THREAT MODELLING METHODOLOGIES: A SURVEY

Shafiq Hussain[1], Asif Kamal[2], Shabir Ahmad[3], Ghulam Rasool[4], Sajid Iqbal[5]
[1]Department of Computer Science, BahauddinZakariya University, Sub-Campus Sahiwal, Pakistan
Email:shafiq.hussain57@gmail.com(Corresponding Author)
shafiq.hussain@research.sunderland.ac.uk
[2]Counter Terrorism DeparmentSahiwal, Punjab Police, Pakistan
Drasifkmal786@gmail.com
[3]Government College of Commerce, Multan
Email: mian_shabbir@hotmail.com
[4]Department of Computer Science, Comsats Institute of IT, Lahore, Pakistan
Email: grasool@ciitlahore.edu.pk
[5]Department of Computer Science,BahauddinZakariya University, Multan, Pakistan
Email: sajidiqbal.pk@gmail.com

ABSTRACT: *The security of software systems can be broadly divided into two categories namely external security and internal security. The internal security of software systems is the main issue for any secure system. This issue of software security depends on the design of software systems and integration of security features into the design. This process involves the identification of security threats to software systems, identification of relevant mitigation measures and their integration into the design of software systems. To identify security requirements of software systems, many techniques have been developed. Threat modelling is one of the approaches to identify security requirements and helps to add them into the design of the software systems. Threat modelling makes it possible to identify all potential threats to the software systems and hence helps software designers to add mitigations to make their software design more secure and reliable. Many threat modelling approaches had been developed over the time since its emergence. This paper presents a survey on various threat modelling techniques. In this paper, we have focussed on the techniques in the context of secure web applications.*

Keywords: Threat Modeling, Formal Methods, Model Checker, Security, Web Applications, SDLC, STRIDE, CORAS.

1. INTRODUCTION

A threat is a possible harm that may be occurred due to vulnerability present in the design of software applications. When vulnerability in the design of systems is exploited, threat is materialized and attack happens. So, first step is making secure software applications is to identify all possible threats to the system. For this purpose, a threat modelling technique is applied. Threat modelling is a technique that makes software systems more secure by including security measures into the design of software systems. Threat modelling is performed at the early stages of software development process with the purpose to identify all possible threats that may cause harm to the software applications. Threat modelling helps to identify all possible threats along with different mitigation techniques [1], [2], and [3]. Many threat modelling approaches had been developed ranging from textual descriptions to graphical notations along with some tool support [4] and[5]. The remaining of this paper is as follows: Section 2 describes various threat modelling approaches,section 3 contains a discussion on the existing threat modelling approaches and section 4 describesthe conclusions along with the identification of future research ideas.

2. THREAT MODELLING APPROACHES

Threat modelling is a key element to integrate security into software systems. It enables us to identify critical areas of design which needs to be protected. Over the time various threat modelling approaches and methodologies have been developed and are being used in the process of designing secure web applications. This approach varies from conceptual frameworks to practical methodologies. The following paragraphs present a brief survey of existing threat modelling methodologies and techniques.

2.1 Stride

STRIDE is most widely used threat model. STRIDE covers the main six broad categories of threats. These threat categories include Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [6]. In this model, a data flow diagram of the system under consideration is developed and STRIDE model is applied at each node of this Data Flow Diagram of the system. Then security threats are identified manually and listed. After this STRIDE model, another model called DREAD model is applied to each identified. The DREAD model is a threat rating model and is used to assign threat severity and priority level of identified threats. The DREAD model stands for Damage potential, Reproducibility, Exploitability, Affected users and Discoverability. When this process of threat rating by the DREAD model is completed, possible mitigation measures are recommended and listed [7] and[8]. STRIDE approach is a manual process and does not have tool support.

2.2 Abuser Stories

Abuser Stories [9] are the extensions of agile practices used to identify security flaws in the system. This process of identifying security flaws is done by identifying the ways how the attackers may abuse the system. In Agile modelling [10], threats modelling process is based on five elements identifying threats, understanding threats, categorizing threats using STRIDE, identifying mitigation strategies and

testing. Abuse cases are designed to launch an attack on the system and identifying the attack paths of the system. All the work in this process is manual without the support of any automated tool.

2.3 Stride Average Model

STRIDE Average Model [11,12,13] is an enhanced model of STRIDE model. In this model, software systemsare divided into five categories based on the application and the environment in which these systems will operate. These systems are then rated according to the category of the application. Then information criticality is determined based on the information and data the application will deal. Required controls are identified and STRIDE average is calculated. At the end potential security risk is calculated. All work is a manual process without the support of any automated tool.

2.4 Attack Trees

Threat modelling using attack trees [13, 14, 15,16] is another methodology used to model threats to software systems. A tree structure is used to represent attack against a system where goal is at the root node and different ways of achieving that goal are the leaf nodes. Each node becomes a sub goal, and children of that node are the ways to achieve that sub goal. OR nodes are used to represent alternatives and AND nodes are used to represent different steps toward achieving the same goal. After the tree is built different values can be assigned to the leaf nodes and these values are used to calculate the security of the goal. These values are totally dependent on the security expert and system engineer. The assignment of values is a manual process. The otherattributes such as time required completing a step, operational expense, expertise required to launch an attack etc. can also be added to attack trees. This approach has a support of an automated tool called SecureI Tree [17] developed by Amenaz Technologies. This is a graphical tool to model attack trees and it is based mathematical attack tree model. Mathematical algorithms are used to calculate the security risks. SecureI Tree has been used successfully in many applications such as buildings, pipelines and electrical transmission lines.

2.5 Fuzzy Logic

The fuzzy logic-based threat modelling [18] technique is based on the fuzzy set theory. In this technique, the STRIDE model is used as a basis for determining the input variables. The input variables determined by using the STRIDE model are passed to fuzzy inference engine. The fuzzy inference engine then generates a list of associated threats as output. This approach has a tool support called MATLAB fuzzy logic toolsetto automate the process to identify security threats.

2.6 SDL Threat Modelling Tool

SDL Threat Modelling Tool [19], [20] and[21] and Application Threat Modelling (TAM) Tool [14] developed by Microsoft are based on the STRIDE model and the DREAD model. In SDL tool and TAM tool, a Data Flow Diagram of the system under consideration is developed. This Data Flow Diagram is then passed as input to these tools. These threat modelling tools then generates reports describing all the possible threats to the system. In addition to this threat report, relevant mitigation measures are also identified.

2.7 T-MAP

T-MAP is a quantitative threat modelling method which quantifies security threats by calculating the severity weights of attack paths and is used for Commercial Off The Shelf (COTS) systems. T-MAP can tell to the management how cost effective the patching etc is. The attack path model is developed by using UML class diagrams [22]. For each step, there are four class diagrams, access class diagram, vulnerability class diagram, target asset class diagram and affected value class diagram. T-MAP has support of an automatic tool called Tiramisu. The Tiramisu tool takes three inputs: the general vulnerability information, an organization's IT infrastructure information, and how an organization's business values depend on its IT infrastructure. It calculates a complete list of associated Attack Paths, and outputs the overall threats in terms of the total weights of Attack Paths. At the core of Tiramisu is Automated Data Collection Engine which collects latest published vulnerability reports from CERT/CC, NIST, SANS, Security Focus, Symantec and Microsoft automatically, formats them and populates the Data Storage and Management layer [23] and[24].

2.8 CORAS

CORAS [25]and [26] is a graphical threat modeling and specification language. It is based on UML. By using CORAS undesirable behaviors are documented in the form of threat scenarios. The CORAS profile offers specialized use case diagrams for modeling threats and unwanted behaviors. CORAS also has a tool support called diagram editor. The diagram editor is just diagram tool to draw threat scenarios and has no analysis facilities. CORAS is based on Australian Risk Management Standard AS/NZS 4360:2004. Threat modeling by CORAS is based on these five activities: Establish the context, Identify risks, Analyze risks, Evaluate risks and Treat risks.

3. DISCUSSIONS

The following table describes a summary of threat modelling methodologies currently being used in industry and academia. Threat modelling is relatively a new field. Only few approaches are currently in practice. Majority of the above described threat modelling tools and approaches are based on STRIDE model and DREAD model. Only few threat modelling approaches have support of an automated tool. The tools are just for drawing purposes for threat scenarios. These tools do not provide a rigorous approach for threat analysis. For more rigorous analysis of threat modelling and security risks to software applications, formal methods based threat modelling tools are need.

Methodology	Secure	Tool	Web Apps
STRIDE	✓		✓
Abuser Stories	✓		✓
STRIDE	✓		✓
Attack Trees	✓	✓	✓

Fuzzy Logic	✓	✓	✓
SDL	✓	✓	✓
T-MAP	✓	✓	✓
CORAS	✓	✓	✓

Threat analysis is very important because many threats are based on other threats. Vulnerability may cause other vulnerabilities. Therefore, threats are interrelated. Issues like inconsistencies and incompleteness always remain there if there is no proper analysis of all possible threats. Other issues like deadlocks, race conditions etc. cannot be resolved by using these existing threat modelling approaches. There is no existing threat modelling approach which provides support for the use formal methods, a very promising approach.

4. CONCLUSIONS

Threat modeling is a process to identify security threats to software applications. In this short survey, a number of threat modeling approaches have been presented. Along with the description of the threat modeling approaches and methodologies, a description of automated tools has also been presented. There are limitations for rigorous analysis of various threats and vulnerabilities in all the existing threat modeling approaches. There is a need for future research for automated analysis of threats and vulnerabilities. Secondly, there is a need of research on the application of formal methods into software security. Threats may be formally specified by formal specification languages such as Event-B and can then be verified and validated by automated tools such as RODIN.

REFERENCES:

1. B. Potter, 2009. Microsoft SDL Threat Modelling Tool. *Network Security*, 2009(1), 15–18.
2. S. Myagmar, A.J Lee, &W. Yurcik, 2005. Threat modeling as a basis for security requirements. In *Symposium on Requirements Engineering for Information Security (SREIS)*.
3. D. De Cock, K. Wolters, 2004. Threat modelling for security tokens in web applications. In *Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004)*.
4. J. Viega, 2005. Building security requirements with CLASP. In *Proceedings of the 2005 workshop on Software engineering for secure systems—building trustworthy applications*. p. 7.
5. J. Whittle, D. Wijesekera & M. Hartong, 2008. Executable misuse cases for modeling security concerns. In *Proceedings of the 30th international conference on Software engineering*. pp. 121–130.
6. A.S Sodiya, S. A. Onashoga, and B. A. Oladunjoye. "Threat modeling using fuzzy logic paradigm." *Informing Science: International Journal of an Emerging Transdiscipline* 4.1 (2007): 53-61.
7. S.V Casteele, 2005. Threat modeling for web application using STRIDE model.
8. Hernan, Shawn, "Threat modeling-uncover security design flaws using the stride approach." *MSDN Magazine-Louisville* (2006): 68-75.
9. Peeters, Johan. "Agile security requirements engineering." *Symposium on Requirements Engineering for Information Security*. 2005.
10. W.S Ambler, "Introduction to security threat modeling. Agile Modeling." (2005).
11. J.P Jesan, "Threat modeling web-applications using STRIDE average model." *Computer Security Conference*. 2008.
12. Uncover Security Design Flaws Using The STRIDE Approach. Available at: http://msdn.microsoft.com/en-us/magazine/cc163519.aspx [Accessed April 30, 2010].
13. Schneier, Bruce, and Adam Shostack. "Breaking up is hard to do: modeling security threats for smart cards." *USENIX Workshop on Smart Card Technology, Chicago, Illinois, USA, http://www. counterpane. com/smart-card-threats. html*. 1999.
14. C.B Haley, 2008. Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering*, 34(1), 133–153.
15. Ahmad, Shabir, and Bilal Ehsan. "The Cloud Computing Security Secure User Authentication Technique (Multi Level Authentication)." IJSER vol.4 issue 12(2013): 2166-2171.
16. M. Hussein, & M. Zulkernine, 2006. UMLintr: a UML profile for specifying intrusions. In *Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on*. p. 8.
17. V. Saini, Q. Duan, & V. Paruchuri, 2008. Threat modeling using attack trees. *J. Comput. Small Coll.*, 23(4), 124-131.
18. A.S Sodiya, S.A. Onashoga, B.A Oladunjoye, 2007. Threat Modeling Using Fuzzy Logic Paradigm.*Information and Beyond: Part I*, 4, 53.
19. A. Shostack, 2008. Experiences Threat Modeling at Microsoft. In *Modeling Security Workshop. Dept. of Computing, Lancaster University, UK*.
20. Get Started: Threat Modeling Tool - Microsoft Security Development Lifecycle (SDL). Available at: http://www.microsoft.com/security/sdl/getstarted/threatmodeling.aspx [Accessed April 30, 2010].
21. Security Briefs: SDL Threat Modeling Tool. Available at: http://msdn.microsoft.com/en-gb/magazine/dd347831.aspx [Accessed May 1, 2010].
22. T. Lodderstedt, D. Basin, & J. Doser, 2002. SecureUML: A UML-based modeling language for model-driven security. *«UML» 2002—The Unified Modeling Language*, 426–441.
23. Ingalsbe, A. Jeffrey "Threat modeling: diving into the deep end." *Software, IEEE* 25.1 (2008): 28-34.
24. Y. Chen, B. B. Boehm, & L. Sheppard, 2007. Value Driven Security Threat Modeling Based on Attack Path Analysis. In *HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES*. p. 4698.
25. denBraber, Folker, et al. "Model-based security analysis in seven steps—a guided tour to the CORAS method." *BT Technology Journal* 25.1 (2007): 101-117.
26. D. Verdon, & G. McGraw, 2004. Risk analysis in software design. *IEEE Security & Privacy*, 79–84.