Integrating Threat Modeling in Secure Agent-Oriented Software Development

Diana M. Rojas *drojas1@islander.tamucc.edu*
Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX 78412 USA

Ahmed M. Mahdy *ahmed.mahdy@.tamucc.edu*
Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX 78412 USA

Abstract

The main objective of this paper 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 [2]. Threat modeling is used to identify, document, and mitigate security risks [6], 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 [13] are applied to the models to discuss how the security level of the system has been impacted. Security attack scenarios have been used to test different enhancements made to the Secure Tropos methodology and the Tropos methodology itself [13] [11]. 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.

Keywords: Secure Tropos, Threat Modeling, Security Attack Scenarios.

1. INTRODUCTION

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. 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 [2].

Threat modeling is the process of identifying, documenting and mitigating security risks [6] 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]. 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" [19]. 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 paper, the Secure Tropos methodology will be used.

1.2 Secure Tropos

This methodology is an agent oriented methodology based on the Tropos methodology. 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 [2]. 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, late requirements, architectural design, and detailed design.

This methodology uses the i* modeling framework [2]. 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.

In addition to these concepts, 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 dependers.

1.3 Threat Modeling

Threat modeling is a process that should be done in the design process [25] in order to identify, document and mitigate security risks [6]. 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 [25].
The procedure for creating a threat model is [19]:

1. Identify the known threats to the system.
2. Rank and group the threats in order by decreasing risk using the STRIDE model.
3. Determine how to respond to the threats.

4. Identify techniques that mitigate the threats: different techniques have been specified to mitigate each STRIDE category.
5. Choose the appropriate technologies from the identified techniques.

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: possible attack, possible attacker, resources attacked, and agents related to the attack [13]. 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].

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 costumers 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 costumer 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.

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 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.

Security Enhanced Actor Diagram
The following security enhanced actor diagram depicts each actor with their goals, soft goals, security constraints and the dependencies that they have with each other.
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.

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 (depender) depends on the actor *Online Store* (dependee) and to fulfill the goal *Obtain the products* it depends on the actor *Provider* (dependee).

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 (depender) 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.

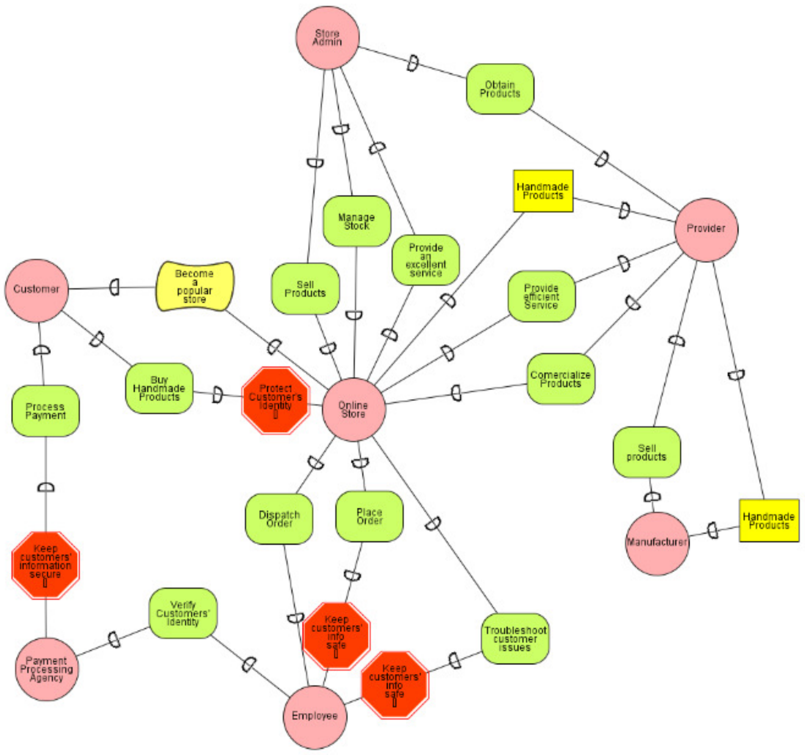

FIGURE 1: Security Enhanced Actor Diagram.

Provider: The actor *Provider*'s main goal is *Commercialize the products*. For this, actor *Provider* (depender) depends on the actor *Online Store* (dependee). This actor uses the resource *Handmade products* for which he depends on the actor *Manufacturer's* (dependee).
Employee: The actor *Employee*'s main goal is *verify Customer's identity*. For this, it (depender) depends on the actor *Payment Processing Agency* (dependee).
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.

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.
The following diagram shows each actor, the goals that depend on them, and the tasks that they have to perform in order to achieve the aforementioned goals.

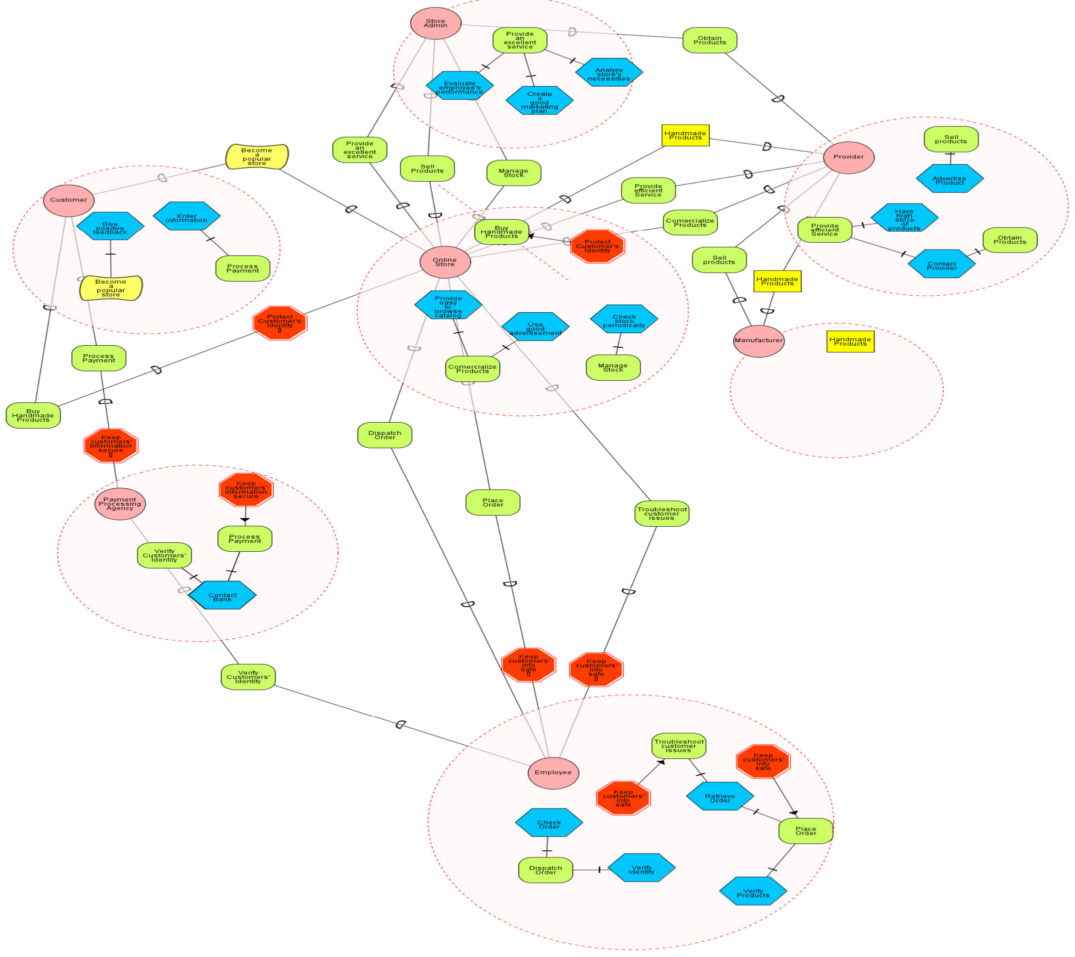

FIGURE 2: Security Enhanced Goal Diagram.

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*.
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 shown in this diagram.
Manufacturer: The actor *Manufacturer* is only involved in providing the resource *Handmade Products*.
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*.

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.*
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*.
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, Crete a good marketing plan* and *Analyze store's necessities*.

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 to be 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.

System's Security Enhanced Actor Diagram

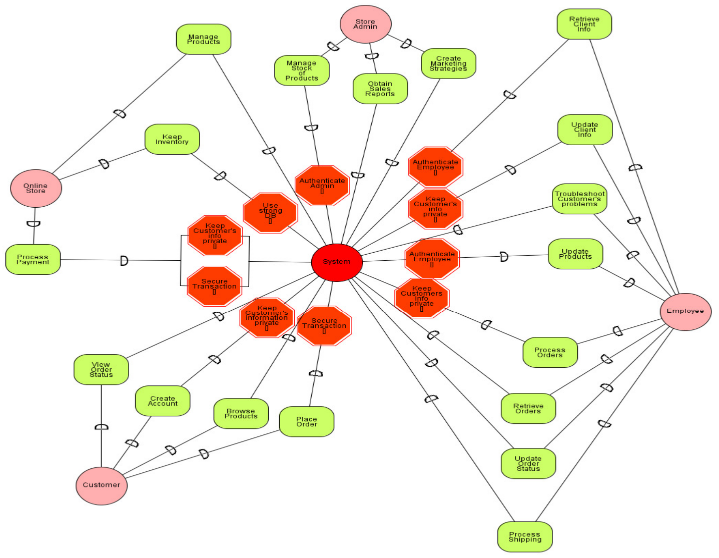

FIGURE 3: System's Security Enhanced Actor Diagram.

The following diagram 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.
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.
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.

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.
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.

System's Security Enhanced Goals Diagrams
The following diagram 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.

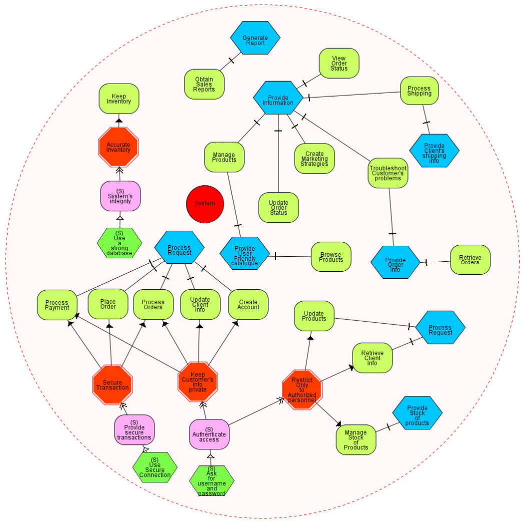

FIGURE 4: System's Security Enhanced Goals Diagrams.

Architectural Design
After having analyzed the early and late requirements, the system's global architecture is defined and the following diagram is generated.

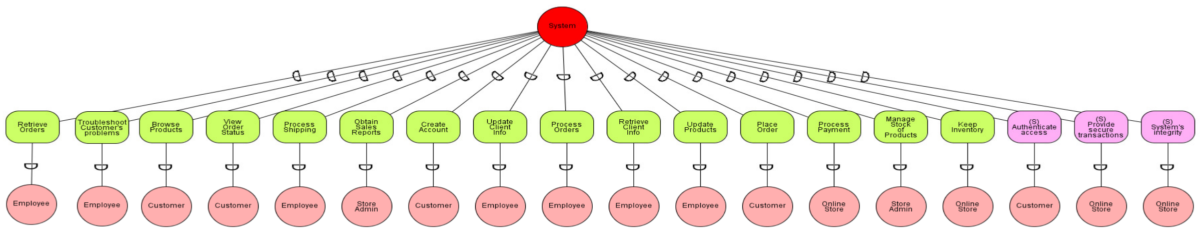

FIGURE 5: Architectural Design.

3. THREAT MODELING

Identify the known threats to the system, rank and group the threats in order by decreasing risk and determine how to respond to the threats and identify techniques that mitigate the threats

Threat	**Mitigation Technique**
1. Spoofing Identity:	
o Hackers trying to gain access to the system	Use a strong Authentication technique
o False identities (imposters)	Use a strong Authentication technique
o Unauthorized users who have access	Use a strong Authentication technique
2. Tampering with data:	
o Poor programming techniques	Use expert programmers
o Data injection	Use stored procedures, use admin login
3. Repudiation:	
o Anonymous users	Use a strong Authentication technique
4. Information disclosure:	
o Open back doors	Use encryption
o Data packet sniffing	Use encryption
o IP Spoofing	Use port security commands
o Port Scanning	Use secure network configuration
5. Denial of service:	
o Flooding	Harden network configuration
o Denial of Service	Harden network configuration
o Ping of death	Harden network configuration
o Buffer Overflow	Harden network configuration
o Overall system failure	Harden system's configuration
6. Elevation of privileges:	
o Weak passwords	Use a strong Authentication technique
o Default passwords	Change default passwords

TABLE 1: Threat and Mitigation Technique.

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. The following goal's diagram 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.

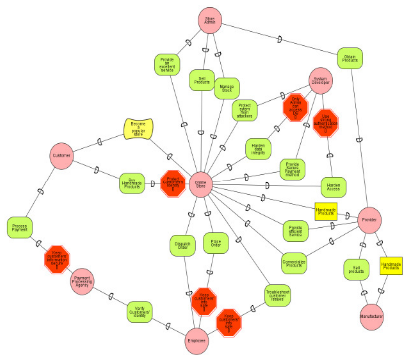

FIGURE 7: Security Enhanced Goal Diagram with Threat Modeling.

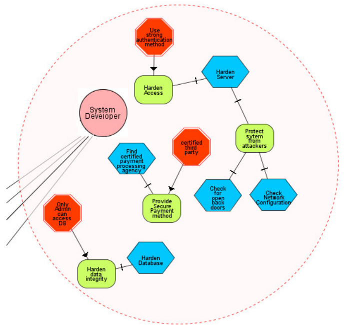
Use strong authentication method
Harden Access
Harden Server
System Developer
Protect sytem from attackers
Find certified payment processing agency
certified third party
Provide Secure Payment method
Check for open back doors
Check Network Configuration
Only Admin can access DB
Harden data integrity
Harden Database

FIGURE 8: System Developer Goal Diagram.

5. APPLICATION OF SECURITY ATTACK SCENARIOS FOR TESTING THE DIFFERENT MODELS

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 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 doing the threat modeling of the system, this threat was identified. The system developer used programming techniques that don't allow inject SQL sentences in the forms. Every input field is also validated. Thanks to these techniques, the website is safe from suffering from a SQL injection attack that compromises the integrity of the information in the database.

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 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 send a notification.

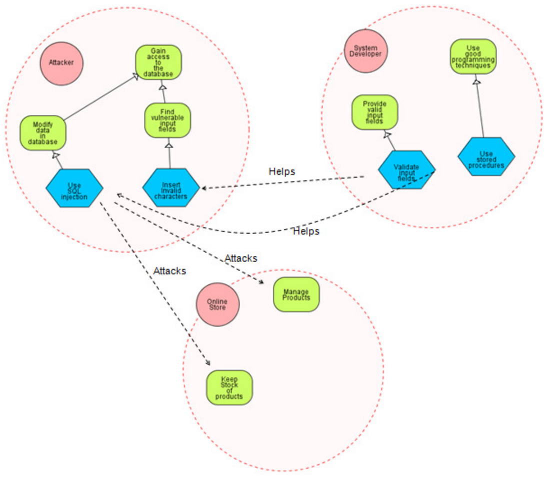

FIGURE 9: Test Case 1: Data Modification through SQL injection.

Test case result: The first model does not consider this kind of attack; resolving in the system becoming vulnerable where it can easily be 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.

Test Case 3: Fraudulent Transaction
Precondition: The attacker will try to make a fraudulent transaction by using stolen login information and by performing a force brute 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 force brute attacks to find passwords and gain access to 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 who is going to be in charge of processing the payment also provides the highest level of security.
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.

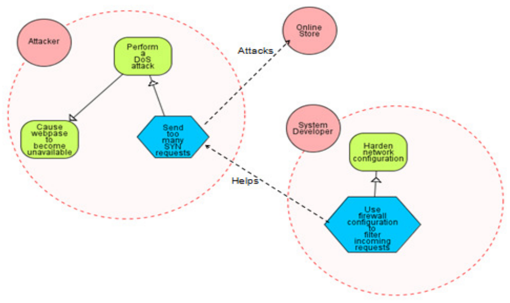
Attacker
Perform a DoS attack
Cause webpase to become unavailable
Send too many SYN requests
Attacks
Online Store
System Developer
Harden network configuration
Use firewall configuration to filter incoming requests
Helps

FIGURE 10: Test Case 2: Denial of service.

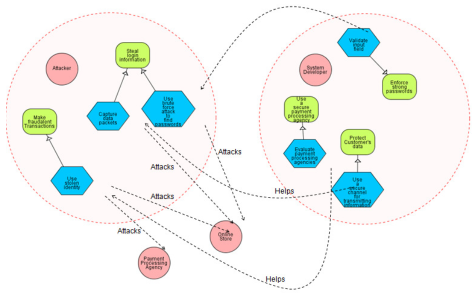
Attacker
Steal login information
Make fraudulent Transactions
Capture data packets
Use brute force attack to find passwords
Use stolen identity
Attacks
Attacks
Attacks
Attacks
Online Store
Payment Processing Agency
System Developer
Validate input field
Enforce strong passwords
Use a secure payment processing agency
Evaluate payment processing agencies
Protect Customer's data
Use a secure channel for transmitting information
Helps
Helps

FIGURE 11: Test Case 3: Fraudulent transaction.

Case Study

After having analyzed and integrated threat modeling into the Secure Agent Oriented Methodology, a sample 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 sample 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.

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

When using the MVC framework methodology, the application is divided into three different components: the models, the views and the controllers [23].

The models are the part of the application that allows maintaining a state which is linked to a database [23]. The views are used to display the application's user interface [23]. 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 [23].

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 [4]. Embedded into MVC, there is an Account Controller class which allows developers include authentication in their application. MVC also allows the creation of usernames and roles through the ASP.Net website administration tool [4]. 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. To ensure that the information transmitted in the webpage uses a secure channel, the developer can make use of the [RequireHttps] syntax.

6. 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 a SQL injection attack was not considered, therefore, the application resulting from this model was vulnerable to this kind of attack. In the second model, where threat modeling was integrated, this kind of attack was 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 threat was considered, therefore, a framework that could help prevent this kind of 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 [4].
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 force brute 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.

7. CONCULSION

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.
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.

8. Acknowledgment

This work has been supported in part by NSF awards CNS-1004902 and CNS-0708573 and a Texas Research Development Fund grant.

9. REFERENCES

[1] D. Basin, M. Clavel, J. Doser and M. Egea, "Automated Analysis of Security-Design Models," Information and Software Technology, vol. 51, no. 5, pp. 815-831, May. 2009.

[2] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology," Autonomous Agents and Multi-Agent Systems, vol. 8, no. 3, pp. 203-236, May. 2004.

[3] B. Mains. (2010, September) Introduction to ASP .NET MVC 2.0. [Online]. Available: http://dotnetslackers.com/articles/aspnet/Introduction-to-ASP-NET-MVC-2-0.aspx.

[4] J. Galloway, "ASP .NET MVC Music Store Tutorial," Microsoft, Oct. 2010.

[5] Comodo. (2010, October) Instant SSL. [Online]. Available: http://www.instantssl.com/ssl-certificate-products/https.html.

[6] S. Ambler. (2010, March) Introduction to Security Threat Modeling. [Online]. Available: http://www.agilemodeling.com/artifacts/securityThreatModel.htm.

[7] J. Jurjens, "Foundations for Designing Secure Architectures," Electronic Notes in Theoretical Computer Science, vol. 142, pp. 31-46, Jan. 2006.

[8] Y. Lee, J. Lee and Z. Lee, "Integrating Software Lifecycle Process Standards with Security Engineering," Computers & Security, vol. 21, no. 4, pp. 345-355, Aug. 2002.

[9] R. Matulevi, N. Mayer, H. Mouratidis, E. Dubois, P. Heymans, and N. Genon, "Adapting Secure Tropos for Security Risk Management in the Early Phases of Information Systems Development," in Proc. 20th International Conf. on Advanced Information Systems Engineering, 2008, pp. 541-555.

[10] Microsoft ASP .NET. (2010, September) ASP.NET MVC Overview. [Online]. Available: http://www.asp.net/mvc/tutorials/asp-net-mvc-overview-cs.

[11] H. Mouratidis, P. Giorgini, and G. Manson, "Using Security Attack Scenarios to Analyze Security During Information Systems Design," in Proc. 6th International Conference on Enterprise Information Systems, 2004, pp. 10-17.

[12] H. Mouratidis, P. Giorgini, and G. Manson, "When Security Meets Software Engineering: A Case of Modeling Secure Information Systems," Information. Systems, vol. 30, no. 8, pp. 609-629, Dec. 2005.

[13] H. Mouratidis and P. Giorgini, "Enhancing Secure Tropos to Effectively Deal with Security Requirements in the Development of Multiagent Systems," Safety and Security in Multiagent Systems: Research Results From 2004-2006, M. Barley, H. Mouratidis, A. Unruh, D. Spears, P. Scerri, and F. Massacci, Eds. Lecture Notes In Artificial Intelligence, vol. 4324, Springer-Verlag, Berlin, Heidelberg, pp. 8-26.

[14] H. Mouratidis and P. Giorgini, "Secure Tropos: A Security-Oriented Extension of the Tropos Methodology," International Journal of Software Engineering and Knowledge Engineering.

[15] H. Mouratidis, J. Jurjens, and J. Fox, "Towards a Comprehensive Framework for Secure Systems Development," Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 4001, Springer-Verlag, Berlin, Heidelberg, pp. 48-62.

[16] MSDN Library. (2010, April) The STRIDE Threat Model. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee823878(CS.20).aspx.

[17] MSDN Library. (2010, April) Security Design by Threat Modeling. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee810542(v=CS.20).aspx.

[18] MSDN Library. (2010, April) Threat Model Analysis. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa561499(BTS.20).aspx.

[19] MSDN Library. (2010, April) Identifying Techniques that Mitigate Threats. [Online]. Available: http://msdn.microsoft.com/en-US/library/ee798428(v=CS.20).aspx.

[20] J. Mylopoulos and J. Castro, "Tropos: A Framework for Requirements-Driven Software Development," Information Systems Engineering: State of the Art and Research Themes, J. Brinkkemper, and A. Solvberg, Eds. Lecture Notes In Computer Science, Springer-Verlag, Berlin, Heidelberg, 2000.

[21] New York State office of Cyber Security and Critical Infrastructure Coordination (2009, October) [Online]. Available: http://www.cscic.state.ny.us/lib/glossary/.

[22] R. Peteanu. (2010, October) Best Practices for Secure Development. [Online]. Available: http://www.arcert.gov.ar/webs/textos/best_prac_for_sec_dev4.pdf.

[23] S. Guthrie. (2010, October) ASP.NET MVC Framework. [Online]. Available: http://weblogs.asp.net/scottgu/archive/2007/10/14/asp-net-mvc-framework.aspx.

[24] Shirvani, A. "Workable attacks against E-commerce," 1st e-Commerce Security Conference Ramiran.Co, Tehran, Iran. 2008.

[25] S. Burns. (2010, March) Threat Modeling: A Process to Ensure Application. [Online]. Available: http://www.sans.org/reading_room/whitepapers/securecode/threat-modeling-process-ensure-application-security_1646.

[26] D. Xu and K. Nygard, "Threat-Driven Modeling and Verification of Secure Software Using Aspect-Oriented Petri Nets," IEEE Transactions on Software Engineering Archive, vol. 32, no. 4, pp. 265-278, Apr. 2006.