ISSN No. 0976-5697

Volume 8, No. 2, March – April 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Integration of Threat Modeling with Attack Pattern and Misuse Case for Effective Security Requirement Elicitation

Md Tarique Jamal Ansari
Department of Information Technology
Babasaheb Bhimrao Ambedkar University
Lucknow, India

Dhirendra Pandey
Department of Information Technology
Babasaheb Bhimrao Ambedkar University
Lucknow, India

Abstract: Today's security is becoming a brainstorming issue due to inventive attacks. To elicit effective security requirement to the system, software developers need to think like an attacker. This paper considers three effective security requirement elicitation techniques, Threat modelling, Misuse case and Attack pattern. Threat Modeling is a technique to prevent the system from any undesired event by modeling all the information which has the potential to harm the system. It is a process for eliciting security requirement by identifying harmful threats to the system. Misuse case represents negative use cases to model threats and mis-actors to represent attackers. Misuse cases are capable of modeling threat and risk analysis process. Attack Pattern works as a method to identify the attacker's perspective. Specifically, Threat modelling, Attack pattern and Misuse case are compared on the basis of some parameters. The comparative analysis provides some merits and demerits of these techniques. This paper investigates how misuse cases enhance the performance of threat modeling. This paper also describes an effective way for security requirement elicitation by integrating threat modeling with attack pattern and misuse cases.

Keywords: Threat Modeling; Misuse Case; Attack Patterns; Secure software development; Security Requirement Elicitation

I. INTRODUCTION

In this era, human life is directly and indirectly affected by various software applications e.g. the medical diagnosis machines are available for complete body checkup and at the same time number of space satellites are launching for communication, educational development, whether forecasting etc. The accuracy, privacy, and security are more important for such kind of software application because human life is sometimes entirely dependent on the software applications. There are several reports available today which focused on the importance of software security by discovering threats to the system.

Symantec discovered more than 430 million new unique pieces of malware in 2015, up 36 percent from the year before[1]. Further various software threats are responsible for financial loss as well as dependability and integrity of the software development. Due to the critical threats on software application such as SQL injection, cross-site scripting (XSS), denial of service (DoS), buffer overflow etc, the functional requirements and non-functional requirements (NFRs) both are major concern for secure software development and it must be considered by the customers, users, and vendors of software applications at the early stage of software development life cycle.

The goal of this research is to design a framework by combining different security requirement elicitation technique to get an effective technique for elicitation of security requirements early in the software development process. Combining three widely used techniques and generalizing their steps is simple and elicits more effectively security requirements. The proposed technique can help not only to summary the viewpoint of different security requirement elicitation techniques but also to make choice in their selection of a particular situation. As a result security engineers can use this improved threat modeling technique to eliminate the flaws of security requirements elicitation techniques in terms of a given condition.

[1]https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

II. LITRATURE REVIEW

Designing secure and the quality computer system is not an easy task today. Hackers break the confidentiality and integrity of the system and in response, software suppliers started providing security as a mandatory feature for their product [1]. Requirement engineering for software application, business system, and data center generally involves its functional requirement. Most of the requirement engineer doesn't consider the security requirement which is the nonfunctional requirement [2].

The need for security requirement exists because the misusers create computer viruses which act as real threats to the system. Integration of use case with misuse to model and analyze the system during the design phase can be enhanced the security by mitigating threats [3]. Several authors proposed techniques for eliciting security requirement at the early phases of software development lifecycle.

Mamadou H. Diallo et al. [9] have proposed a comparative study of three techniques: The Common Criteria, Misuse Cases, and Attack Trees to specifying security requirements. They analyzed that each of these techniques worked well with some strength and weakness. The Common Criteria are difficult to understand and use, but are easy to analyze. Misuse Cases are easy to understand and use, but produces output that is not easy to read. In contrast, Attack Trees produce clear output but are difficult to analyze. They supported to combine the use of attack trees and misuse case. Attack patterns are very much similar to attack trees [10].

Suvda Myagmar et al. [1] have investigated that threat modeling can be used as a basis for security requirement

specification. Guttorm Sindre and Andreas L. Opdahl [5,8] have proposed an extension of UML use cases know as Misuse Cases for representing the unwanted functionality of the system. The main goal of their work is to provide a better way for security requirement elicitation. Tatsuya Abe et al. [14] have proposed a technique to model knowledge about the potential threats in the form of patterns by developing the negative scenarios which are used for business process modeling. They tried to transform the normal scenarios into the negative scenarios.

Xiaohong Yuan et al. [15] described a technique for developing abuse cases based on threat modeling and attack patterns. Inger Anne Tøndel et al. [16] have linked the misuse case and attack trees to get high-level view of threats towards a system through misuse case diagram. They also introduced links to security activity descriptions in the form of UML activity graphs to describe mitigating security activities for each identified threat.

III. IMPROVED THREAT MODELING

The integration of Attack pattern, Misuse case and Threat Model is considered as an improved Threat Model. This section contains an overview of three approaches Threat modeling, Misuse case and Attack Pattern. The aim of this section is to explain why these modeling techniques have been selected for effective security requirement elicitation as well as mitigation.

A. Misuse Case

Misuse case is an extension of UML use cases to specify the performance that is not required in the system. These Misuse cases provide support for eliciting security requirements. Guttorm Sindre and Andreas L. Opdahl investigate that Misuse cases are helpful in eliciting security requirement. They have proposed a systematic approach to eliciting security requirements based on use cases [5].

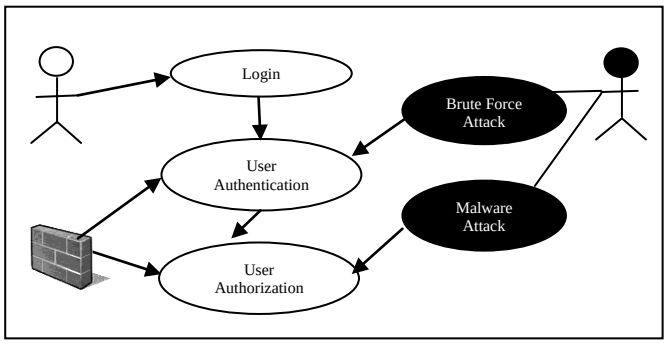


Figure 1 Misuse Case for Firewall Access Control

The approach extends traditional use cases to also cover misuse and is potentially useful for non-functional requirements. In this model, the attacker is considered as misuser that can perform several attacks like DoS , brute force attack, SQL injection etc.

B. Attack Pattern

An attack pattern is based on the concept of design pattern which represents the malicious attack. It is also used for characterizing individual types of attacks [4]. The concept of attack pattern was promoted by Erich Gamma. In his book [11], he discussed the solutions to distinct problems experienced in object-oriented software design and how to package this solution for the large level in the form of design pattern [10]. Attack patterns are quite useful while developing abuse and misuse cases. Attack patterns work as a blueprint for creating any attack to the system [6]. Attack patterns provide the attacker with all the information that he or she requires to achieve a particular objective. Common Attack Pattern Enumeration and Classification (CAPEC) and Common Weakness Enumeration (CWE) are available online to provide threat information in the form of attack pattern. Sean Barnum et al. [10] have documented the basic concept, formation, and usage about the attack patterns as an effective technique in the design, development and deployment of secure software. The following example of attack patterns discovering the access controls that are enforced by a firewall are essential to determining [4]:

Attack Pattern for Access Control Discovery:
Goal: Identify firewall access controls
Precondition: Attacker knows firewall IP address.
Attack:
OR 1. Search for specific default listening ports.
2. Scan ports broadly for any listening ports.
3. Scan ports stealthily for listening ports.
OR 1. Randomize target of scan.
2. Randomize source of scan.
3. Scan without touching target host.
Precondition: Attacker knows firewall access controls.

Michael Gegick and Laurie Williams have designed attack patterns for highlight security vulnerabilities in a software-intensive system design. Their approach is to match the attack patterns to vulnerabilities in the design phase may encourage security efforts to start early by the developers and to become integrated with the software process [12]. Attack patterns provide possible value during all phases of software development [10].

C. Threat Modeling

Threat modeling is not a code reviewing process, but it does complement the security code review process. The formation of threat modeling in the SDLC ensures that applications are being developed with security built-in from the very beginning. Security built in with the documentation produced as part of the threat modeling process can give the reviewer a better understanding of the system. This allows the reviewer to see the entry points of the application and the associated threats with each entry point. A threat model cannot be created by simply brainstorming an adversary's possible objectives. This is not a systematic approach and is likely to leave large portions of the attack space uninvestigated. Threat modeling is a technique for analyzing the security of an application. An attacker only has to find one security flaw to compromise the whole system [7]. Thus, it is important to be systematic during the threat modeling process to ensure that as many possible vulnerabilities and threats are invented by the developers, not the attackers.

Threat analysis should be used at the very first stages of system design. Although the effort required to threat model an existing system is the same as for threat modeling a system during its early design stages, it is harder and costlier to mitigate the threats identified in an existing system due to architectural constraints [1]. Threat modeling allows development teams to understand a system's threat profile by observing the

application software through the eyes of a hacker, and helps to determine the top-level security risks modeled to the system. Threat modeling is considered as an important step in the security requirement engineering paradigm. Its assurances include revealing the highest security risks to a software product, determining how attacks can manifest, helping to find bugs, and controlling penetration testing based on a threat model [17].

IV. COMPARISION BETWEEN APPROACHES

Misuse Cases, Attack Pattern and Threat Modeling provide information about potential threats but each of three techniques has some strength and weakness. Table 1 [9,10] outline the comparison between the three methods based on learnability, usability, clarity of output, solution inclusiveness, and analyzability.

A. Criteria for Evaluation

- **Learnability:** Learnability shows that how long would it take designers to learn and use the techniques? It also shows that the particular technique is easy for the learner or not.
- **Usability:** Usability measures how simple or complex a technique can be used. Once the information is collected, how usable is the process of designing a diagram, tree, or table.
- **Solution Inclusiveness:** Solution Inclusiveness mainly shows that the particular technique is having the complete solution for any problem. In other words, Can these techniques specify any threat and its solution?
- **Clarity of Output:** This principle is concerned with the simplicity associated with reading and using the outputs from the specification technique. It would be more helpful if the techniques provide clear solutions to attacks that are reasonable and usable by software designers.
- **Analyzability:** The analyzability measures how easily a designer can understand and analyze the results provided by the techniques.

B. Comparison of the Techniques

Table 1 shows the comparison between the three approaches Misuse Case, Attack Pattern and Threat Modeling based upon some criteria of evaluation like learnability, usability, solution inclusiveness, clarity of output and analyzability.

Table 1 Comparison between three approaches

	Misuse Case	**Attack Pattern**	**Threat Modeling**
Learnability	Simple to learn based on use case	Difficult to learn for beginners	Simple to learn
Usability	Simple to use based on use cases	Difficult to use for beginners	Simple to use
Solution Inclusiveness	Solution included	Solution included	Solution included
Clarity of Output	May be difficult to learn for large system	Clear output	May be difficult to learn for large system
Analyzability	Easy to analyze	Easy to analyze	Easy to analyze

After analyzing this comparison it is noticed that all these three approaches having some merits and demerits. It is also observed from the above comparison that there is a need for integrating these approaches so that effective elicitation of security requirements can be achieved.

V. THE PROPOSED METHOD

Each of the three methods has some merits and demerits, but they can be joined together to elicit effective security requirements. The proposed method is used for eliciting effective security requirements by integrating attack pattern and misuse cases with Microsoft's threat modeling. The following figure 2 shows this integration.

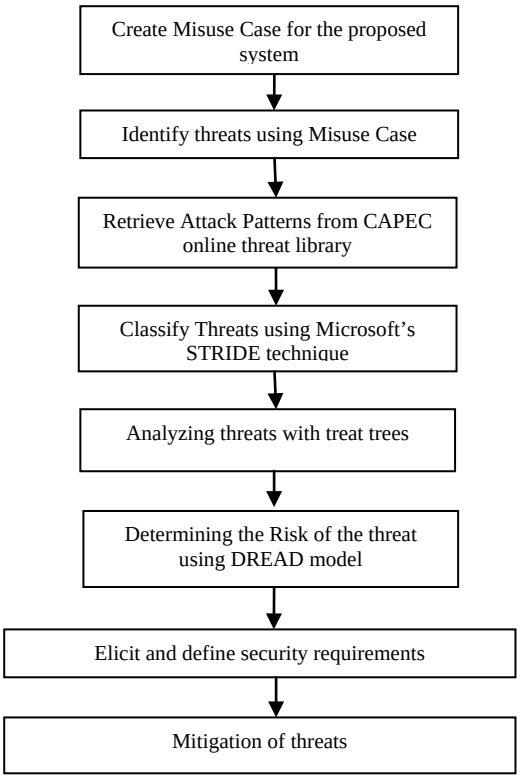


Figure 2 Integration of Threat Modeling with Attack Pattern and Misuse Case

Step 1. Create Misuse Case for the proposed system

The first step of the proposed technique is to create misuse case diagram for the given system. Misuse case diagram represents the unwanted functionality of the system. Misuse case represents all the possible attacks on the system. The easy and mostly used method for creating misuse cases is usually through a process of informed brainstorming. Several theoretical methods require fully specifying a system with rigorous formal models and logics, but such activities are extremely time and resource intensive [6].

Step 2. Identify threats using Misuse Case

The second step of this improved threat modeling technique is to identify the potential threats from the misuse case diagram. The capability to capture threats from misuse cases and then the equivalent mitigating security use cases requires expert knowledge. There are several key areas where results must be made that affect the security of the system like the identity of the misusers, the scope of the misuse cases and the corresponding mitigations [18].

Step 3. Retrieve Attack Patterns from CAPEC online threat library:

After identifying potential threats from misuse case the third step is to retrieve attack patterns from CAPEC (Common Attack Pattern Enumeration and Classification) online library. Several attack patterns can be easily retrieved through keywords from CAPEC.

Step 4. Classify Threats using Microsoft's STRIDE technique:

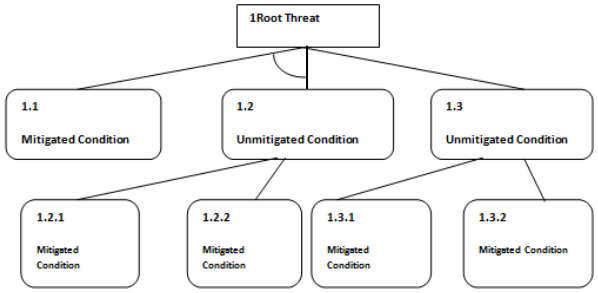
Classify the all potential threats using Microsoft's STRIDE technique after identifying threats in step 2.The Microsoft's STRIDE technique is a classification method for identifying known threats. Known threats can be grouped according to the nature of attack. The STRIDE acronym is made from the first letter of each of the following categories. These categories uniquely identified a particular threat.

Table 2 Microsoft's STRIDE Model

STRIDE	**Description**
Spoofing	Using others credentials to gain access to assets.
Tampering	Changing data to make an attack.
Repudiation	Occurs when a user denies performing an action, but the target of the action has no way to prove.
Information Disclosure	The disclosure of information to a user who does not have permission to see it.
Denial of Service	Reducing the ability of valid users to access resources.
Elevation of Priviledge	When an unprivileged user gains privileged status.

Step 5. Analyzing threats with treat trees:

After classifying the known threat the next step is to analyze and determine the potential threats. Analyzing threats with treat trees: The identified threats must be analyzed to identify the areas where the attacker can easily harm with attacks path. Threat tree is an effective way to analyze threats. Threat trees can be represented in a graphical or textual form within the threat modeling document. A threat tree consists of a root node or threat and child node(s). Each child node represents conditions needed for the adversary to find and identify the threat. Threat trees are used to determine the vulnerabilities associated with a threat. To identify a threat's vulnerabilities, begin at a node without any children and traverse it up to the root threat [13]



1 Root Threat
1.1 (and) (Mitigated) Mitigated Condition
1.2 (and) Unmitigated Condition
1.2.1 (Mitigated) Mitigated Condition
1.2.2 (Mitigated) Mitigated Condition
1.3 Unmitigated Condition
1.3.1 (Mitigated) Mitigated Condition
1.3.2 Unmitigated Condition

Fig 3 A sample threat tree

Step 6. Determining the Risk of the threat using DREAD model:

Another step in analyzing the threats is to determine the risk of the threat and the threat's conditions or child nodes by using the DREAD model. A threat modeling team calculates security risks as an average of numeric values assigned to each of five categories by using the DREAD model [13]. The following table shows the DREAD model functionality.

Table 3 DREAD Model

DREAD	**Description**
Damage potential	The loss if the vulnerability is exploited
Reproducibility	How easy is it to reproduce the threat exploit?
Exploitability	What is needed to exploit this threat?
Affected users	How many users will be affected?
Discoverability	How easy is it to discover this threat?

If the identified threat poses significant risk to the application, the potential threat is rated with high value and needs to be addressed quickly.

Step 7. Elicit and define security requirements:

After analyzing and determining potential threats the next step is to elicit and define security requirements. These security requirements are helpful for the designer while developing the software application. The aim of this step is to map the threats identified for mitigation into security requirements. It accomplishes this by analyzing and determines the threats which obtain from the previous steps. This step completed with elicitation and documentation of security requirements.

Step 8. Mitigation of threats

The last step of improved threat modeling technique is to mitigate all the potential threats. Mitigation of threat is a very important step in software development because if any threat is left unresolved then it will become vulnerability. Vulnerability is weakness for any system. It allows hackers to break the security of any system and illegally access the important assets and data. Mitigation of threats reduces or eliminates the potential threats.

VI. DISCUSSION

Generally software security is not noticed by the developers during early phases of software development life cycle. Therefore, to ensure software security from the beginning the following model has been designed that list all the actions including improved threat modeling to be performed during the life cycle of software development.

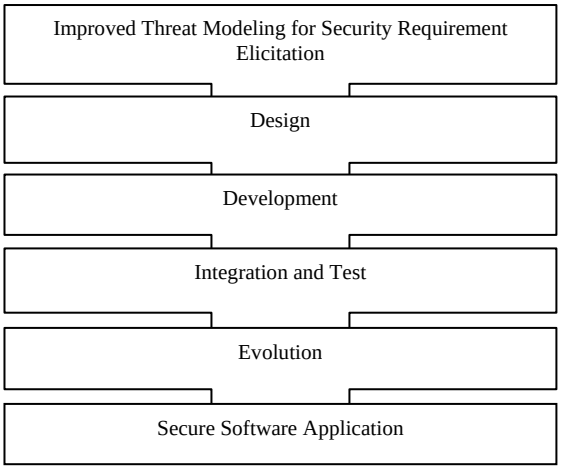


Figure 4 Secure Software Development Model

VII. CONCLUSION

This paper presented the technique to model improved threat modeling by integrating threat model with attack patterns and misuse cases. The main aim of this proposed technique is to elicit the negative scenarios that realize the threats. The improved threat modeling technique is described with a sequence integration of misuse case, attack patterns and threat modeling. MITRE's CAPEC attack patterns are also linked with this model which uniquely identifies each attack pattern. The proposed improved threat modeling approach needs to be further validated by several experiments in order to verify if the technique is useful for the developers to elicit the security requirements in an effective way.

VIII. REFERENCES

[1] Myagmar, Suvda, Adam J. Lee, and William Yurcik. "Threat modeling as a basis for security requirements." Symposium on requirements engineering for information security (SREIS). Vol. 2005. 2005.

[2] Salini, P., and S. Kanmani. "Survey and analysis on security requirements engineering." Computers & Electrical Engineering 38.6 (2012): 1785-1797.

[3] Alexander, Ian. "Misuse cases: Use cases with hostile intent." IEEE software 20.1 (2003): 58-66.

[4] Moore, Andrew P., Robert J. Ellison, and Richard C. Linger. Attack modeling for information security and survivability. No. CMU-SEI-2001-TN-001. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2001.

[5] Sindre, Guttorm, and Andreas L. Opdahl. "Eliciting security requirements with misuse cases." Requirements engineering 10.1 (2005): 34-44.

[6] Hope, Paco, Gary McGraw, and Annie I. Antón. "Misuse and abuse cases: Getting past the positive." IEEE Security & Privacy 2.3 (2004): 90-92.

[7] Schneier, Bruce.: Why cryptography is harder than it looks. EDI FORUM-OAK PARK-. Vol. 10. THE EDI GROUP, LTD., (1997).

[8] Sindre, Guttorm, and Andreas L. Opdahl. "Capturing security requirements through misuse cases." NIK 2001, Norsk Informatikkonferanse 2001, http://www. nik. no/2001 (2001).

[9] Diallo, Mamadou H., et al. "A comparative evaluation of three approaches to specifying security requirements." 12th Working Conference on Requirements Engineering: Foundation for Software Quality, Luxembourg. 2006.

[10] Barnum, Sean, and Amit Sethi. "Attack patterns as a knowledge resource for building secure software." OMG Software Assurance Workshop: Cigital. 2007.

[11] Gamma, Erich. Design patterns: elements of reusable object-oriented software. Pearson Education India, 1995.

[12] Gegick, Michael, and Laurie Williams. "Matching attack patterns to security vulnerabilities in software-intensive system designs." ACM SIGSOFT Software Engineering Notes. Vol. 30. No. 4. ACM, 2005.

[13] Burns, Steven F. "Threat modeling: A process to ensure application security." GIAC Security Essentials Certification (GSEC) Practical Assignment (2005).

[14] Abe, Tatsuya, Shinpei Hayashi, and Motoshi Saeki. "Modeling security threat patterns to derive negative scenarios." 2013 20th Asia-Pacific Software Engineering Conference (APSEC). Vol. 1. IEEE, 2013.

[15] Yuan, Xiaohong, Emmanuel Borkor Nuakoh, and Huiming Yu Imano Williams. "Developing Abuse Cases Based on Threat Modeling and Attack Patterns." Journal of Software 10.4 (2015).

[16] Tøndel, Inger Anne, Jostein Jensen, and Lillian Røstad. "Combining misuse cases with attack trees and security activity models." Availability, Reliability, and Security, 2010. ARES'10 International Conference on. IEEE, 2010

[17] M. Howard and D. LeBlanc. Writing Secure Code. Microsoft Press, 2nd edition, 2002.

[18] Johnstone, Michael N. "Modelling misuse cases as a means of capturing security requirements." (2011).