tr>
<tr>
<td>Include</td>
<td>The insertion of additional behavior into a base use case that explicitly describes the insertion</td>
<td>&lt;&lt; Include&gt;&gt;</td>
</tr>
</tbody>
</table>

Table 3.1 kinds of use case relationship

3.3.4 Extends relationship

A generalize relationship where one use case extends another use case by adding actions to a general use case. The extending uses include behavior from the use case being extended, depending on condition on the extension [7].

When a use case extends another use case it means that first may include some of the behavior of the use case it extends. It doesn’t have to include the entire behavior it can choose which parts of the behavior from the generalized use case it wants to reuse. The use case being extended must be complete. Since use case normally are described in plain text, it can be hard to define which parts in an extended use case are reused from the generalized use case, which are redefined, and which are added to the generalized use case [7].
3.3.5 Uses relationship

A generalization relationship where one use case uses another use case, indicating that as part of the specialized use case, the behavior of the general use case will also be included. When a number of use cases have common behavior, this behavior can be modeled in a single use case that is used by the other use cases [7].

3.3.6 Use Case in Security Context

Use Cases differ from requirements in two main ways. First Use Cases are used to generate a shared understanding of the problem to be solved, the key relationships and actors in a system. Bittner and Spence [22] refer to this as building up a shared understanding as opposed to decomposing features. The result of this is the second difference, which is that a Use Case model places requirements in a certain context. Context is critical in security in that the context can show how the Use Cases are related to the assets which the security mechanisms must protect, and the overall flows, dependencies and assumptions that the system makes [23].

3.3.7 Use/Misuse Case Security Requirements

The aim of security requirement engineering is to identify the scope of protection and evaluate the trade off analysis. The goal of security requirement is to explore the threats in the early phase of a system development lifecycle. The primary purpose of the security is to cover those assets that threat agent try to break. Misuse Case method purposed by the Sindre and Ophahl define the scenario of ‘negative’ agents which attempts to overcome the system. Misuse case shows the behavior of system by presenting the threat mitigation. Misuse cases and also present functional and non functional requirements [50,55].

![Fig 3.1 Threat model](image-url)
Consider the threat model, which shows relation between Assets, threats and vulnerability. During analysis functional requirements of the system are captured using use cases.

Development of misuse/use cases are recursive in nature, moving from system to subsystem level or lower a necessary. Lower-level cases can highlight aspect not considered at high level therefore it forces analysis at high level. The approach offers rich possibilities for exploring, understanding, and validating the requirements in any direction. If you know the mitigation approaches, you can proceed with development by trading off user requirements and system constraints. If you don’t know the mitigation approaches, you can proceed with development and use/misuse case analysis. When mitigations demand new subsystems or components, the new plans in turn usually cause new types of threat. We can analyze these threats to evaluate the need for further countermeasures. Mitigation measures rarely neutralize security threats [50,55].

3.4 Sequence diagram

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

3.5 Class diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class [3]

3.6 Object diagram

The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time [3].

3.7 State diagram

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.
3.8 Activity diagram

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions [3]. Activity diagram capture actions and their results. They focus on work performed in the implementation of an operation and the activities in a use case instance or in an object. The activity diagram is a variant of a state diagram and has a slight different purpose, which is to capture actions and their results in terms of object state changes. The states in the activity diagram transition to the next stage directly when the action in the state is performed (without specifying any events as in the normal state diagram). Another difference between activity diagram and state diagram is that their actions may be placed in swimlanes. A swimlane groups activities, with respect to who is responsible for them or where they reside in as organization. An activity diagrams is an alternative way of describing interactions, with the possibility of expressing how actions are taken, what they do (change of object states), when they take place (action sequence), and where they take place (swimlanes). Swimlanes are used for several different purpose, for example to show explicitly where actions are performed (in which object), or to show in which part of an organization work (an action) is performed. An action is performed to produce a result. The implementation of an operation may be described as a set of related actions, which is later translated to code lines. As defined earlier an activity diagram shows the actions and their relationships and can have a start and end point. [7]

Activity diagram may be used for different purpose, including.

- To capture the work that will be performed when an operation is executing (the instance of the operation implementation). This is the most common usage of activity diagrams.
- To capture the internal work in an object [7].
- To show how a set of related actions may be performed, and how they will affect objects around them [7].
- To show a business works in terms of workers (actors), workflow, organization, and objects (physical and intellectual factors used in the business) [7]

3.9 Collaboration diagram

A collaboration diagram groups together with the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

3.10 Component diagram

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they
are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase [3].

3.11 Deployment diagram

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed [3].

Now that we have an idea of the different UML diagrams, let us see if we can somehow group together these diagrams to enable us to further understand how to use them [3].

3.12 Role of Modeling in Security Engineering

Models are precise specification of requirement and design. They enable analysis, both formal and informal and informal. Models are used in security and policy specifications however, security modeling is typically disjoint and expressed in different ways. If integration of system design models with security models is poorly design it effect the environment analysis of TOE. Models are security design which combine security requirements and design. Security models express security requirement and polices [53]. The decomposition is a fundamental concept of modeling to control complexity, problems presented by large complex systems as problems of

- Intellectual gap- i.e the level of abstraction presented by the elements (implementation specifies, functional characteristics) is different from the level of abstraction that is of interest (Security) [35].

There are recommendations to add constructs to the CC paradigm that are currently used in information system modeling. These include framework, collaboration, interface and aspect [35].

3.13 Modeling TOE Security Environment and Protection Profile

Modeling is a proven information engineering tool use to address high level security requirements. TOE security environment need to be address using modeling, which captures the security requirement of an organization. The TOE environment consists of threats, assumptions consists of behavior in the operating environment. The modeling captures the behavior of threat and shows how threat agents access the assets. Modeling captures the functionality of system; Misuse Case modeling captures the non functional requirements and shows threat mitigation. The modeling shows the behavior of the threat agents, who weaken the TOE by using different entry points [52,54]. Modeling facilitates TOE security Environment in following way.

- Misuse/use cause facilitate to capture functional and non functional requirements. Which make refinements for Protection Profile [35].
Modeling enables to define scope for ST that is a component of a set of requirements [35].
Chapter 4

Modeling Firewall Security Requirement Using UML

This chapter explains the how a firewall works. It explores behavior of the firewall in the environment using UML diagrams. It also explores different threats in the environment which cause damage to assets.

4.1 What Are Firewalls

A network firewall is a barrier that controls the flow of information between a trusted (internal) network and an untrusted (external) network. A firewall lets users in your internal network use resources from an external network such as the Internet without compromising your network's data and other resources [25].

Firewall separates a protected network from an unprotected one, the internet. It screens and filters all connection coming from the internet to the protected (corporate) network, and vice versa, through a single, concentrated security checks point. A firewall makes sure that you cannot reach the internet from the internal network, or vice versa, unless your pass through this checkpoint. But before setting up the firewall we need to define what type of firewall best suite need analysis of topology, of network, such as hubs, switches, routers, and cabling, are suitable for a specific firewall model. To better understand the firewall we need to understand the corporate network based on the layers of its International Standards Organization ISO model. There are repeaters and hubs acting at the first layer, switches and bridges at the second layer, switches and bridges at the second layer, and routers at the third layer, a firewall passes through all these layers as it acts at the sixth and seventh level, the layers responsible for the session establishment controls and applications. Thus with a firewall we can control the flow of information throughout the establishment of sessions or even by determining which operations will or will not be allowed [24].

4.2 The Firewall Role of Protection

A firewall greatly improves network security and reduces risks to servers on your network by filtering inherently insecure services. As a result, your network environment is exposed to fewer risks because only selected protocols are able to pass through the firewall. For example, a firewall could prohibit certain vulnerable services such as NFS from entering or leaving a protected network [24]. This provides the benefit of preventing the services from being exploited by outside attackers, but at the same time permits the use of these services with greatly reduced risk of exploitation. Services such as NIS or NFS that is particularly useful on a Local Area Network basis can thus be enjoyed and used to reduce the server management burden [24].

Firewall can also provide protection from routing-based attacks, such as source routing and attempts to redirect routing paths to compromised sites through ICMP redirects. It
could reject all source routed packets, and ICMP redirects and then informs administrators of the incidents [24].

4.3 Application Level Firewall

4.3.1 Proxy Servers

Application level and circuit level firewall are two different implementations of a proxy server. A proxy server “stands in” for both the client and a server during a connection. A proxy servers act as the “man in the middle”, so that there is no direct contact between a client on an internal network and a server on an untrusted network [26].

Fig 4.1 shows how proxy works [39]

Technically, the proxy is not the firewall. The proxy runs on the firewall. This is an important distinction. The firewall stops the traffic from flowing through while the proxy allows the controlled access. The proxy is only a software solution to allow communication between two networks in a controlled manner [26].

Fig 4.1 shows relation shows relation between proxy server, application protocol and proxy client. When a request is send by the client to proxy server in application layer server it analyzes the packet and send to external real server reply will be send back to real client using same services.

4.3.2 Application Level Proxy

Proxy provide a gateway between a trusted and un trusted network through which information can pass therefore it is referred to as “Application Level gateways”. An
application level proxy operates at the connection level that can establish connections. Proxy also authenticate the user and authorize the source and destination addresses and permits or denies the protocol [26].

4.4 Application level Firewall in Protocol Context

Fig 4.2 shows the relation between different layers. An Application level firewall works on Application layer which permits connection, where both end of the connections are maintained through the proxy. Application level firewall supports different types of internet applications such as Simple Mail Transport protocol (SMTP) for email, HTTP proxy for web services [38]. Proxies manage traffic through a firewall for a specific service such as HTTP or FTP [1].

Fig 4.2 Show the relationship between network layers

Proxies use protocols such as HTTP and FTP, information about communication passing through the firewall is maintained by the proxy. Proxy does not allow direct communications between external server and internal computers and shield the internal IP address. Proxy services force all network packets to be examined and filtered for suitability. All packets are accessed and processed by the proxy services these packets are
evaluated according to the rules. Proxy services permit access to those packets which fulfill the rules, while denying access to others [1].

An application level firewall evaluates network packets for valid data at the application layer before allowing a connection. The firewall examines the data in all network packets at the application layer and maintains complete connection state and sequencing information. Other security items such as user password and service request that appear in the application layer data can be validated by the firewall [1].

4.5 User Connection with FTP or Telnet

When the internal user wants to connect to an external service such as FTP or Telnet they send a request to the proxy server for the connection. The proxy server decides whether to permit or deny the request based on an evaluation of a set of rules that is managed for the individual network service. Proxy servers only allow those packets through that comply with the protocol definitions because the servers understand the protocol of the service they are evaluating. The proxy client is the component that talks to the server on the external network on behalf of the real client on the trusted network. The proxy server evaluates a real client's request for a service against the policy rules defined for that proxy and determines whether to approve the request. The proxy server forwards the request to the proxy client if the request is approved. The proxy client contacts the real server on the external network on behalf of the client. The proxy client relays requests from the proxy server to the real server and relays responses from the real server to the proxy server. Then the proxy server relays the requests and responses between the proxy client and the real client[1].

4.6 Application Level Firewall in UML

1. As shown in Fig 4.5, Fig 4.4 and Fig 4.3, An application level firewall analyzes the complete command set for a single protocol in application space. The network packet propagates up the hardened network stack until it reaches the highest protocol layer found in the packet.[1]

2. Each network packet must pass through the low-level protocols in the kernel before being passed up the stack to application space. As shown in Fig 4.5, Fig 4.4 and Fig 4.3. When the network stack finishes processing the packet. Its data is passed from kernel space to application space to the proxy server that is listening on a specific TCP or UDP port [1].
Fig 4.3 shows how Application level Firewall Works in different layers. [39]

3. As shown in Fig 4.5, Fig 4.4 and Fig 4.3 The proxy service processes the received data it compares the data with acceptable command set rules as well as host and user permission rules to determine whether to accept or deny the packet. Proxy may perform the other functions, such as data modification, authentication, logging, URL filtering and HTTP object caching [1].

4. Once in the application space the proxies perform a thorough inspection of the packet headers and packet data. After inspection and acceptance the packet must travel back down to the kernel, and then back down the stack for distribution. Additional checks can be performed by application level firewalls to ensure that a network packet has not been spoofed [1].
Fig 4.4 Show the activity of Application level Firewall
Fig 4.5 Sequence diagram of Application level firewall
4.7 Identification of Security Threats

4.7.1 Sniffer programs

Sniffer programs monitor network traffic for usernames and passwords, subsequently making these available to the hacker [2]. A sniffer is a piece of software that grabs traffic flowing into and out of a computer attached to a network. They are available for several platforms in both commercial and open-source variations. Some of simplest packages are actually quite easy to implement in C or Perl, use a command line interface and dump captured data to the screen. More complex projects use a GUI, graph traffic statistics, track multiple sessions and offer several configuration options. Sniffers are also the engines for other programs. Network utilization and monitoring programs often use sniffers to gather data necessary for metrics and analysis. Law enforcement agencies that need to monitor email during investigations, likely employ a sniffer designed to capture very specific traffic [2].

4.7.1.1 How Does a Sniffer Work?

Every computer connected on the LAN has two addresses. One is MAC (media access control) which uniquely identifies each node in a network which is stored on the network card itself. The other is IP address, which is used by the applications. The Data link layer uses an Ethernet header with the MAC address of the destination machine rather than the IP address. The Network Layer is responsible for mapping IP network addresses to the MAC address as required by the Data Link Layer Protocol. It looks up MAC address of destination machine in a table usually called the ARP (address resolution protocol) cache. If no entry is found for the IP address, the address Resolution Protocol broadcast a request packet to all machines on the network. The source machine in all its communications with the destination machine then uses this MAC address [40].
As shown in Fig 4.6, the sniffer program tells a computer, specifically its Network Interface Card (NIC), to stop ignoring all the traffic headed towards other computers and pay attention to them. It does this by placing the NIC in a state known as promiscuous mode. When two computers want to communicate with each other, they send packets on the network with the destination MAC address. All computers on the Ethernet compare frames’ destination MAC address with their own. A machine running a sniffer program breaks these rules and accepts all frames. Such a machine is said to have been put into promiscuous mode and can effectively listen to all the traffic on the network [40]. Once a NIC is promiscuous, a status that requires administrative or root privileges, a machine can see all the data transmitted on its segment. The program then begins a constant read of all information entering the PC via the network card. Data traveling along the network comes as frames, or packets, bursts of bits formatted to specific protocols. Because of this strict formatting, a sniffer can peel away the layers of encapsulation and decode the relevant information stored within: source computer, destination computer, targeted port number, payload, in short - every piece of information exchanged between two computers [8].
4.7.2 Cracker programs

As shown in fig 4.7 and 4.8. These programs, widely available on the Internet, run in background mode on a machine, encrypting thousands of different words and comparing these to the encrypted passwords stored on the machine [2]. Crack programs are software programs that are used to break passwords these methods are based on vulnerabilities existing in crypto algorithms and their implantation. These vulnerabilities exist in case of absolutely week algorithms it may be one byte patching method in which one byte in the program will result in correct decrypting without right password [41].

In case of secure algorithms (when attacker can only generate passwords and check them) two main methods exist - brute force attack and dictionary attack.

4.7.2.1 Brute force attack

Brute force attack is used when there is no additional information on password and attacker simply tries all possible passwords - one-characters, two-characters etc. brute force attack is a method of defeating a cryptographic scheme by trying different possibilities for example trying with all possible combination in order to decrypt a message [42].

4.7.2.2 Dictionary Attack

If cracker knows that the password is a certain word they use dictionary attack. Then only words from dictionary are tested as password candidates. A dictionary attack refers to the general technique of trying to guess some secret by running through a list of likely possibilities, often list of word from the dictionary. Dictionary attacks use words which are frequently use and use them with different possibilities [41, 42].

4.7.2.3 Ruled Based Attack

In rule base attack cracker obtains some information about the password for example the cracker knows the one or two digits number. Cracker writes the rule and program generates suitable passwords [41,42].
Fig 4.7 shows relations between victim and attacker using Abuse Use Case.
4.7.3 Port Scanners

These programs, again available freely on the Internet, will send messages to all the TCP and UDP ports on a remote computer to see if any of them are open and waiting to receive a call. Once an open port has been located, the hacker will then try to get in to the computer through it TCP and UDP use port numbers to identify higher layer services. Systems administrators use port scanners to determine what TCP/UDP services are available on a server. A basic rule of server security is to disable any service that the system isn't using because any open TCP/UDP service offers intruders a possible entry into your system. You can use a port scanner to ensure that only the desired TCP/UDP services are running.

Port numbers 0 through 1023 are well-known ports that systems administrators usually use for system processes or for programs that privileged users are running. If attackers exploit a well-known port, they can potentially gain control of a server. Attackers use several generic schemes to scan ports [2]
A Port Scan is one of the most popular reconnaissance techniques attackers use to discover services they can break into. All machines connected to a Local Area Network (LAN) or Internet run many services that listen at well-known and not so well known ports. A port scan helps the attacker find which ports are available (i.e., what service might be listing to a port). Essentially, a port scan consists of sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can therefore be probed further for weakness.

4.7.4 Simple Port Scan Techniques

There are different port scanning techniques which are described as follow

4.7.4.1 Stealth port scan

The stealth scans send a single frame to a TCP port without any TCP handshaking or additional packet transfer. Hacker can only make use of open port; if a port is closed they cannot do anything [14].

When a computer makes a connection to a remote machine, it connects to a port number (between 1 and 65535). If the port is ‘open’ then a reply is sent saying the connection was successful, if the port is ‘closed’ then a rejection message is sent. This is very much like going to a house and reaching for the door handle to open the door [13].

Stealth is considered to be any scan that is concerned with a few of the following:

- Setting individual flags (ACK, FIN, RST)
- NULL flags set
- All flags set
- Bypassing as casual network traffic
- Varied packet dispersal rates

The scans described below use of inverse mapping technique for open port assumptions [12].

4.7.4.2 SYN|ACK scanning

The theory behind this method is not unlike the SYN method. A standard response would act as follows:

4.7.5 TCP Connection Establishment

As shown in Fig 4.9(a), TCP connection establishment uses a three-way handshake. The first step of connection establishment is when the client sends a TCP packet with the SYN flag set (a TCP SYN).
Upon receiving this packet, the server responds with the TCP packet with both the SYN and ACK flags set (a TCP-SYN/ACK). When the client receives the TCP-SYN/ACK or with a TCP data packet with the ACK flag set or with a TCP data packet that also has ACK flag set. When the server receives the ACK from the client, the connection establishment phase of TCP is complete [12].
As mention in Fig 4.9 (b), if the client does not receive a SYN/ACK from the server, it will resend the SYN typically after waiting for 3 seconds. If SYN/ACK still does not arrive, the client will send another SYN after 6 seconds. In most implementations, the maximum time between SYNs is 64 seconds. If server does not receive ACK with the time out period, it will resend the SYN/ACK in the same pattern as the SYNs are sent [12].
Fig 4.10 Sequence diagram of SYN|ACK Scanning
As shown in 4.10, the server ignores the SYN|ACK packet sent to an open port. Imagine sending a SYN|ACK packet and receiving no response due to stately packet filters, firewalls or even timeout limits blocking transmission, thus the scanner would then produce false positives for that port. Naturally this scan is not considered as reliable as TCP connect() scans because of this very reason. This type of assumption falls under what is known as "inverse mapping" [12].

4.7.6 FIN scanning

The FIN scan method uses inverse mapping to discover closed ports. Unfortunately, this techniques relies on bad BSD network code which most operating systems have based their TCP/IP stacks on (all the better for scanning). Ideally, once a FIN flagged packet is sent, a closed port will resend with an RST bit. Open ports, alternatively will not send a packet back, therefore what precisely is not answered with the FIN bit, is assumed to be open through this process of inverse mapping [12].

Take a look at the negotiation for open/closed port recognition displayed 4.11.

![Fig 4.11 Sequence diagram of Fin Scanning](image)

No reply signaled by the server is iconic of an open port. The server's operating system silently dropped the incoming FIN packet to the service running on that port. Opposing this is the RST reply by the server upon a closed port reached. Since, no service is bound on that port, issuing a FIN invokes a reset (RST) response from the server [12].

Arguably there are two ways to test for an open port. The first is receiving a list of closed port responses and subtracting these port replies from a list of the port probes originally sent. For example, sending 3 packets to ports 1, 2, 3 on a remote host [12].

If the response back is an RST for ports 1 and 3, we then compare the original port list: 1, 2, 3 to the received ports: 1, 3 and deduce that 2 is the open port via comparison [12].
The second test involves using a timeout for the packet response. If the timeout limit is reached to receive the packet in question then we assume it to be open. Obviously, this method is test for false positives and should be avoided where possible. The responses for the packet could be obscured because of firewalls, filters, routers, slow links, and heavy traffic, thus is not a solid test to be used as a rule of thumb for stealth FIN scanning [12].

4.7.7 UDP Scanning

Port scanning usually means scanning for TCP ports, which are connection-oriented and therefore give good feedback to the attacker. UDP responds in a different manner. In order to find UDP ports, the attacker generally sends empty UDP datagram. As shown in Fig 4.12. If the port is listening, the service should send back an error message or ignore the incoming datagram. If the port is closed, then most operating systems send back an "ICMP Port Unreachable" message. Thus, hacker can find out if a port is NOT open, and by exclusion determine which ports are open. Neither UDP packets, nor the ICMP errors are guaranteed to arrive, so UDP scanners of this sort must also implement retransmission of packets that appear to be lost (or you will get a bunch of false positives) [10].

Fig 4.12 Use Case of UDP Scanning threaten by the attacker
For the hacker it is critical to identify listening ports, because it helps him identify the operating system and application in use. The services detected as listening may suffer from vulnerabilities which may result from two reasons [11].
- Misconfiguration of the service
- The version of the software is known to have security flaws

### 4.7.7.1 TCP connect() scan

With this type of scan we use the basic TCP connection establishment mechanism. To open a connection to an interesting port on the targeted machine:[11]

1. A SYN packet is sent to the target’s system interesting port [11].

2. Now we wait to see what type of packet is sent back from the target [11].

   If a SYN/ACK packet is received it usually means the port is in a LISTENING state [11].

   If a RST/ACK packet is received, it usually means the port is not LISTENING and the connection will RESET [11].

3. We finish the three-way handshake (if SYN/ACK packet was received) by sending an ACK [11].

4. A connection is terminated after the full connection establishment process has been completed. This kind of scan is easily detected. Inspecting the target system log will show a number of connections and error messages immediately after each one of them was initiated. [11]

### 4.8 Assumptions of Application level firewall

These are conditions that are assumed to exist in Basic PP.

- F.AUTHORIZE states that it must prevent the assets from unauthorized access.

- F.SECURITY POLICY states that Firewall must be configured to deny all network connection aimed directly at the firewall host.

- F.USER_ACCOUNT states that network must be configured with user accounts and passwords.

- A.NO_ENCLAVE_PROTECTION deals with the flow of information between internal and external networks [36].

- A.NO_EVIL states that assets cannot intentionally be hostile, be properly trained and follow the guidelines set out [36].
• **A.NO_PUBLIC_DATA** states that the TOE only holds TOE data and therefore does not hold public data [36].

• **A.REMOTE_USERS** states only the remote cannot access the network with authorization [36].

Fig 4.13 defines the relation between threat agent, TOE developer and user this figure define how threats are mitigated between use/abuse cases.
Fig 4.13 Use case of Assumptions of Application level firewall
4.9 Threats to the TOE

Threat agents who are either unauthorized persons or external IT entities perpetrate the threats. The possible threats the TOE may face from a threat agent are listed below.

- **T.ADDRESS_SPOOFING.** A threat agent attempt to gain access to one of the physically connected internal network, or user gaining access to a system on an opposing side of the firewall, by sending an IP packet with a fake source address. [36]

- **T.CRACKSOFTWARE** states that threat agent can use brute force and dictionary attacks in order to gain access to the network

- **T.EXLOITING** states that an attacker compromises the assets by exploiting the use of tools which contain know security faults.

- **T.ATTACK_CONFIGURATION_DATA.** A threat agent may try to read, modify, or destroy security-critical data.[36]

- **T.PORT_SCANNING** states threat agent can use different port scanning techniques on targeted port.

- **T.ATTACK_POTENTIAL.** In the basic the threat agent is only using obvious vulnerabilities to attempt to circumvent the FTP services.

- **T.BRUTE_FORCE.** A threat agent tries to repeatedly guess authentication data in order to launch an attack against the TOE.

- **T.KEY_COMPROMISE.** A threat agent with the use of stolen or compromised cryptographic keys may decrypt sensitive data and gain unauthorized access to sensitive data.[36]

- **T.MASQUERADE.** A threat agent through the use of stolen or compromised cryptographic keys may masquerade as a peer TOE and thus gain unauthorized access to sensitive data. Also, through the use of captured ID and authentication data, they could masquerade as an AA.[36]

- **T.REMOTE_ATTACK.** A threat agent may be able to view, modify, and/or delete security-related information that is sent between a remotely located AA and the TOE.[36]

- **T.RESIDUAL_INFO.** A threat agent may attempt to gather residual information from previous information flows or internal TOE data in order to gain authorized access to sensitive data.[36]
• T.SERVICE_MISUSE. A threat agent on the internal network may try to connect to services other than those expressly permitted. Additionally, a threat agent may attempt to send information through the TOE in order to exploit resources on the internal network.[36]

• T.UNAUTHORIZED_BYPASS. A threat agent may attempt to bypass the security of the TOE so as to access and use security functions and/or non-security by exploiting vulnerabilities [36]
Chapter 5

Modeling FTP Vulnerabilities Using UML

This chapter explains some FTP vulnerabilities which can be exploited by the threat agents. It is very important to address the vulnerabilities because ignorance of vulnerabilities weaken the TOE.

5.1 Insecure software.

Internet software, particularly shareware, free or low cost packages often have bugs or design flaws in them usually as a result of poor design or insufficient testing of the software. But due to their ready availability and low cost, many people still take the packages. Examples include: the UNIX send mail program which has had numerous vulnerabilities reported in it, and a freeware FTP product which contained a Trojan Horse that allowed privilege access to the server.
As shown in fig 5.1, A Trojan horse is code hidden in a program such as a game or spreadsheet that looks safe to run but has hidden side effects. During program execution, it seems to function as the user expects, but in fact it is destroying, damaging, or altering information in the background. It is a program on its own and does not require a host program in which to embed itself. Unscrupulous people are always ready to exploit these weaknesses [2].
5.2 Vulnerability of the FTP (Bounce Scan)

The FTP bounce attack is an exploit of the FTP protocol whereby an attacker use PORT command to request access to ports indirectly using victim machine. This port scan can be used to access the ports that the attacker cannot access through direct connection [10, 11]. FTP bounce scanning takes advantage of a vulnerability of the FTP protocol itself. It requires support for proxy ftp connections. This bouncing through an FTP server hides where the attacker comes from. This technique is similar to IP spoofing in that it hides where the attacker comes from [10,11].

5.2.1 FTP session phases

1. The FTP client opens a connection to the FTP control port (normally, port 21/TCP) of the FTP server. A second (data) connection is required to be opened between the server and the client and later to allow the server to send back data which the client has request to the client machine at the later time [43].

2. The client wants to download a file from the FTP server. To establish the data connection. The client sends out a FTP PORT command to the server machine. This command includes parameter with IP address that instructs server machine and port in order to download the file to the client [43].

5.2.2 FTP Bounce Scan Scenarios

As shown in Fig 5.2. When FTP server or border devices that allow the FTP client to specify an IP address different than its own in the FTP ‘PORT’ command which makes insecure situation.

The situation could allow attacker to carry out a port scan against a host on the internal from third party FTP server that is going to act as an intermediate host. This offers more stealth for the attacker as the victim site will see that scan originates from the third party and not from the attacker [43].
Fig 5.2 Sequence diagram of Bounce scan attacker
Chapter 6:

Discussions

This chapter discusses security view of UML diagrams.

6.1 Security Analysis

Security analysis is performed in order to secure assets in organization; security analysis is performed to protect assets from attackers. As new technologies and business model come into use in different social technical context protection of value able asset become even more complex. The main purpose of this paper is to integrate the threats in operating environment with unified modeling language and capture the behavior of the threats. There are different approaches used in security analysis (e.g functional decomposition or use case modeling) were designed for the analysis of functional requirement, which are used commonly to analyze security requirements [49]. According to agent oriented software engineering, security is the non functional requirement. Functional requirements capture behavior and task of the system. The security modeling represents the connection between threats, threat agents, objectives and security mechanisms which presents how to protect assets against different attacks. Security modeling shows the behavior of the security environment using different actor and use case. Threat on the other hand presents circumstances that can put in danger the security features of the system. Security as non-functional requirement decomposes different entities [49].

6.2 Role of UML in Security Modeling

Modeling is performed in order to capture requirement, it captures the important aspects of the things to be modeled. It defines different aspects of a system and ignores aspects which are not important. UML can perform identification of security threats as part of requirement elicitation and model them using Use Case. UML sequence diagram is used to describe the decision making process an attacker would go through and harm the assets [49]. As Use case modeling is a proven method for the elicitation of various system requirements, both functional and non-functional a Use cases based security requirements specification can further facilitate the design and implementation of Protection Profile where asset protection is the major issue [49,51,5]. Thus analyst can analyze threat, and threat agents and security use cases that can help TOE developer to understand the systems desired response to threats. The goal of environment analysis is to protect assets by identifying threats and threat agents. TOE environment analysis different threats and vulnerabilities are identified and modeled using UML [52].

The advantage of using UML for TOE environment analysis is to compliant security threats and different constraints should be imposed which depend on the choices of implementation techniques in order to mitigate the threats. If [49], we consider the general development cycle, the treatment of security threats in the earlier phases of system development can reduce overall development cost due to the absence of a variety of vulnerabilities.
Moreover if we know the mitigation approaches, we can proceed with both development and use case analysis, which is initially top down. We can analyze threats and assumptions to evaluate the need for further countermeasures. Analysis and design become intertwined as design choices crystallize and the system requirements become more specific. However the analysis is performed using Sequence diagrams to identify possible failure modes and sequence of action as it gives the visual presentation of the threats [49,50].

6.3 Identification of Patterns

Modeling is a technique used to specify patterns mainly framework and collaborations. The modeling includes the threats and assumptions analysis that drives into requirements. The framework includes a separate view that provides an abstraction to show the relationship between the threats and policies. The modeling shows Common Criteria requirement as attributes of the objects and components. The modeling framework should include a separate view that provides an abstraction that focuses on the mapping of the objects to their requirements.

Security requirements are defined by successive refinement. The articulation of requirements starts at a high level of abstraction; these requirements address the security environment consisting of threats and assumptions.

6.4 TOE Security Environment Analysis

TOE security environment consist of assumptions made on the operational environment. To study TOE security environment modeling is performed. Modeling not only captures the behavior of the TOE security environment but it represents the interaction between threat agent and threats. During development of PP the developers unaware of the complex issues of vulnerabilities. Developers perform their work in textual form which doesn’t explore the behavior and threat mitigation issues. If [37] the behavior of the threat is ignored it will weaken TOE and give edge to threat agents and harm assets. The vulnerabilities are very important to address because threat agents use different methods to harm assets.

We can achieve that objective by security requirement engineering which shows behavior in terms of Abuse/Use cases. Misuse cases show the functional and non functional requirements in terms of threat mitigation. Security Use/Abuse cases help TOE developer to understand the desired behavior of threats. The security engineering identifies the scope of protection and evaluates the trade of analysis. In this research first of all operational environment of application level firewall is described using sequence diagram and activity diagram. The security engineering gives the scope of protection to developer to understand how they can protect assets. The sequence diagram shows the decision making process an application level firewall can go through. In next step different threats are identified and examined using sequence diagrams. These sequence diagrams shows how the threat agents make different decision making to access resources. Threat
modeling using sequence diagram represent extension points to relevant aspect of security objective.
Chapter 7

Conclusion and Future Work

This chapter contains some conclusions and future work.

7.1 Conclusion

The visual presentation shows the operating environment, it visualize the behavior of threats and assumptions in TOE security environment. In Common Criteria standard functional requirements are performed using text. Security requirement engineering is performed using Use/Abuse case and sequence diagram. Which define how functional and non-functional requirements are modeled using Use/abuse Case. It defines how threat agents mitigate or threaten the security environment. Security requirement engineering is performed to address the objectives for ST. Security engineering demonstrate how threat agent exploits vulnerabilities and harm assets. It make easier for the PP developer to understand because CC doesn’t explain the method to analyze threat in operating environment. If the developer is unaware of threats behavior it is very difficult for the developer to define security objective. Security engineering defines the actor in terms of threat agents, users and different use/ abuse case are defined in terms of threat and attack methods. These relationships help PP developer to refine security objectives.

7.2 Future Work

Use/abuse case are excellent ways for analyzing security requirements. It can be effectively used in requirement eliciting for emerging environment i.e. grid environment. Use cases provide a highly reusable way of organizing, analyzing and specifying security requirement for emerging environment. The functional and non-functional requirement can easily be gathered using Use/Abuse case diagrams.
References


2. David W Chadwick, “Network firewall Technologies” IS institute, University of Salford, Source= http://sec.isi.salford.ac.uk/download/Firewalls.PDF


   Dated: 2005-08-28


36. MJ Kelly, V Matyas, A Patel, “Application level firewall protection profile for high Robustness environments initial considerations”. Dec 2004,


48. Haralambos Mouratidis, Paolo Giorgini, Gordon Manson, “Modelling Secure Multiagent System”, source: 
http://trinity.dit.unitn.it/~tropos/papers_files/SecureTroposCR.PDF


54. Diego Ray, “Integration of Security Patterns in Software Models based on Semantic Descriptions”, MCYT project

