Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 8, Issue. 1, January 2019, pg.50 – 57

Security Testing of Web Applications Using Threat Modeling: A Systematic Review

Sanjukta Mohanty, Arup Abhinna Acharya, Deepti Bala Mishra, Namita Panda

School of Computer Engineering, KIIT University, Bhubaneswar, India-751024

{mailtorani.sanjukta@gmail.com, aacharyafcs@kiit.ac.in, dbm2980@gmail.com, npandafcs@kiit.ac.in}

ABSTRACT: Due to the increasing heterogeneity and complexity of web, testing the web application for security becomes an essential task. The objective of the security testing is to find the vulnerabilities or weaknesses of software applications. As the web software is highly accessible web application vulnerabilities arguably have greater impact than vulnerabilities in other types of software. So the importance of security increases exponentially to restrict the unauthorized access. To address the security issues like confidentiality, integrity, availability etc. early in the software development life cycle we go for security development models that guides the development process. That's why many more studies on security testing have been conducted and various testing techniques have been developed. Threat modeling is one of the important technique for building a secure software which guides the development process by identifying the possible vulnerabilities and threats at the early stages of SDLC. This research paper focus on analyzing the different existing threat modeling techniques for detecting threats used for security testing. An efficient framework for security testing is also drawn for our future work.

Keywords: Security Testing, vulnerabilities, Threat, Threat Modeling, Threat Trees

I. Introduction

Software security testing is a process for validating the secure implementation of a software. The objective of security testing is to find vulnerabilities, keep them away from the final product, and confirm that the security of the software is at an acceptable level [10]. Software security issues like denial of service attacks, corruption of data and disclosure of confidential information have been a major concern to the cyberspace community. Though more effort has been devoted to enhance the security of the organization still they are subjected to more cybercrime and most of the applications are now vulnerable to some form of attacks. To date, researchers have developed various security testing techniques that generate test cases or identify vulnerabilities focusing on specific attacks like SQL injection or XSS, generate test cases using model based approaches, threat modeling or use case modeling. Although each of these approaches has strengths and weaknesses for different types of security issues, threat modeling particularly provides a

systematic way to identify threats that might comprise the security. This approach can be used not only during software testing phase but also during software design and development phases for secure design and risk mitigation. Due to these advantages, a threat modeling approach has been a well-accepted practice by the industry [3].
Threat modeling is a method of identifying and documenting the security risk associated with an application. It identifies the critical assets of the system, analyze those and prioritize system vulnerabilities. It also helps in identifying the entry points and exit points of the system in advance during design phase so that there won't be any loopholes left for the attackers to compromise the system. Threat modeling must be addressed early in SDLC, so that organization might eliminate threats at a proper time and prevents it getting dissipated to the subsequent phases. During the design phase the designer must think in terms of the attacker's point of view, the possible assets that can be attacked, the ways by which those assets can be attacked are to be analyzed properly. After getting the various attack possibilities, a visualization diagram can be drawn with the available information by focusing on the ways by which an attacker could possibly attack the system. Most probably DFD or UML is the obvious choice for producing the attack visualization as well as data flow. DFD is preferred for structured programming and UML is used for object oriented programming. However researchers are focusing on the technique of making the visualization of DFD in UML. This might help in the developer to develop a secured code and tester tests the system in a more effective way to give a secured system [9]. In this article we have drawn a security framework where we have taken two UML diagrams like misuse case and misuse deployment diagram. From these two diagrams basic threats are identified. Than these threats are analyzed to determine detailed threats. Threat trees are built from the threats. After traversing the threat trees threat sequences are generated. Than we plan to optimize the test sequence by random test generation method.

The rest of the paper is organized as follows: section II describe the different threat modeling approaches. Comparative Analysis table of various threat modeling are described in section III and conclusion and future work are described in section IV.

II. Different Threat Modeling Approaches

A. Generation of threat trees from DFD (Data Flow Diagram)

Huang Song et al [1] proposed an approach where they generated test cases with three steps. Identify typical security defects, build threat trees from the defects, generate test cases. For revealing the typical security defects they have taken the DFD of a software (Logging on Process) . From the DFD they build threat trees as follows: (a) a typical security defect is selected as the root of the tree (b) typical security defect is analyzed: Analyze the direct condition of test that defect. That defect is marked as the father node, and all of its direct condition are regarded as its sub-node. (c) father node's type (AND/OR) is represented as the logic relationship with its sub-node. The sub-nodes whose father node's type is AND must be queued from left to right as the node events' order. (d) each sub-node is decomposed if possible in to father and it's sub-node and process it as step b. Then repeat step a to step c until all sub-nodes cannot be decomposed or needn't be decomposed. In their research they have analyzed some defects but elaborately described about SQL injection defect and build a threat tree for the same. After the threat tree is built, the test sequence is generated by traversing through it. A threat tree traversal algorithm (Tri-T algorithm) based on depth-first search is used to generate test case. Aaron Marback [2] et al designed a technique that finds the test cases from design components.

They developed a security testing techniques which not only generate test cases but also make executable test cases. Threat modeling consists of the following four steps. First, the application is decomposed into different components then it is represented in terms of data flow diagrams (DFDs). Second, the identified components/assets are represented as threat targets then threats are identified according to the STRIDE category and threats are modeled using threat trees. Third, the risks of the threats are evaluated using the DREAD method and threats are ranked by descending order of security risks. They have taken a case study of osCommerce and for it they have identified five threat trees and presented only three out of these five threats. The five threats are as- a data modification threat, an user credential disclosure threat, an SQL injection threat, a cross site scripting threat, a multiple log in attempt threat. They have developed a technique that: (1) With the help of Microsoft Threat Modeling Tool they created threat trees and analyzes the same. (2)They put the relevant data into a tree data structure. (3) Traverse this tree data structure to generate valid test sequences (3) A mechanism is provided to map actions in the test sequences to the applicable tests (4) An executable test script is generated for every test sequence (5) an input specification is provided for every test script and (6) test inputs are generated including valid and invalid inputs. Aron Marback et al [10] designed a threat model-based security testing approach that automatically generates security test sequences from threat trees and transforms them into executable tests. The security testing approach they considered consists of three activities: 1. building threat models with threat trees 2. Generating test sequences from threat trees 3. Creating executable test cases by considering valid and invalid inputs. They have presented two real-world web applications of osCommerce (a web-based storefront and shopping cart application) and Drupal to evaluate the effectiveness of security testing approach. Through the threat modeling process they have identified 13 threats for the osCommerce application and presented only six out of these 13 threats. The six threats are 1) Hijacking a user's session ID. 2) Stealing user information with SQLinjection 3) Multiple login attempts 4) Data modification. 5) Denial of service using local resources. 6) User data disclosure. After developing threat trees, they generated executable test cases from them. To do this, they implemented a test generation technique that (1) imports threat trees created using the Microsoft Threat Modeling Tool and creates test sequences; (2) provides a graphical interface so that testers map events (gathered from the threat trees) in the test sequences to the applicable unit tests; (3) adds input parameters to test sequences; (4) generates test inputs including valid and invalid inputs; and (5) generates test scripts that execute all test sequences with assigned input values.

B. Generation of threat trees from UML diagrams

i. Threat modeling with UML misuse case and activity diagram:

Inger Anne Tondel et al [3] presented an approach where both the misuse cases and attack trees are linked together to build a high-level view of the threats. Further, they addressed the links to security activity descriptions in the form of UML activity graphs which can be used to describe mitigating security activities for each identified threat. Their approach mainly traced software functionality to threats. Misuse cases are an extension of UML use cases in order to specify functionality that is not wanted in a system. Their purpose is to provide support for eliciting security requirements. As an addition to showing regular actor/use case relations, misuse case diagrams can model threats (misuse cases) that threaten use cases, and countermeasures that mitigate these threats. The attacker is represented as a misuser that initiates the misuse cases, either intentionally or inadvertently. Attack trees aim at modeling security threats by focusing on the attackers and the different ways they may try to attack systems. Then, based on this knowledge, system developers are more likely to design countermeasures that are able to hinder these attacks. In their approach misuse case diagrams give an overview of threats towards an application and the other model types provide more details on threats and countermeasures. Fabricio A. Braz et al. [4] proposed an approach which employed two things: identification of system threats and determination of policies to stop or mitigate their effects. For this, first they have investigated the flow of events in a use case or group of use cases to expose the related threats and from the threats security use cases are determined then they selected an appropriate security policies to mitigate the identified threats. They also improved the ways to find the policies to control these threats and map the policies to security patterns.

ii. Threat modeling with UML sequence diagrams:

Linzhang Wang et al. [5] developed a threat model-driven security testing technique which identify the undesirable threat behavior at runtime. Threats are modeled with UML sequence diagrams. From a design-level threat model they extract a set of threat traces, each of which is an event sequence that should not occur during the system execution. They establish a linkage between models, code implementations, and security testing which are extended to form a systematic methodology that can test certain security policies. A UML sequence diagram is used to model the behavior of the system by representing the realization of a use case scenario. It identifies the objects and classes involved in the scenario and the sequence of messages exchanged between the objects in order to realize the behavior of the scenario. UML sequence diagrams are exploited to describe the interactions an intruder would go through to misutilise the system, resulting in a threat. The threat scenario is represented by a sequence of message exchanges. A message sequence is a path from the first message to the last message in the sequence diagram. They applied the grey-box coverage criteria to the sequence diagram to get the paths, each of which represent an independent threat scenario. In this paper, they focused on message exchanges via procedure calls. A threat model represented by a sequence diagram (*SDTM*) consists of a tuple of (*O,M,E*) where O is a finite set of objects , E is a finite set of events and M is a finite set of labeled messages. By traversing the possible paths of the sequence diagram, they got the message sequences. Each message sequence represents a single scenario of the threat model. First, message sequences are developed from the threat models than the threat traces are derived from each message sequence. Second, track the threat scenario depicted in the threat model in a running program. Third, to exercise the program under test more thoroughly, the random testing method is applied. After they get the input parameters and their corresponding domain, a large amount of test cases can be created, and arranged into a data pool. Fourth, they recompile the instrumented code, and design a test driver to input test execution with random test data. The runtime execution traces are recorded into a trace file. Last, the runtime execution traces are matched with the threat traces in order to determine whether the threat is still existing.

iii. Threat modeling with UML deployment diagrams:

Susan J. Lincke et al. [6] considered deployment diagram for analyzing security threats because it provides a single eye view of possible security attacks and the security defenses to mitigate them. This technique can be used in more than software development, since it may be used in audit, testing, security planning, and security education. They have employed the Misuse Deployment Diagram to sketch the implementation of a single application or service with it's software and hardware application and security components. It is a useful system design phase, where functions of the software are allocated to hardware, software and software components. A preliminary MDD is needed in the requirements phase where major attacks are addressed. It doesn't replace other UML security diagrams, but is enhanced by them and provides a single –picture overview of the security system. They have presented a Web Registration Case Study in which in the initial stages, email addresses are collected from users at presentations. The users were then emailed a link to the website where materials could be obtained. Feedback was collected annually. Biswajit Ghosh et al [7], proposed an approach on client-server based application where they have used misuse cases and misuse deployment diagram to detect the possible threats. After analyzing the detailed threats they have created the threat trees and derive the test sequences by traversing the threat trees. They have taken an example of a blog site to show how their approach works. The blog site consists of few of the following requirements and features. Every user needs to register to the website before he or she can see the available posts and contents of the site. Users can manage his or her account such as change password and personal details. Users can post or upload videos, images etc. that other users can see. Admin can delete any post that other users have uploaded if found inappropriate. Based on these features they developed Misuse deployment diagrams and Misuse cases of the system for understanding the security risks.

Step 1: Constructing Misuse cases and Misuse deployment diagrams.

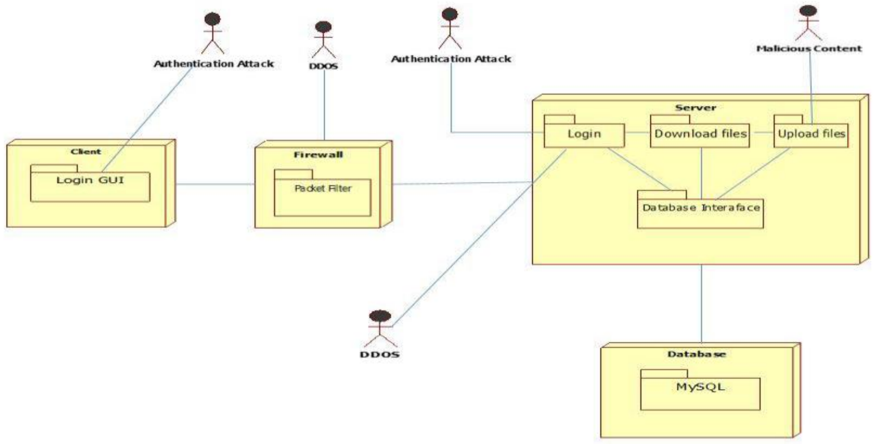

Fig.1 A Misuse Deployment diagram showing an overall view of the blog site

From these diagrams the possible basic threats are identified like- a) Authentication attack. b) Upload malicious content. c) Denial of service. d) Stealing information.

Step 2: Analyze threats to determine detailed threats. *Step 3:* Construct threat trees.

a) Authentication Attack:

i) SQL injection

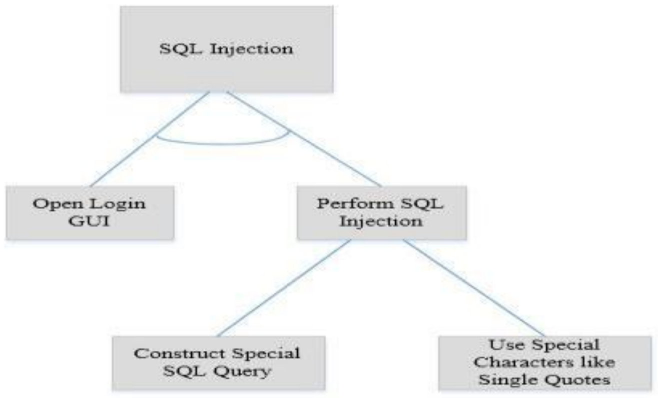
SQL Injection
Open Login GUI
Perform SQL Injection
Construct Special SQL Query
Use Special Characters like Single Quotes

Fig. 2 Threat tree of SQL injection attack

b) Upload Malicious content Attack: i) XSS attack

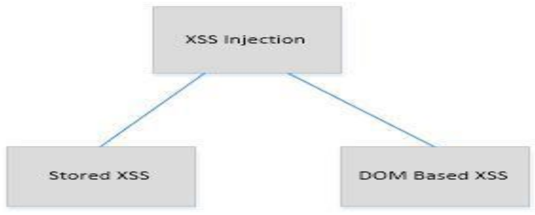

Fig. 3 Threat tree of XSS attack

Step 4: Generating test sequences from threat trees. Threat trees are traversed using algorithm called Tri-T which is proposed by Huang Song et al [1] and the results are

a. Authentication attack: SQL Injection→ Open Login GUI → Construct special SQL query.

b. Authentication attack: SQL Injection→ Open Login GUI→ Use special characters like single quote.

III. Comparative Analysis

We have studied several research articles related to threat modeling techniques of security testing and after analyzing their techniques we have observed the number of threats identified by each article and given a brief evaluation in Table I. Comparative analysis.

Table I. Comparative Analysis

Sl.No.	Authors	Techniques Used	Test Application	Results
1.	*Linzhang et al.[5],2007*	UML sequence diagram	ATM	Presented one threat scenario i.e unauthorized access.
2.	*Fabricio et al.[4], 2008*	Activity Diagram	Open Account and Money Transfer module of Finance application	Modeled 8 threats also presented a security policy.
3.	*Aron et al. [2], 2009*	Threat Trees.	osCommerce	2. Identified 5 threats. Out of 5 threats 3 threats are represented.
4.	*Huang et al.[1], 2010*	Threat Trees, Tri-T algorithm(based on DFS)	Log In process of Registration module	Identified 5 threats like: SQL injection, Meta character, Authentication, Improper error handling, CSS. Described elaborately about SQL injection threat.
5.	*Inger et al. [3], 2010*	Misuse case, Attack Trees, Activity diagram.	SHEILD repository	Presented the threats of CSS along with it also provides some countermeasures to mitigate the threats.
6.	*Susan et al.[6], 2012*	Misuse case diagram, Deployment diagram	Web Registration	Provides threats like DDOS, spoofing, SQL injection and CSS both in client side and server side applications.

7.	*Aron et al. [10], 2013*	Threat Trees, Test sequence generation algorithm based on BFS.	osCommerce, Drupal	1.Identified 13 threats and out of it six threats presented.
8.	*Biswajit et al.[7], 2014*	Misuse case diagram, Deployment diagram, Threat Trees.	Client-Server based Application	Identified 8 threats like SQL injection, Usrename & Password Enumeration, Bruteforce attack, XSS, DDOS attack, Stealing information, Evesdropping, Steal cookies.

IV. Conclusion and Future Work

We have gone through several research articles relating to security testing using threat modeling where DFD or UML is the obvious choice for generating the attack visualization as well as data flow. By using DFD or UML different threats are identified and then represented using threat trees. Here we have proposed a security framework based on the research of Ghosh et al.[7] which is diagrammatically represented in Figure 4. where basic threats are identified from the misuse case and misuse deployment diagram then all possible types of threats are analyzed. After the creation of respective threat trees different test sequences are derived then we will apply a random test generation technique to optimize the test sequence which can be able to provide a possible attack path or a threat scenario.

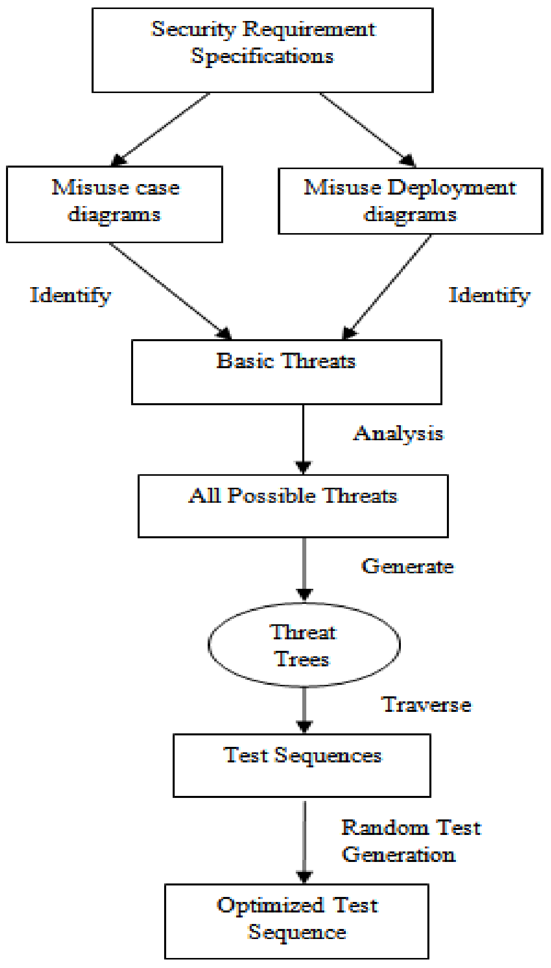

Fig 4. Security framework.

References:

1. Song, Huang, Wang Liang, Zheng Changyou, and Hong Yu. "A software security testing method based on typical defects." In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, vol. 5, pp. V5-150. IEEE, 2010..
2. Marback, Aaron, Hyunsook Do, Ke He, Samuel Kondamarri, and Dianxiang Xu. "Security test generation using threat trees." In *Automation of Software Test, 2009. AST'09. ICSE Workshop on*, pp. 62-69. IEEE, 2009.
3. Tøndel, Inger Anne, Jostein Jensen, and Lillian Røstad. "Combining misuse cases with attack trees and security activity models." In *Availability, Reliability, and Security, 2010. ARES'10 International Conference on*, pp. 438-445. IEEE, 2010..
4. Braz, Fabricio A., Eduardo B. Fernandez, and Michael VanHilst. "Eliciting security requirements through misuse activities." In *19th International Conference on Database and Expert Systems Application*, pp. 328-333. IEEE, 2008.
5. Wang, Linzhang, Eric Wong, and Dianxiang Xu. "A threat model driven approach for security testing." In *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*, p. 10. IEEE Computer Society, 2007.
6. Lincke, Susan J., Timothy H. Knautz, and Misty D. Lowery. "Designing system security with UML misuse deployment diagrams." In *Software Security and Reliability Companion (SERE-C), 2012 IEEE Sixth International Conference on*, pp. 57-61. IEEE, 2012.
7. Ghosh, Biswajit, and Arup Abhinna Acharya. "A Novel Approach for Security Testing of Client Server Based Applications using Misuse Deployment Diagrams, Misuse Cases and Threat Trees." Priya, S. Shanmuga, and S. S. Arya. "Threat Modeling for a Secured Software Development." *International Journal of Advanced Research in Computer Science* 7, no. 1 (2014).
8. Marback, Aaron, Hyunsook Do, Ke He, Samuel Kondamarri, and Dianxiang Xu. "A threat model based approach to security testing." *Software: Practice and Experience* 43, no. 2 (2013): 241-258.