ABSTRACT

The main objective of this thesis is to integrate threat modeling when developing a software application following the Secure Tropos methodology. Secure Tropos is an agent-oriented software development methodology which integrates “security extensions” into all development phases. Threat modeling is used to identify, document, and mitigate security risks, therefore, applying threat modeling when defining the security extensions shall lead to better modeling and increased level of security. After integrating threat modeling into this methodology, security attack scenarios are applied to the models to discuss how the security level of the system has been impacted. A security attack scenario describes an attack situation in a multiagent system. It identifies the agents, their secure capabilities, possible attackers, and their goals. Security attack scenarios have been used to test different enhancements made to the Secure Tropos methodology and the Tropos methodology itself. The system modeled using this methodology is an e-Commerce application that will be used to sell handmade products made in Ecuador through the web. The .NET Model-View-Controller framework is used to develop our case study application. Results show that integrating threat modeling in the development process, the level of security of the modeled application has increased. The different actors, goals, tasks, and security constraints that were introduced based on the proposed integration help mitigate different risks and vulnerabilities.
# TABLE OF CONTENTS

ABSTRACT .................................................................................................................................... ii  
TABLE OF CONTENTS ............................................................................................................... iii  
LIST OF FIGURES ........................................................................................................................ v  
LIST OF TABLES ........................................................................................................................ vii  
1 INTRODUCTION AND BACKGROUND INFORMATION ............................................... 1  
  1.1 Information Security ...................................................................................................... .. 2  
  1.2 Secure Tropos ............................................................................................................. ...... 3  
  1.3 Threat Modeling ........................................................................................................... .... 6  
  1.4 Security Attack Scenarios ............................................................................................ 9  
    1.4.1 Creation of the scenario ............................................................................................ 9  
    1.4.2 Validation of the scenario ....................................................................................... 10  
    1.4.3 Testing and redefinition of the system according to the scenario ....................... 10  
2 SECURE TROPOS MODEL ................................................................................................ 11  
  2.1 Early Requirements ........................................................................................................ 11  
    2.1.1 Security Enhanced Actor Diagram ......................................................................... 12  
    2.1.2 Security Enhanced Goal Diagram ........................................................................... 14  
  2.2 Late Requirements ......................................................................................................... . 20  
    2.2.1 System’s Security Enhanced Actor Diagram .......................................................... 21  
    2.2.2 System’s Security Enhanced Goals Diagrams ........................................................ 23  
  2.3 Architectural Design ...................................................................................................... 25  
3 THREAT MODELING ......................................................................................................... 27  
  3.1 Identify the known threats to the system........................................................................ 27  
  3.2 Rank and group the threats in order by decreasing risk ................................................. 28  
  3.3 Determine how to respond to threats and identify techniques that mitigate threats ..... 29
LIST OF FIGURES

Figure 1. Tropos Concepts [3, 16, 14] ............................................................................................ 4
Figure 2. Types of Dependency [3, 16, 14] .................................................................................... 5
Figure 3. Example of Security Constraints [3, 16, 14] ................................................................ 6
Figure 4. Security Enhanced Actor Diagram ................................................................................ 12
Figure 5. Security Enhanced Goal Diagram ............................................................................... 15
Figure 6. Customer Goal Diagram ............................................................................................. 16
Figure 7. Employee Goal Diagram .............................................................................................. 17
Figure 8. Manufacturer Goal Diagram ....................................................................................... 17
Figure 9. Online Store Goal Diagram ......................................................................................... 18
Figure 10. Payment Processing Agency Goal Diagram ............................................................... 19
Figure 11. Provider Goal Diagram ............................................................................................... 19
Figure 12. Store Admin Goal Diagram ......................................................................................... 20
Figure 13. System’s Security Enhanced Actor Diagram ............................................................. 22
Figure 14. System’s Security Enhanced Goals Diagrams ......................................................... 23
Figure 15. Architectural Design ................................................................................................. 26
Figure 16. Security Enhanced Goal Diagram with Threat Modeling ......................................... 32
Figure 17. System Developer Goal Diagram .............................................................................. 33
Figure 18. System’s Actor Diagram with Threat Modeling ....................................................... 34
Figure 19. Test Case 1: Data Modification through SQL injection ......................................... 36
Figure 20. Test Case 2: Denial of Service ..................................................................................... 37
Figure 21. Test Case 3: Fraudulent transaction .......................................................................... 39
Figure 22. Application: Index .................................................................................................... 40
Figure 23. Application: Browsing................................................................................................. 41
Figure 24. Application: Add to Cart ............................................................................................. 41
Figure 25. Application: Log On.................................................................................................... 42
Figure 26. Application: Shipping Information ............................................................................. 42
Figure 27. Application: End Purchase .......................................................................................... 43
Figure 28. Models ......................................................................................................................... 44
Figure 29. Views........................................................................................................................... 44
Figure 30. Controllers .................................................................................................................. 45
Figure 31. Syntax ......................................................................................................................... 46
Figure 32. Account Controller ...................................................................................................... 47
Figure 33. ASP .net Web Site Administration Tool ..................................................................... 47
Figure 34. [Authorize] Attribute ............................................................................................... 48
Figure 35. [RequireHTTPS] Attribute ....................................................................................... 49
Figure 36. Java Script Injection Attack ....................................................................................... 51
Figure 37. Injection Attack Detection ......................................................................................... 51
LIST OF TABLES

Table 1. Threat Type and Mitigation Technique ................................................................. 8
Table 2. Actor – Goal – Task ............................................................................................ 29
Table 3. Threat and Mitigation Technique ....................................................................... 34
1 INTRODUCTION AND BACKGROUND INFORMATION

Information security has always been an important issue to be addressed when developing a software application. Unfortunately, it appears to only have been taken into account during the last implementation phases. This is not a result of carelessness. Software developers are aware of the importance of security. Software developers have not come to fully comprehend and/or accept that security is an aspect that affects all the different components of an application. Therefore it should be integrated into the development process from the commencement through the last phase. Another reason why security has not been well integrated in the software development process is that there are not enough tools that can handle security engineering and its integration with the development process. In addition to the lack of tools, there are not enough software developers who are also security experts of whom can create guidelines in which to follow when developing secure software [15]. Some methodologies have been developed to overcome these problems and be able to integrate security while developing a software application.

One of the approaches that have been developed to integrate information security in the software development process is the Secure Agent Oriented Software Development Methodology Tropos. As this methodology is agent oriented, it describes the system and the environment it interacts with in each of its five phases. As well, it integrates “security extensions” into all the phases. By integrating security from the early development phases, the software product will be considered a secure application [3].

Threat modeling is the process of identifying, documenting and mitigating security risks [1] resulting from threats and vulnerabilities in a system. A security threat is an action that attackers can perform to violate a security goal [26]. According to the Microsoft Development
Software Network (MSDN), threats can be grouped using the STRIDE model which defines the following categories: spoofing identity, tampering with data, repudiation, information disclosure, denial of service, and elevation of privilege [18, 20]. To create a threat model the following steps can be followed: identify the known threats to the system, rank and group the threats in order by decreasing risk, determine how to respond to the threats, identify techniques that mitigate the threats and choose the appropriate technologies from the identified techniques [21].

The threat model resulting from this process can be used as the base to determine which security extensions should be added to the system's model when using the Tropos methodology.

1.1 Information Security

Information security can be defined in many ways depending on the personal point of view therefore many definitions have been given. One of them states that information security is "The protection of information systems against unauthorized access to or modification of information, whether in storage, processing or transit, and against the denial of service to authorized users of the provision of service to unauthorized users, including those measures necessary to detect, document, and counter such threats" [21]. This definition puts emphasis in the three key aspects of security: confidentiality, integrity and availability.

Integrating information security in the software development process has become fundamental in order to deliver a high quality system; therefore many methodologies that implement information security in the software development process have been created. In this thesis, the Secure Tropos methodology will be used.
1.2 Secure Tropos

This methodology is an agent oriented methodology based on the Tropos methodology [11]. The Tropos methodology was developed because of the need of a flexible architecture that could allow developers make changes or incorporate additional requirements more easily.

In agent oriented methodologies, there is an agent that has its own goals. To reach these goals, it interacts with the environment and other agents that surround it. Because of this interaction with the environment and other agents, in this methodology, the main focus is set on plans and actions to fulfill the goals instead of procedures and methods as well as in ways to communicate and negotiate instead of mechanical functionalities.

Based on these approaches, the Tropos methodology was created [3, 16, 14, 22]. This methodology focuses on the different phases of system requirements analysis as well as system design and implementation putting emphasis on the early requirements analysis phase. Secure Tropos incorporates security extensions in each of the different phases.

The different phases that Tropos define are:

- Early Requirements: in this phase the developers study an organizational setting that has already been defined and try to understand what the problem is. In this phase, the different actors and their dependencies are defined.
- Late requirements: in this phase, in addition to the actors and their dependencies, functions, and qualities are defined. Based on the actors and dependencies, the functional and non-functional requirements of the system are defined.
- Architectural design: in this phase, different subsystems define the global architecture of the system. The different subsystems (actors) are connected through data and control flows (actor dependencies).
Detailed design: in this phase, the global architecture is defined with more details. Here, inputs, outputs, and other important information are defined.

This methodology uses the i* modeling framework [3, 16, 14, 22]. This framework uses the concepts of actors, tasks, softgoals, goals, and resources. An actor can be any agent that has a goal or softgoal. Depending on the goal that an actor has, he or she will have a task. The actor can have access to a set of resources in order to successfully complete the tasks and fulfill their goals. There exists a dependency relationship between actors. In this framework, actors (depender) can depend on other actors (dependee) in order to fulfill their goals; therefore, the system can be considered a set of actors who depend on each other to fulfill their goals.

![Figure 1. Tropos Concepts][3, 16, 14, 22]

In addition to these concepts shown in Figure 1, Secure Tropos also includes security extensions to the concepts and to the dependencies. A security constraint is a restriction that is related to security issues (privacy, integrity and availability). This constraint influence in the analysis and design of the system because a security constraint will help achieve a secure goal along with the other two secure entities. In order to have secure dependencies, constraint labels are included in the relationships between dependees, dependum and dependees.

There are three kinds of secure dependencies:
a. Depende Secure Dependency: Constraint label is located between the depender and the dependum.

b. Depender Secure Dependency: Constraint label is located between the dependum and the dependee.

c. Double Secure Dependency: Constraint labels are located between the depender and the dependum and between the dependum and the dependee.

These kinds of dependencies are shown in Figure 2.

![Figure 2. Types of Dependency][1]

**Figure 2. Types of Dependency [3, 16, 14]**

These concepts and security extensions can be represented in Figure 3 which shows the actors model for an Electronic Single Assessment Process. This is an integrated health and social care information system for the care of older people.
Figure 3. Example of Security Constraints [3, 16, 14]

This model shows how the different actors (Older Person, Professional, DoH, R&D Agency, and Benefits Agency) depend on each other to fulfill their goals. It also shows the security extensions integrated to the relationships. For example, in the goal Obtain OP Personal Information that the Professional has, he depends on the older person and at the same time it has to take into account that this info can be shared only if the consent is achieved [3, 16, 14].

1.3 Threat Modeling

Threat modeling is a process that should be done in the design process [4] in order to identify, document and mitigate security risks [1]. Through threat modeling, the security of an application can be defined, potential threats, vulnerabilities, and bugs can be identified in an early phase, and documents to create security specifications and testing can be created [4].
The procedure for creating a threat model is [17, 21]:

1. Identify the known threats to the system: In order to identify any threats to the system, assets, either tangible or intangible [24], have to be identified first because they are an attacker's main goal. Once the assets have been identified, threats against those assets can be derived. Entry points to the system should be considered because those are the vulnerabilities that an attacker will look into when trying to gain access to the system's assets. After the assets are identified, boundary levels have to be defined. These boundary levels will help specify who can access the assets and who cannot. Boundary levels will help determine if there is a threat to an asset according to the access privileges.

2. Rank and group the threats in order by decreasing risk: After the assets and the threats have been identified, these threats should be grouped and categorized. For this, the STRIDE model can be used. STRIDE defines six categories:

   1. Spoofing identity: Illegally using someone's identity such as user name and password in order to gain access to the system. Duplicate user names can also fall into this category.
   2. Tampering with data: modification of data with a malicious purpose.
   3. Repudiation: denial of performing an action that could give the system a proof of what procedure has been executed.
   4. Information disclosure: protected data is exposed to unauthorized users.
   5. Denial of service: valid users are not able to access the system.
   6. Elevation of privilege: attacker uses illegitimate mechanisms to gain trust levels that he does not have.
3. Determine how to respond to the threats: after having identified the threats, mitigation techniques for each threat should be determined.

4. Identify techniques that mitigate the threats: different techniques have been specified to mitigate each STRIDE category [21]. These different threats and mitigation techniques are shown in Table 1.

<table>
<thead>
<tr>
<th>Threat type</th>
<th>Mitigation technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spoofing identity</td>
<td>• Authentication</td>
</tr>
<tr>
<td></td>
<td>• Protect secrets</td>
</tr>
<tr>
<td></td>
<td>• Do not store secrets</td>
</tr>
<tr>
<td>Tampering with data</td>
<td>• Authorization</td>
</tr>
<tr>
<td></td>
<td>• Hashes</td>
</tr>
<tr>
<td></td>
<td>• Message authentication codes</td>
</tr>
<tr>
<td></td>
<td>• Digital signatures</td>
</tr>
<tr>
<td></td>
<td>• Tamper-resistant protocols</td>
</tr>
<tr>
<td>Repudiation</td>
<td>• Digital signatures</td>
</tr>
<tr>
<td></td>
<td>• Timestamps</td>
</tr>
<tr>
<td></td>
<td>• Audit trails</td>
</tr>
<tr>
<td>Information disclosure</td>
<td>• Authorization</td>
</tr>
<tr>
<td></td>
<td>• Privacy-enhanced protocols</td>
</tr>
<tr>
<td></td>
<td>• Encryption</td>
</tr>
<tr>
<td></td>
<td>• Protect secrets</td>
</tr>
<tr>
<td></td>
<td>• Do not store secrets</td>
</tr>
<tr>
<td>Denial of service</td>
<td>• Authentication</td>
</tr>
<tr>
<td></td>
<td>• Authorization</td>
</tr>
<tr>
<td></td>
<td>• Filtering</td>
</tr>
<tr>
<td></td>
<td>• Throttling</td>
</tr>
<tr>
<td></td>
<td>• Quality of service</td>
</tr>
<tr>
<td>Elevation of privilege</td>
<td>• Run with least privilege</td>
</tr>
</tbody>
</table>

5. Choose the appropriate technologies from the identified techniques: after having identified the threat and the category, the appropriate mitigation technique should be applied.
1.4 Security Attack Scenarios

A security attack scenario is an attack situation where the agents of a multiagent system, their secure capabilities and possible attackers with their goals are described. A security attack scenario helps identify how the secure capabilities of the system prevent the attacker from achieving his goals. A security attack scenario should contain most of the characteristics of the system so that its security requirements can be validated [13].

The main elements of an attack scenario are [13]:

- Possible attack
- Possible attacker
- Resources attacked
- Agents related to the attack

To model a security attack scenario, the same methodology as Tropos is used. The actor is the attacker who has goals and tasks to achieve the goals. The attacks are depicted as dash-lined links (attack links) [13].

The creation of a security attack scenario consists of three stages [13]:

1. Creation of the scenario
2. Validation of the scenario
3. Testing and redefinition of the system according to the scenario.

The different stages can be interchanged and reiterated according to the developer’s needs [13].

1.4.1 Creation of the scenario

In this stage, the intensions of the attacker are analyzed in terms of goals and tasks. A set of attacks need to be identified according to the attacker’s goals. Based on the attacks, a set of
agents who have capabilities to prevent these attacks are identified. The attack and the agent will be associated through the dashed-lined link [13].

1.4.2 Validation of the scenario

After the scenario has been created, it needs to be validated. For this, software inspections are used. The main purpose of the inspections is to detect any possible violations or inconsistencies of the Tropos syntax. To do this, validation check lists are used. Other mechanisms can also be used depending on the developer’s experience and the system itself [13].

1.4.3 Testing and redefinition of the system according to the scenario

After the scenarios have been created and validated, test cases are identified and are used to test the security of the system.

The tests cases are depicted from the security attack scenarios and should include [13]:

- Precondition: system’s state before the attack
- System expected security reaction: the system’s reaction to the attack
- Discussion: discussion for deciding result of the test case
- Test case result: output of the test case

The test cases help the developers decide whether the system can prevent the attacks or not. If it has been determined that an attack cannot be prevented, the developer needs to assign extra capabilities to the system.
2 SECURE TROPOS MODEL

The system to be modeled using the Secure Tropos methodology is an e-commerce application that will be used to sell handmade products from Ecuador through the web. The customers will be able to browse through the catalogue of products and add them to a cart. They will also be able to use a search engine to look for any specific items they are looking for. The catalogue will have different categories to make it easier for customers to find different products. Customers will have the option to create an account which will help them for future purchases. Their account will contain their personal and shipping information and the products they have purchased. For the payments, a third party, PayPal will be used. Employees will be in charge of placing the orders and dispatching them. Employees also will have to handle any issues that the customers have such as returns, exchanges, etc. The store’s administrator will use his business skills to make the online store provide an excellent service, create marketing strategies to sell the products and will be in charge of managing the stock of products to make sure that the customers get their products on time.

2.1 Early Requirements

In this first stage, the organizational setting is studied and the different actors, goals and dependencies are identified.

The different actors that have been defined for this model are:

- Customer: Person who will make his purchase in the online store.
- Online Store Administrator: Person that will be in charge of the store’s management.
- Provider: A wholesale retailer that will provide the products for the online store.
- Employee: Person who will take care of the online sales (dispatch, place orders) and verify customer’s identity.
- Post office: Facility authorized to deliver the products to the customers.
- Payment processing agency: Agency that will take care of the payment processing.

The result of the system’s analysis in this stage is the Security Enhanced Actor Diagram and the corresponding Security Enhanced Goals Diagrams.

2.1.1 Security Enhanced Actor Diagram

Figure 4 shows the security enhanced actor diagram depicting each actors, goals, soft goals, security constraints and the dependencies that they have with each other.

![Figure 4. Security Enhanced Actor Diagram](image)
2.1.1.1 Customer

The actor Customer’s main goal is Buy Handmade Products. For this, he depends on the online store, so the actor Online Store becomes the dependee and the actor Customer the depender. At the same time, the security constraint Protect Customer’s Identity is imposed by the actor Customer in this relationship.

2.1.1.2 Online Store Administrator

The actor Online Store Administrator’s main goals are: Sell the products, Manage the stock, and Obtain the products. To fulfill the goal Sell the products and Manage the stock, this actor (dependee) depends on the actor Online Store (dependee) and to fulfill the goal Obtain the products it depends on the actor Provider (dependee).

2.1.1.3 Online Store

The actor Online Store’s goals are: Place Order, Dispatch Products, and Troubleshoot Customers’ issues. This actor’s soft goals are: Become a popular store and Provide an excellent service. To fulfill the goals Place the order, Dispatch the products, and Troubleshoot Customers’ issues, this actor (dependee) will depend on the actor Employee (dependee). To achieve the soft goal Become a popular store, it will depend on the actor Customer (dependee), and finally, to achieve the soft goal Provide an excellent service, it will depend on the actor Online Store Administrator (dependee). The security constraint Keep Customer’s Info Safe is introduced in the dependency link for achieving the Place order and Dispatch products goals.

2.1.1.4 Provider

The actor Provider’s main goal is Commercialize the products. For this, actor Provider (dependee) depends on the actor Online Store (dependee). This actor uses the resource Handmade products for which he depends on the actor Manufacturer’s (dependee).
2.1.1.5 Employee

The actor Employee’s main goal is verify Customer’s identity. For this, it (depender) depends on the actor Payment Processing Agency (dependee).

2.1.1.6 Payment Processing Agency

The goal of the actor Payment Processing Agency is Process the payment. This depender depends on the actor Customer (dependee) who is going to provide with all the necessary information to be validated. The security constraint Keep Customer’s information secure is introduced in the relationship.

2.1.2 Security Enhanced Goal Diagram

After having identified the different actors, goals, and dependencies, a deeper analysis of the diagram is performed. The results of this analysis warrant several goals within the diagram. Each actor is separated, and each goal that the actor helps fulfill, wherein the actor is described. Tasks are introduced in this diagram.

Figure 5 shows each actor, the goals that depend on them, and the tasks that they have to perform in order to achieve the aforementioned goals.
Figure 5. Security Enhanced Goal Diagram
2.1.2.1 Customer

For the goal Process Payment to be achieved by the actor Payment Processing Agency, the task that the actor Customer is going to perform is Enter information. Similarly, for the soft goal Become a popular store to be achieved by the actor Online Store, this actor is going to have to perform the task Give positive feedback (Figure 6).

Figure 6. Customer Goal Diagram

2.1.2.2 Employee

The actor Employee is going to help the actor Online Store achieve its goal Dispatch Order by performing the tasks Check Order and Verify Identity. The Actor Online Store also depends on this actor to achieve the goal Place Order. For this, the tasks that the actor Employee is going to perform are Verify Products and Retrieve Order. One last goal that this actor is going to help achieve is Troubleshoot Customer’s issues. This goal is going to be achieved by performing the
task *Retrieve Order*. The security constraint *Keep Customer’s info safe* imposed by the actor *Customer* is also included in this goals diagram (Figure 7).

**Figure 7. Employee Goal Diagram**

### 2.1.2.3 Manufacturer

The actor *Manufacturer* is only involved in providing the resource *Handmade Products* (Figure 8).

**Figure 8. Manufacturer Goal Diagram**
2.1.2.4 Online Store

The actor Online Store is going to help the actor Provider, achieve its goal Commercialize Products, by performing the task: Provide easy to browse catalog. This task is also going to help this actor achieve the actor Customer’s goal Buy Handmade Products. The actor Online Store is also going to help the actor Store Admin achieve its goal Manage stock by performing the task Check stock periodically (Figure 9).

![Figure 9. Online Store Goal Diagram](image)

2.1.2.5 Payment Processing Agency

For the actor Payment Processing Agency to help the actors Employee and Customer achieve their goals Verify Customers’ identity and Process Payment respectively, it is going to perform the task Contact Bank (Figure 10).
2.1.2.6 Provider

The actor Provider is going to help the actor Manufacturer achieve its goal Sell Products by performing the task Advertise Product. This actor is also going to help the actor Online Store achieve its goal Provide efficient Service by performing the tasks Have high stock of products and Contact Provider. The task Contact Provider is also going to help the actor Store Admin to fulfill its goal Obtain Products (Figure 11).
2.1.2.7 Store Admin

The actor Store Admin is going to help the actor Online Store achieve its goal *Provide an excellent service* by performing the tasks *Evaluate employee’s performance*, *Create a good marketing plan* and *Analyze store’s necessities* (Figure 12).

![Store Admin Goal Diagram](image)

Figure 12. Store Admin Goal Diagram

2.2 Late Requirements

After having finished the early requirements analysis of the system where the different actors, goals and dependencies were identified, the system is again analyzed taking into account all the functional and non-functional requirements.

In this stage, the system is introduced as an actor. The system’s security is further analyzed therefore new security constraints are added as well as new goals. As a result of this analysis, the System’s Security Enhanced Actor Diagram and the corresponding System’s Security Enhanced Goals Diagrams are generated.
2.2.1 System’s Security Enhanced Actor Diagram

Figure 13 depicts the different actors that either depend on the system to fulfill their goals or are dependees for the system to fulfill its goals. New security constraints have been introduced to the system.

2.2.1.1 Employee

The actor Employee depends on the system to fulfill the goals: Retrieve Client info, Troubleshoot Customer’s problems, Process Orders, Retrieve Orders and Process Shipping. The security constraints Restrict only to authorized personnel and Keep Customer’s Info private are imposed to the system.

2.2.1.2 Store Admin

The actor Store Admin depends on the system to fulfill the goals: Create Marketing Strategies, Obtain Sales Reports and Manage Stock of Products. The security constraint Restrict only to Authorized Personnel is imposed to the system.

2.2.1.3 Online Store

The actor Online Store depends on the system to fulfill the goals: Manage Products, Keep Inventory and Process Payment. The Security constraints Keep Customer’s Info, Secure transaction and Use strong database are imposed to the system.
2.2.1.4 Customer

The actor Customer depends on the system to fulfill the goals: View Order Status, Create Account, Browse Products and Place Order. The security constraints Keep Customer’s Info Private and Secure transaction are imposed to the system.
2.2.2 System’s Security Enhanced Goals Diagrams

Figure 14 shows the different goals and tasks that the system is going to perform in order to help the different actors who depend on the system to achieve their goals.

Figure 14. System’s Security Enhanced Goals Diagrams
The following table describes the different goals that depend on the system, the dependers and the tasks that the system is going to perform to help the dependers achieve their goals.

**Table 2. Actor – Goal – Task**

<table>
<thead>
<tr>
<th>Actor</th>
<th>Goal</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Store Admin</strong></td>
<td>Manage Stock of Products</td>
<td>Provide stock of products</td>
</tr>
<tr>
<td></td>
<td>Obtain Sales Reports</td>
<td>Generate report</td>
</tr>
<tr>
<td></td>
<td>Create Marketing Strategies</td>
<td>Provide information</td>
</tr>
<tr>
<td><strong>Employee</strong></td>
<td>Retrieve Client Info</td>
<td>Process request</td>
</tr>
<tr>
<td></td>
<td>Troubleshoot Customer's problems</td>
<td>Provide order's info</td>
</tr>
<tr>
<td></td>
<td>Process orders</td>
<td>Process request</td>
</tr>
<tr>
<td></td>
<td>Retrieve Orders</td>
<td>Provide order's info</td>
</tr>
<tr>
<td></td>
<td>Process shipping</td>
<td>Process request</td>
</tr>
<tr>
<td><strong>Customer</strong></td>
<td>View Order Status</td>
<td>Provide information</td>
</tr>
<tr>
<td></td>
<td>Create Account</td>
<td>Process request</td>
</tr>
<tr>
<td></td>
<td>Browse Products</td>
<td>Provide User Friendly Catalogue</td>
</tr>
<tr>
<td></td>
<td>Place Order</td>
<td>Process request</td>
</tr>
<tr>
<td><strong>Online Store</strong></td>
<td>Process Payment</td>
<td>Process request</td>
</tr>
<tr>
<td></td>
<td>Manage Products</td>
<td>Provide information, provide user friendly catalogue</td>
</tr>
</tbody>
</table>
The security constraint *Secure Transaction* is satisfied by the secure goal *Provide secure transaction* by performing the secure task *Use secure connection*.

The security constraints *Keep Customer’s info private* and *Restrict only to Authorized Personnel* is satisfied by the secure goal *Authenticate access* by performing the secure task *Ask for username and password*.

The security constraint *Accurate Inventory* is satisfied by the secure goal *System’s integrity* by performing the secure task *Use a strong database*.

### 2.3 Architectural Design

After having analyzed the early and late requirements, the system’s global architecture is defined and shown in Figure 15.
Figure 15. Architectural Design
3 THREAT MODELING

3.1 Identify the known threats to the system

After having identified that the main assets of this e-commerce site are the websites, its database, and reputation, the following threats can be identified [21,24]:

- Hackers trying to gain access to the system and steal the customer’s information
- Open back doors whereby attackers can access the system’s database
- The use of weak passwords to manage the customer’s accounts
- The use of default passwords to access the system’s configuration
- People trying to use false identities (imposter’s) to make purchases
- Unauthorized users whom gain access to the system’s database
- Anonymous users trying to make fraudulent transactions
- Poor programming techniques that allow attackers to gain access to the system and modify features
- System failure
- Attackers using data packet sniffing tools to capture confidential information
- Attackers performing IP Spoofing
- Attackers using port scanning tools to look for ways to access the system
- Attacks by means of:
  - Viruses
  - Trojan horses
  - Logical bombs
  - Syn Flooding
  - Denial of Service
• Ping of death
• Data injection
• Buffer Overflow

3.2 Rank and group the threats in order by decreasing risk

In order to rank and group the threats, the STRIDE model is used:

1. Spoofing Identity:
   o Hackers trying to gain access to the system
   o False identities (imposters)
   o Unauthorized users who have access

2. Tampering with data:
   o Poor programming techniques
   o Data injection

3. Repudiation:
   o Anonymous users

4. Information disclosure:
   o Open back doors
   o Data packet sniffing
   o IP Spoofing
   o Port Scanning

5. Denial of service:
   o Syn Flooding
   o Denial of Service
   o Ping of death
6. Elevation of privileges:
   - Weak passwords
   - Default passwords

3.3 Determine how to respond to the threats and identify techniques that mitigate the threats

Table 3 indicates how the different threats were grouped according to the STRIDE model and the corresponding technique to mitigate these risks.
Table 3. Threat and Mitigation Technique

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Spoofing Identity:</td>
<td></td>
</tr>
<tr>
<td>o Hackers trying to gain access to the system</td>
<td>Use a strong Authentication technique</td>
</tr>
<tr>
<td>o False identities (imposters)</td>
<td>Use a strong Authentication technique</td>
</tr>
<tr>
<td>o Unauthorized users who have access</td>
<td>Use a strong Authentication technique</td>
</tr>
<tr>
<td>2. Tampering with data:</td>
<td></td>
</tr>
<tr>
<td>o Poor programming techniques</td>
<td>Use expert programmers</td>
</tr>
<tr>
<td>o Data injection</td>
<td>Use stored procedures, use admin login only when needed</td>
</tr>
<tr>
<td>3. Repudiation:</td>
<td></td>
</tr>
<tr>
<td>o Anonymous users</td>
<td>Use a strong Authentication technique</td>
</tr>
<tr>
<td>4. Information disclosure:</td>
<td></td>
</tr>
<tr>
<td>o Open back doors</td>
<td>Use encryption</td>
</tr>
<tr>
<td>o Data packet sniffing</td>
<td>Use encryption</td>
</tr>
<tr>
<td>o IP Spoofing</td>
<td>Use port security commands</td>
</tr>
<tr>
<td>o Port Scanning</td>
<td>Use secure network configuration</td>
</tr>
<tr>
<td>5. Denial of service:</td>
<td></td>
</tr>
<tr>
<td>o Syn Flooding</td>
<td>Harden network configuration</td>
</tr>
<tr>
<td>o Denial of Service</td>
<td>Harden network configuration</td>
</tr>
<tr>
<td>o Ping of death</td>
<td>Harden network configuration</td>
</tr>
<tr>
<td>o Buffer Overflow</td>
<td>Harden network configuration</td>
</tr>
<tr>
<td>o Overall system failure</td>
<td>Harden system's configuration</td>
</tr>
<tr>
<td>6. Elevation of privileges:</td>
<td></td>
</tr>
<tr>
<td>o Weak passwords</td>
<td>Use a strong Authentication technique</td>
</tr>
<tr>
<td>o Default passwords</td>
<td>Change default passwords</td>
</tr>
</tbody>
</table>
4 INTEGRATION OF THREAT MODELING INTO THE
SECURE TROPOS METHODOLOGY

After having identified the different threats that the system has to face through the Threat Modeling process, the system is again modeled following the Secure Tropos Agent-Oriented Methodology.

In the Early Requirements phase, it was determined that a new agent, goals and security constraints have to be introduced. The new agent is the system developer. The system developer is the person who is going to develop the system. The system developer does not have any goals to fulfill but it is the dependee for some of the actor Online Store’s goals. Security constraints are also imposed to fulfill the goals (Figure 16).
Figure 17 depicts the tasks that the actor *System Developer* has to perform in order to fulfill the actor *Online Store*’s goals.
The actor *System Developer* is going to help the actor *Online Store* perform the goal *Harden Access* by performing the task *Harden Server* taking into account the security constraint *Use strong authentication method*. The goal *Harden data integrity* is going to be fulfilled by performing the task *Harden Database*. The security constraint imposed by the actor *Online Store* is *Only Admin can access DB*. The goal *Protect system from attackers* is going to be fulfilled by performing the following tasks: *Check Network Configuration*, *Check for open back doors* and *Harden server*.

In the Late Requirements analysis, the actor *System Developer* is introduced since the system itself has goals to fulfill that depend on the *System Developer*. These goals are: *Prevent*
Attacks, Prevent System Failure, Protect Data Integrity, Enforce Strong Password, Deny Access to attackers, Close Open Doors.

The system’s goal diagram does not need any changes since the actor System developer does not have any goals that depend on the system. In the architectural design, the system remains the same as well (Figure 18).

Figure 18. System’s Actor Diagram with Threat Modeling
5 Application of Security Attack Scenarios for Testing the Different Models

5.1 Test Case 1: Data Modification through SQL injection

Precondition: The attacker tries to gain access to the database in order to modify the information in the tables by injecting SQL queries through input fields.

System expected security reaction: The system should not allow the attacker obtain any information that could help him gain information about the database. Every input field should be validated and good programming techniques such as the use of stored procedures should be used.

Discussion: The attacker will try to find if the website has any vulnerable input fields that can provide information about the kind of database that is being used as well as the database configuration and design. Once this vulnerability has been identified, the attacker injects SQL sentences with malicious purposes.

Test case result: In the first model, this kind of threat was not considered; therefore, it is possible for the attacker to find input fields that allow him inject SQL sentences.

In the second model, after performing the threat modeling of the system, this threat was identified. The system developer used programming techniques that do not allow to inject SQL sentences in the forms. Every input field is also validated. Thanks to these techniques, the website is safe from suffering a SQL injection attack that compromises the integrity of the information in the database (Figure 19).
Figure 19. Test Case 1: Data Modification through SQL injection

5.2 Test Case 2: Denial of Service

Precondition: The attacker tries to make the website unavailable by performing a DoS attack.

System expected security reaction: The system should be able to detect that someone is trying to perform a DoS attack. A notification should be sent by the system whenever a DoS attack has been detected.

Discussion: The attacker can maliciously cause the system to become unavailable by performing a DoS Attack. The attacker can use numerous methods, i.e., he can send a number of SYN (synchronize number) requests to the system and cause a SYN flood. The system should be designed and configured in such a way whereby it can detect this kind of attacks. Network
configurations play an important role to mitigate this threat. For example, the existence of firewalls that can help detect and block this kind of threats as well as alert the network manager by sending him a notification.

Test case result: The first model before the threat modeling was integrated, does not consider this kind of attack; resulting in the system becoming vulnerable and easily compromised.

In the second model, this threat was identified and the actor *System Developer* was introduced. One of the main goals of this actor is to configure the system in a way that it is protected from attackers. To achieve this goal, this actor is going to check the network configuration and also make sure that there are not any open back doors through which the attacker can access the system (Figure 20).

![Figure 20. Test Case 2: Denial of Service](image)
5.3 Test Case 3: Fraudulent transaction

Precondition: The attacker will try to make a fraudulent transaction by using stolen login information and by performing a brute force attack to obtain the password.

System expected security reaction: The system should use a secure connection when sending personal information to third parties. The system should be able to recognize when an attacker is trying to log in using a stolen identity. The system must enforce the customer to set a strong password.

Discussion: Attackers have found different ways to make fraudulent transactions. One of them is the use of a stolen identity. It is easy for an attacker to obtain login information if the information that is sent through the web is not sent through a secure connection. Attackers also use brute force attacks to find passwords and gain access to user’s accounts. The system should be able to provide a secure environment to the customer. The system is usually in danger of this kind of attack at the time the customer is going to perform the payment. A way to mitigate this is to use a secure channel to send the information to a third party. The system developer should make sure that the third party that is going to be in charge of processing the payment also provides the highest level of security by demanding certificates that prove its credibility.

Test case result: The first model is designed keeping in mind that the customer’s information should be kept private all the time, therefore a secure channel is always used. As for making sure that the third party provides the same level of security, it is uncertain.

In the second model, one of the goals of the actor System Developer is to make sure that the third party provides a high level of security, wherefore, this threat is mitigated (Figure 21).
Figure 21. Test Case 3: Fraudulent transaction
6 APPLICATION: CASE STUDY

After having analyzed and integrated threat modeling into the Secure Agent Oriented Methodology, a test version of the e-commerce application was developed. This version allows users to browse through a catalogue and add products to a cart. The current application also allows the creation, update and deletion of the different products with the restriction that only the user Administrator is allowed to do these actions.

Figures 22, 23, 24, 25, 26 and 27 show the functionalities that this test application has.

Figure 22. Application: Index
Figure 23. Application: Browsing

Figure 24. Application: Add to Cart
Figure 25. Application: Log On

Figure 26. Application: Shipping Information
Figure 27. Application: End Purchase

An analysis was made to choose the appropriate framework for developing e-commerce applications and the .Net MVC (Model-View-Controller) framework was chosen.

When using the MVC framework [10, 12, 7] methodology, the application is divided into three different components: the models, the views and the controllers.

The models are the part of the application that allows maintaining a state which is linked to a database (Figure 28).
Figure 28. Models

The views are used to display the application’s user interface (Figure 29).

Figure 29. Views

The controllers are used to handle the interaction with the end user. While the views are used to display information, the controller is used to handle what is shown in the views (Figure 30).
One of the main reasons that the .Net MVC framework methodology was chosen is because it helps develop a secure application taking into account the different threats that were found in the threat modeling process and implement the security constraints that were introduced in the modeling stages.

MVC allows the developer to use different syntaxes to indicate which code will be executed only and which code will be executed and the results shown. The use of these different syntaxes helps protect the site against cross-site scripting and HTML injection attacks [6] (Figure 31).
Embedded into MVC, there is an Account Controller class which allows developers to include authentication in their application. MVC also allows the creation of usernames and roles through the ASP.Net website administration tool [6] (Figures 32, 33).
Figure 32. Account Controller

Click a row to select a user and then click **Edit user** to view or change the user’s password or other properties. To assign roles to the selected user, select the appropriate check boxes on the right.

To prevent a user from logging into your application but retain his or her information in your database, set the status to inactive by clearing the check box.

Figure 33. ASP .Net Web Site Administration Tool
By assigning different roles to the users, the developer can restrict the access to certain parts of the application, for example in this case, only the users who have administrative roles are allowed to create, update or delete the products through the website (Figure 34).

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using EcuadorStore.Models;
using EcuadorStore.ViewModels;

namespace EcuadorStore.Controllers
{
    [Authorize(Roles = "Administrator")]
    public class StoreManagerController : Controller
    {
        EcuadorStoreEntities storeDB = new EcuadorStoreEntities();

        // GET: /StoreManager/
        public ActionResult Index()
        {
            var products = storeDB.Products
                           .Include("Category").Include("Provider")
                           .ToList();

            return View(products);
        }
    }
```

Figure 34. [Authorize] Attribute

To ensure that the information transmitted in the webpage uses a secure channel, the developer can make use of the [RequireHttps] syntax (Figure 35).
namespace EcuadorStore.Controllers
{
    [RequireHttps]

    [Authorize]
    public class CheckoutController : Controller
    {
        EcuadorStoreEntities storeDB = new EcuadorStoreEntities();
        const string PromoCode = "FREE";

        public ActionResult Index()
7 ANALYSIS AND COMPARISON

After having identified the secure capabilities of each of the models by using security attack scenarios, it has been determined that integrating threat modeling in the Secure Agent-Oriented Software Development methodology increases the level of security of the resulting application.

Threat modeling helps identify the threats, risks and vulnerabilities of the system. By designing a model after the threats have been identified, the different actors, goals, tasks and security constraints are defined with the mere purpose of mitigating these risks.

For instance, in the first model, the possibility of an injection attack was not considered; therefore, the application resulting from this model was vulnerable to this kind of attacks. In the second model, where threat modeling was integrated, this kind of attacks were identified as a threat, and security constraints, actors, goals and tasks were introduced to mitigate this risk. At the time of choosing the framework for developing the application, this kind of threats were considered, therefore, a framework that could help prevent this kind of injection attacks was used. The .Net MVC (Model –View – Controller) framework use two different syntaxes to indicate which code will be executed only and which code will be executed and the results shown. The use of these syntaxes helps prevent injection attacks [6]. By separating which code will be executed, MVC is able to detect whether the input from the user contains characters that are used to execute procedures. This feature was tested in the application. The text “<script>alert(“This is an attack”)</script>” was introduced in an input field (Figure 36) and a warning message indicating that an attempt to perform this attack was detected.
Figure 36. Java Script Injection Attack

A potentially dangerous Request.Form value was detected from the client (Product.ProductID="<script>alert("This ...")."

Description: Request Validation has detected a potentially dangerous client input value, and processing of the request has been aborted. This value may indicate an attempt to compromise the security of your application, such as a cross-site scripting attack. To allow pages to override application request validation settings, set the requestValidationMode attribute in the httpRuntime configuration section to requestValidationMode="Off". Example: <httpRuntime requestValidationMode="Off"/> After setting this value, you can then disable request validation by setting validateRequest="false" in the Page directive or in the <pages> configuration section. However, it is strongly recommended that your application explicitly check all inputs in this case. For more information, see http://go.microsoft.com/fwlink/?LinkId=152313.

Exception Details: System.Web.HttpRequestValidationException: A potentially dangerous Request.Form value was detected from the client (Product.ProductID="<script>alert("This ...")."

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:

[HttpRequestValidationException (0x80004005): A potentially dangerous Request.Form value was detected from System.Web.HttpRequest.Validate( String value, String collectionKey, RequestValidationSource reqSource)]

Figure 37. Injection Attack Detection
Another threat that was not considered while designing the first model is that an attacker may try to make the system unavailable through a Denial of Service attack. In the second model, the possibility of this kind of attack was considered and a new actor was introduced. This new actor’s main goal is to ensure that the network configuration is correct by using a robust firewall configuration and closing any open back doors. With these tasks, the risk of being under a Denial of Service attack is (if not eliminated), identified and mitigated.

While designing the first model, one main security goal was to protect the customers’ information. Therefore, a security constraint was introduced. This security constraint only considered attackers trying to steal the customers’ information, but it did not consider that an attacker may try to use brute force attacks to obtain a password or to use already stolen information in order to complete their transaction. In the second model, these last risks were considered and new actors, security constraints and tasks to help mitigate these risks were introduced. In this case, it is also shown that during the development of the application itself, measures can be taken to prevent people from stealing information.

When developing the application, the use of the [RequireHTTPS] attribute that the MVC framework provides, helps ensure that the information entered by the user is sent through a secure channel. The use of this secure protocol, which encrypts the information with two keys [5], prevents attackers from intercepting the information that is transmitted through the web. It also helps users identify if the website where they are entering their information is the real website and not one of the fake websites that attackers use to get user’s information (phishing attack). The use of the ASP .Net Web Site Administration tool to administrate the different accounts also helps increase the level of security of the application. For example, when creating a new account, a function enforces users to create a password that contains at least 6 characters.
The capability to allow the assignation of roles to different users helps the developer restrict the access to certain parts of the application.

By integrating threat modeling in the development process, different risks are mitigated in the early stages of the development as well as in the late stages, being it is the development of the application itself. The developer keeps in mind the different threats that need to be mitigated when choosing what framework to use in the development.

These comparisons have helped to have a clearer picture of how integrating threat modeling in the Secure Agent-Oriented Software Development methodology Secure Tropos, increases the security level of the resulting application. This is achieved by adding actors, security constraints and tasks that help mitigate the different risks and vulnerabilities that put in danger the security of a system.
8 RECOMENDATIONS

Nowadays, Information Security has become a vital characteristic of software applications. The main goal of a developer should be to design and deploy a secure application that not only protects the information, but is available, and provides data integrity, the three pillars of information security.

The development of the application using the .Net MVC framework allows for creating secure web-based applications thanks to the embedded characteristics it has. Similar frameworks should be used when developing these kinds of applications; they are already designed to provide security. This is a characteristic that developers should take advantage of.

Identifying the different threats risks and vulnerabilities that an application may have to face is a very useful way of improving the security features of a system. Threat modeling should become a key process when designing and developing a software application. Throughout the development of this thesis, it has been proven that by integrating threat modeling in the development process, more risks and vulnerabilities were identified and the main purpose of the model was to mitigate these risks. As a result, a more secure application was modeled and developed.
9 CONCLUSIONS AND FUTURE WORK

The Secure Tropos methodology integrates “security extensions” to the agent oriented software development process. These security extensions give a level of security to a system. By integrating threat modeling to this methodology, the level of security can be improved since the security extensions, actors, goals, and tasks will be defined based on the threats that could compromise the security of a system. After having applied different security attack scenarios, it was proved that by integrating threat modeling in the development process, the level of security of the modeled application increased. The different actors, goals, tasks, and security constraints that were introduced based on the threat modeling, help mitigate the different risks and vulnerabilities that were found.

Integrating threat modeling in the development process is a software engineering approach that helps mitigate security problems within the development process. Other approaches and software development methodologies such as SecureUML [2], UMLsec [8, 17], Security Engineering Model [9], and others, are also used to mitigate security problems. The proposed approach is to complement not replace existing IA techniques.

Secure Tropos is a relatively new software development methodology, therefore other processes can be integrated in order to ensure that the resulting application has all the security features that customer’s demand. Threat modeling has helped increase the security level of the modeled application in the early stages, further research needs to be done in order to ensure that the resulting system will be flexible enough in order to adapt to new threats that appear every day.
REFERENCES


