A Novel Approach for Security Testing of Client Server Based Applications using Misuse Deployment Diagrams, Misuse Cases and Threat Trees

Biswajit Ghosh[1], Arup Abhinna Acharya[2]
[1]*(School of Computer Engineeering, KIIT University, India)*
[2]*(School of Computer Engineeering, KIIT University, India)*

***Abstract*:** *Security testing is one of the most important security practices today. To secure an application it's important to go for a security testing phase during the development life cycle. Many useful enhancements are done using UML diagrams to model security like Misuse cases, Mis-sequence diagrams and Misuse deployment diagrams etc. Misuse deployment diagrams can be used to model a client server environment with security aspects. Since in client server environment it is important to know where to apply your security, it's better to use Misuse Deployment diagrams which can give an overall view of the system in a single model. Based on this fact, an approach is proposed in this paper by combining Misuse deployment diagrams and Misuse Case diagrams with threat trees to generate security test cases. The approach is demonstrated with a case study where Misuse deployment diagrams, Misuse Cases are used to show that possible security defects can be identified using these diagrams and combined those with threat trees. Finally the threat trees are traversed to generate test sequences. This approach is suitable for detecting threats that need to be tested in location specific manner in client-server applications.*

***Index Terms*:** *Security testing, Misuse deployment diagrams, threat trees, Misuse cases, test sequence.*

I. Introduction

Many enhancements are done on UML to model security aspects of a system like Misuse case diagrams, Mis-sequence diagrams, Misuse deployment diagrams etc. Each has their own advantages and uses. We will mainly focus on Misuse deployment diagrams, which are deployment diagrams that show security concerns. According to The Open Web Application Security Project's [1] top 10 security defects of 2013, top attacks are like injection attacks, cross site scripting, sensitive data exposure etc. These are the biggest threats to a web application and known as possible defects. According to a survey by CERT/CC [13] 75% of security breaches are because of these defects. Modeling a client-server environment with Misuse deployment diagrams are beneficial [2], because it is required to locate where security need to be implemented. Mis-sequence diagrams can also be used but it will be difficult to understand because of the complexity of message passing between each objects. It will lack to give an overall view of a system with a single diagram. Misuse deployment diagrams can give an overview of a systems security with the help of a single diagram. So finding defects both at server and client side is easier. Misuse cases can also identify possible attacks but they are unable to locate where to apply security. Misuse cases are useful mainly in requirement phase of software development to identify possible security requirements. Misuse cases and Misuse deployment diagrams can be combined to understand system security deeply. We can take help of OWASP [1] top attack list to identify possible defects from the diagrams.

In this paper a novel approach is given for security testing of Client server based applications by combining Misuse deployment diagrams, Misuse case diagrams with threat trees and identifying possible defects.

Rest of the paper is organized as follows: Section II discusses some basic concepts about threat modeling and security testing with illustration. Section III presents discussion about related works. Section IV describes the proposed approach of security testing. Section IV is divided into two subsections A and B. In Subsection A, a case study is given and in Subsection B the results are shown. In Section V the proposed approach is compared with some related works. In Section VI, discussion and possible future works are given. And Section VII contains the conclusion.

II. Basic Concepts

a) Threat Modeling [3]

Threat modeling is a method of documenting and identifying the security risks associated with an application. This methodology helps development teams to identify security strengths and weaknesses. Its primary goal is to identify attack goals that can be used to secure a system. The main process of threat modeling is given below

i. Understanding adversary's view.
ii. Characterizing security of the system.
iii. Determining threats.

Now in analyze threat phase that is a subpart of Determining threat, we can analyze the threats that is found during many modeling approaches using threat trees (non-UML) [3], threat nets (petri-nets), Mis-sequence diagrams, Jackson's problem diagram by Haley et al [4]. Misuse activity diagrams suggested by Braz et al [5]. Since our aim is to use models that are easily understandable and lightweight so we are using threat trees.

b) Threat trees [3]

Threat trees are used to find out whether the conditions necessary for a threat to be realized exist and unmitigated. A threat tree consists of root node, which is the main threat, and it has one or more child nodes that need to be true for threat to be realized. Two nodes at the same level can be connected to its parent node with an AND or an implicit OR condition.

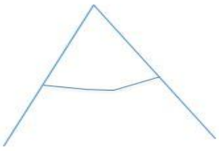

Fig 1: AND node

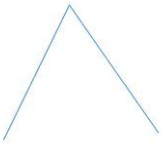

Fig 2: OR node

c) Misuse deployment diagrams

Misuse deployment diagrams are useful to model a client-server application in a distributed environment where software is related with many hardwares also. Location where to place security code in client-server environment is as important as having the code. Misuse deployment diagrams consist of three things-

i. Nodes
  a. Device nodes(Hardware)
  b. Execution environment nodes (eg browser)
ii. Artifacts.
iii. Misuses: We can merge Misusers with Misuses and only represent it as Misuses in the diagram.

III. Related Works

Software security testing aims at finding vulnerabilities in a system. The main problem of security testing is it requires experience and resources. It is difficult for developers and testers who are new in this field. Many work has been done to give a way how to model security and then generate security test cases from them. Pari Salas et al [6] proposed an approach to generate test cases from three models, specification, implementation and attack model. Their approach consists of generating test cases and automating the process from implementation details.

Threat modeling has become a viable practice for secure software development. Threat modeling is comprised of three high level steps: understanding the view of attacker, characterizes the view of attacker, characterize security of the system and finally determine threats [3]. Based on this many work has been done to model security and generating security test cases. Marback et al [7] proposed an approach of generation of test cases from threat trees. Threat tree is a threat analysis model in a tree structure which shows how to realize a threat. They have used DFD's to model system functionality and then developed threat trees from possible threats and finally generated test cases. Song et al [8] proposed an approach similar where possible threats are identified from the data flow diagrams of a system. Then from identified possible defects threat trees are generated and after traversing the threat tree they generated test sequence. Many modification s are done on UML diagrams later to model security in an object oriented development. Tondel et al [9] proposed an approach to combine UML Misuse case diagrams with non-UML threat trees. Here instead identifying threats from DFD's they proposed a model to identify threats from Misuse case diagrams and finally linking threats with threat trees to get a broader view of the threats and how to mitigate the threats. Lincke et al [2] proposed a new model which is extension of UML deployment diagram, in which security is represented. They named it Misuse deployment diagrams. It has several advantages over Misuse cases in terms of the location specific identification of the threats. Based on this approach in, in this paper, we have combined Misuse deployment diagrams with threat trees to generate more suitable test cases for a client-server environment.

IV. Proposed Work

Securing a client-server application needs an overall knowledge about the server side and client side security, keeping in mind an approach is given where Misuse deployment diagrams are used to get a clear overall view of the system. Misuse case diagrams are combined for better understanding of the system. From that basic threats are identified. Next we need to analyze each threat for determining detailed threats. Each threat can be achieved through several ways. These ways are the detailed threats to the system. Next threat trees are created from each detailed threat to know how it can be realized. Last Threat trees are traversed to get test sequences. We could have used Misuse cases to identify security requirements but that wouldn't have given us detailed view of the security risks possible at specific places like server and client side. Mis-sequence diagrams can give location specific information but due to complex massage passing between objects can make it complex to identify threats. So Misuse deployment diagrams are used that will give an overview with a single diagram and we can specify several attacks based on specific location. An example of generated test sequence is given that will clear how location is important in some cases of security. Steps for the given approach can be summarized as follows:

1) Model Misuse deployment diagrams and Misuse cases to identify basic threats.
2) Analyze basic threats to determine detailed threats.
3) Generate threat trees from detailed threats.
4) Finally test sequences can be generated by traversing threat trees.

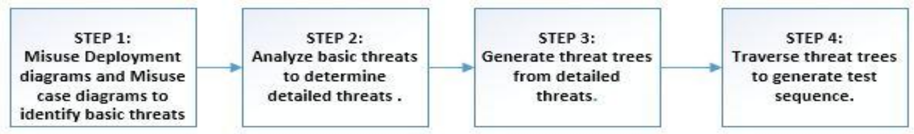
STEP 1:
Misuse Deployment diagrams and Misuse case diagrams to identify basic threats
STEP 2:
Analyze basic threats to determine detailed threats .
STEP 3:
Generate threat trees from detailed threats.
STEP 4:
Traverse threat trees to generate test sequence.

Fig 3: A framework for security testing of Client-Server based application.

A. Case Study

We have taken an example of a blog site to show how our approach works. The blog site consists of few of the following requirements and features.

i. Every user needs to register to the website before he or she can see the available posts and contents of the site.
ii. Users can manage his or her account such as change password and personal details.
iii. Users can post or upload videos, images etc. that other users can see.
iv. Admin can delete any post that other users have uploaded if found inappropriate.

Based on these features we can create Misuse deployment diagrams and Misuse cases of the system for understanding the security risks. Now we can apply our approach as follows:

Step 1: Constructing Misuse cases and Misuse deployment diagrams.

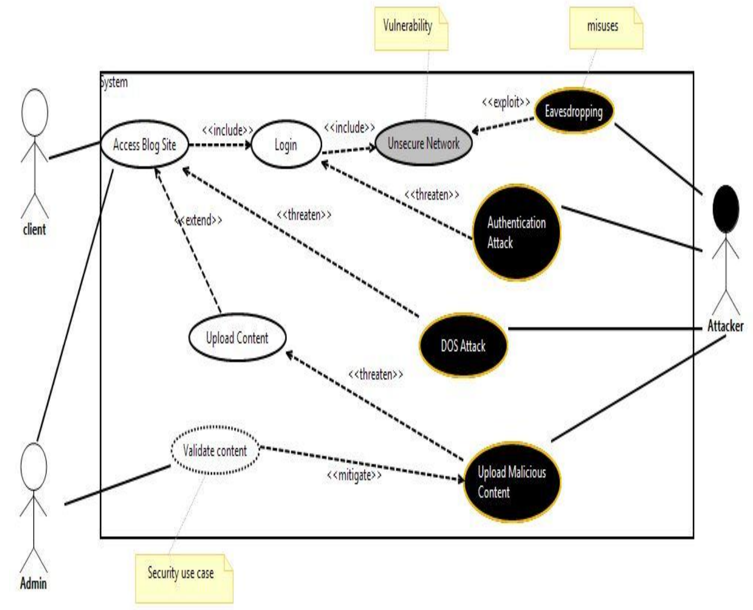

Fig 4: Misuse Case diagram of the Blog site.

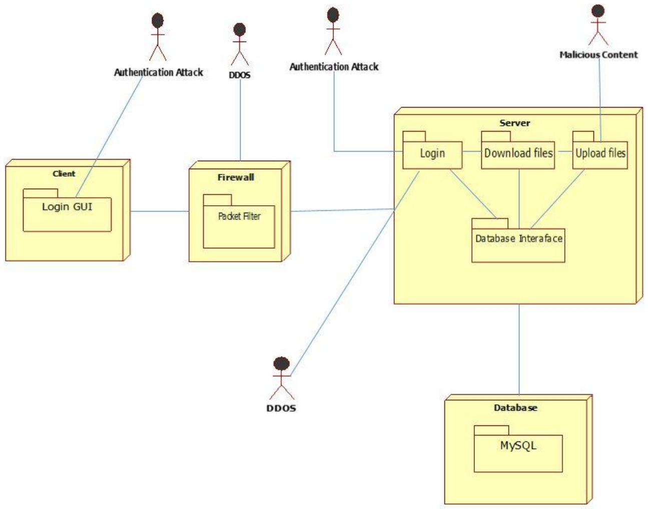

Fig 5: A Misuse Deployment diagram showing an overall view of the blog site.

Now from these diagrams we can identify the possible basic threats like-

a) Authentication attack.
b) Upload malicious content.
c) Denial of service.
d) Stealing information.

Step 2: Analyze threats to determine detailed threats.

a) Authentication attack can be achieved using
i) SQL Injection,
ii) Username and password Enumeration,
iii) Brute force attack etc.
b) Upload malicious content
i) XSS attack
ii) Upload contents violating sites policy etc.
c) Denial of service
i) Distributed denial of Service attack.
d) Stealing information:
i) Man in the middle attack or eavesdropping
ii) XSS to steal user side cookies etc.

Step 3: Construct threat trees.

a) Authentication Attack:

i) SQL injection

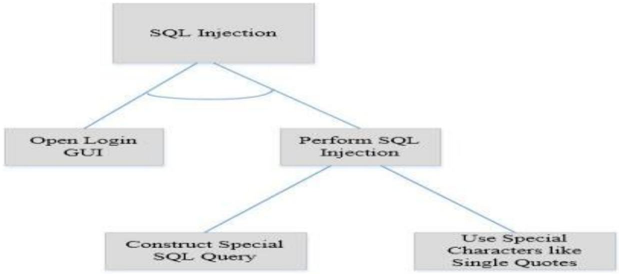
SQL Injection
Open Login GUI
Perform SQL Injection
Construct Special SQL Query
Use Special Characters like Single Quotes

Fig 6: Threat tree of SQL injection attack.

ii) Username and Password Enumeration

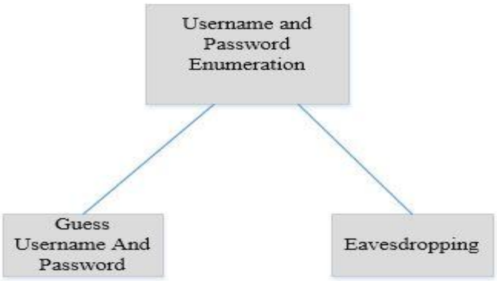
Username and Password Enumeration
Guess Username And Password
Eavesdropping

Fig 7: Threat tree of Username and Password Enumeration attack.

b) Upload Malicious content Attack:

i) XSS attack

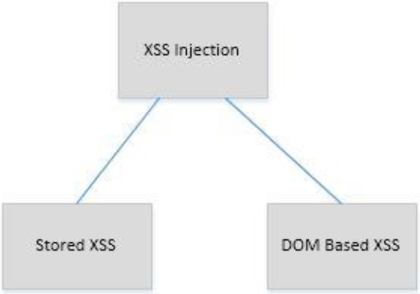

Fig 8: Threat tree of XSS attack.

ii) Upload contents that violates sites policy

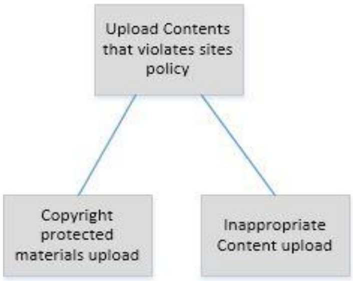
Upload Contents that violates sites policy
Copyright protected materials upload
Inappropriate Content upload

Fig 9: Threat tree of Upload contents that violates sites policy

For simplicity we have shown threat trees of Authentication attack and Upload malicious content 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 [8]. Starting from the root node that is the root attack we have to check the type of the nodes. If it is an AND node then we have to include all the connected child nodes to get the final sequence and if it is an OR node then we can go any of the connected child nodes and get a sequence. The details are given in the Tri-T algorithm [8].

B. Results

For illustrating our approach we have shown the results of Authentication attack only.

Applying Tri-T algorithm on threat trees the following results are achieved:

1. From SQL injection threat tree in Fig 6 under authentication attack we got

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.

2. Results achieved from threat tree of username and password enumeration under authentication attack in Fig 7 are:

a. Authentication Attack: Username and password Enumeration→Guess username and password.

b. Authentication Attack: Username and Password Enumeration→Eavesdropping.

Now as you can see from the Misuse Deployment diagram in Fig 5 we can specify where to apply these test sequences. If we take the test sequence that is:-

Authentication attack: SQL Injection→ Open Login GUI→Use special characters like single quote, in login field it can give us a false negative if client side validation is provided and won't allow us to insert special symbols. But the attack is still possible if we bypass client side validation using HTTP tampering and modification during transition. So we have to apply this same test sequence to test server side security by deactivating client side validation and test whether the server can still filter the malicious input and that will give a correct result whether the application is secure from SQL injection using special characters or not. Only client side validation will not help. From username and password enumeration threat tree in Fig 7 we get a result that is:-

Authentication attack: Username and Password Enumeration→Eavesdropping. This is a special kind of attack that is difficult to represent with Misuse deployment diagram also because it occurs during transition but it can be understood from Misuse case diagram in Fig 4. Here we have to check whether the data in transit can be read if sniffed during transition. HSTS (HTTP Strict Transport Protocol) [10] can be used to prevent this where all data will be transferred with HTTPS protocol. We need to test the server whether it supports HSTS or not. So here also location of attack and where to apply test cases is important. In this way other test sequences can be generated from other threat trees also.

V. Comparison with related works

We have gone through several papers related to our work and after analyzing their techniques we have given a brief evaluation in Table 1. The parameters are as follows:

1. Related work: Name of the paper or work which is being compared.
2. Technique Used: Technique used for security testing.
3. Implementation Knowledge Required: Whether the approach requires code level or detailed implementation details.
4. Support for Client-Server based applications: Whether the approach for security testing or generated test cases can handle variations of attack possible in a client-server based application where location specific security code is required.
5. Support for OOD: Whether the approach supports Object Oriented Development or any UML diagram support is discussed.
6. Executable Test Case: Whether the approach contains information about how to generate executable test cases or test scripts that can automate the process of testing.

Table 1: Comparison with related works

Sl. no	Related Works	Technique Used	Implementation Knowledge Required	Support for Client-Server based applications	Support for OOD	Executable Test case.
1	A Software Security Testing Method Based On Typical Defects [8]	From DFD of a system they have identified Typical Defects and generated Threat trees. And finally traversing threat trees they have generated test sequences	No	Not given	Not given	No
2.	Security Test Generation using Threat Trees [7]	Using Threat Modelling and DFDs of a system to generate Threat trees. Traversed threat trees and finally provided an approach to make test sequences generated from threat trees executable with added parameters	Yes	Not given	Not given	Yes
3	A Threat Model Driven Approach for Security Testing [12]	From UML sequence diagrams they extracted threat traces and after applying random test cases in the system they compared if both consists the same traces. If matched a security threat gets detected in the system.	Yes	Not given	Yes	Random test cases were used and results were mapped against threat model based threat traces.

4.	Model-based Security Vulnerability Testing [6]	Combined three models Specification Model, Implementation Model, And attack model. And based on these test cases were generated	Yes	Not given	Yes	Yes
5	Our Approach	Used Misuse Deployment diagrams and Misuse cases to identify possible threats and location information of possible attacks. After analyzing and identifying detailed threats we have generated threat trees and traversing trees we get the test sequences	No	Yes	Yes	No

VI. Discussion and Future Work

We have not applied our approach on any working system. We have just taken an example to show how our approach can be beneficial. We have to apply our work and see the results and based on that further modification may be required. We have shown few of the attacks possible to describe our approach but there are more complex attacks possible so we will try to work on those attacks and try to show how it can be represented and test cases can be generated. We have used SeaMonster [11] to draw the Misuse case diagram shown in Fig 3 and Used Microsoft Visio and Star Uml to draw the Misuse Deployment diagram in Fig4 and all the threat trees in Fig 5, Fig 6, Fig 7 and Fig 8.

VII. Conclusion

Based on the research of Lincke et al [2], we presented an approach to link Misuse Deployment diagrams with threat trees so that we can identify more threats in compare to other approaches. The approach is given keeping in mind of developers who are new in security and dealing with client-server based applications. The test cases or test sequences generated by our approach can be reused in similar kind of application. Since these test sequences are generated to clear the understanding about how to go for testing specific threats and they are not generated based on implementation so they cannot be directly executed but can be used as a template for similar types of applications. But developers can add parameters and values according to the implementation of their system and execute these test cases.

References

[1]. Open Web Application Security project (OWASP) www.owasp.org/index.php/Top_10_2013-Top_10

[2]. Susan J Lincke, Timothy H. Knautz and Misty D.Lowery, " Designing System Security with UML Misuse Deployment Diagrams ", in 2012 IEEE Sixth International Conference on Software Security and Reliability Companion, 978-0-7695-4743-5/12 IEEE, pp 57 – 61.

[3]. Frank Swiderski and Window Snyder, Threat Modeling: Microsoft Professional, 2004.

[4]. Charles B. Haley, Robin Laney, Jonathan D Moffett and Bashar Nuseibeh, " Security requirements engineering: A Framework for Representation and Analysis," in IEEE Transactions on Software Engineering, vol. 34, No. 1, 0098-5589/08 IEEE, Jan.-Feb, 2008, pp 133 – 153.

[5]. Fabricio A Braz, Eduardo B Fernandez, and Michael Van Hilst, " Eliciting Security requirements through Misuse activities " in 19th International Conference on Database and Expert Systems Application, 1529-4188/08 IEEE, 2008, pp 328 – 333.

[6]. Percy A. Pari Salas, Padmanabhan Krishnan, and Kelvin J. Ross, "Model-based Security Vulnerability Testing", in Australian Software Engineering Conference (ASWEC'07), 0-7695-2778-7/07, IEEE, 2007. pp 284 – 296.

[7]. Aaron Marback, Hyunsook Do, Ke He, Samuel Kondamarri, and Dianxiang Xu, " Security Test Generation using Threat Trees," AST'09, Vancouver, Canada, 978-1-4244-3711-5/09 IEEE, May 18-19, 2009, pp 62 – 69.

[8]. Huang Song, Wang Liang, Zheng Changyou, and YU Hong, "A Software Security Testing Method Based On Typical Defects", in 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), 978-1-4244-7237-6/10 IEEE, 2010, pp V5-150-153.

[9]. Inger Anne Tondel, Jostein Jensen, and Lillian Rostad, "
Combining Misuse cases with attack trees and security activity models", in 2010 International Conference on Availability, Reliability and Security, 978-0-7695-3965-2/10 IEEE, 2010, pp 438 – 445.

[10]. HSTS (HTTP STRICT TRANSPORT SECURITY) https://www.owasp.org/index.php/HTTP_Strict_Transport_Security.

[11]. SeaMonster. SHIELDS project. http://www.shields-project.eu/?q=node/30.

[12]. Linzhang Wang, Eric Wong and Dianxiang Xu, "A Threat Model Driven Approach for Security Testing" in Third International Workshop on Software Engineering for Secure Systems (SESS'07), 0-7695-2952-6/07 IEEE, 2007, pp 10.

[13]. Computer Emergency Response Team. https://www.cert.org/

AUTHOR PROFILE

Biswajit Ghosh received his B.Tech degree in the year 2011 in Information Technology from Camellia Institute of Technology, Madhyamgram, affiliated to West Bengal University of Technology and currently perusing M.Tech in Computer Science Engineering with specialization in Software Engineering from KIIT University, Bhubaneswar. His area of interest includes Software Testing, Web Application Security, and Penetration Testing.

Arup Abhinna Acharya is a Professor and research scholar in the School of Computer Engineering, KIIT University, Bhubaneswar, Odisha, INDIA. He received his Masters degree from KIIT University Bhubaneswar. His research areas include Object Oriented Software Testing, Software Cost Estimation, and Data mining. Many publications are there to his credit in many International and National level journal and proceedings. He is having more than ten years of teaching experience. He is a member of ISTE**.**