Evaluating the Effectiveness of Microsoft Threat Modeling Tool

Imano Williams, Xiaohong Yuan
Computer Science Department
North Carolina A&T State University
1601 E Market St
Greensboro, NC 27411 USA
irwilli1@aggies.ncat.edu, xhyuan@ncat.edu

ABSTRACT: *Today, it is becoming the norm for software security best practices to be incorporated into all the software development life cycle (SDLC) stages. One main reason is that web applications relentlessly being susceptible to malicious cyber at attacks by cyber criminals. Software threat modeling is one of the best practices used in software security of the SDLC. To significantly reduce software implementation bugs and design flaws it is important to incorporate software threat modeling early in the SDLC. Graduate students in a Secure Software Engineering course were introduced to the process of software threat modeling and Microsoft Threat Modeling Tool 2014. A class assignment was used to evaluate the effectiveness of the graduate students using Microsoft Threat Modeling Tool 2014 in a two part assignment: A) brainstorming/manual process to conduct threat modeling and B) Microsoft Threat Modeling Tool 2014 to conduct threat modeling. This paper presents our analysis and evaluation of students work using A and B, and the students feedback using A and B on a mock online shopping web application using the Microsoft Threat Modeling Tool 2014.*

Keywords: Software Security, Threat Modeling, Risk Management, Tool Evaluation, Architectural Risk Analysis

Received: 8 July 2015, Revised 15 August 2015, Accepted 23 August 2015

1. Introduction

In the past, network security has been emphasized more instead of software security [1]. However, most security vulnerabilities were due to insecure software, or software that was poorly designed or that was implemented with security flaws that could be exploited by cyber criminals. Therefore, it is important to integrate best practices for developing secure software into the software development lifecycle, so that software continues to function correctly under malicious attacks [2].

Though it is important to develop secure software non-experts, especially students, in the field of software security need to have a good understanding of the processes needed to carry out secure software development. According to Howard [3], until people know the issues, they will not effectively design, develop, test, or document software systems. Additionally, Howard [3] mentioned that the lack of "security as threats" education for students today, plays a key role in security unawareness. Though this statement may not be entirely true, since the teaching of software security to students has gained momentum to assist them in developing more secure software. According to Ardi [4], "*Developers should also be aware of product specific security* *issues and the threats that might affect the product after deployment*."

Students, who may become future software professionals, need to gain an understanding of software security threats and how design and implement mitigation techniques against those threats. Computer Science department in universities and colleges are responsible for their students' software security education. Therefore, it is important to teach students about threat modeling, as well as other methods and tools that would help guide students to discover, assess, and mitigate software security threats [5, 6].

In the secure software engineering course at our university, graduate students were taught some of software security best practices, namely: Risk Management Framework, Architectural Risk Analysis, and Threat Modeling. The Microsoft Threat Modeling process and the Microsoft Threat Modeling Tool 2014 were introduced to the students. The Microsoft Threat Modeling Tool goes hand in hand with the threat modeling process.

In this paper we evaluate the effectiveness of the SDL Threat Modeling tool in assisting students with threat modeling through the comparison of two different threat modeling methods by students: A) using a manual process and B) using the SDL Threat Modeling Tool 2014. Additionally, the students' feedback on the manual process and on the Microsoft Threat Modeling Tool 2014 was analyzed for any correlation with the work they produced in the threat modeling assignment using both methods on a mock online web application.

The rest of the paper is organized as follows. Section 2 provides the background information on threat modeling and the SDL Threat Modeling Tool. Section 3 explains our methodology for evaluating the effectiveness of the SDL Threat Modeling Tool. Section 4 presents the results obtained from this evaluation. Section 5 discusses the results of evaluation, the limitations of our evaluation, and the ways to improve the teaching of threat modeling. Lastly, Section 6 concludes this paper.

2. Microsoft Threat Modeling & Tool 2014

Threat modeling proposed by Microsoft is a systematic process used to identify and rank potential threats/risks that may affect design level and architectural artifacts in a software system. These potential threats can be hypothesized based on the interactions between the various data flow elements over a trust boundary. The process usually includes six steps, namely [7, 8]:

1) Identify assets: Identify the valuable assets that your systems must protect.

2) Create an architecture overview: Use simple diagrams, Data Flow Diagrams (DFD), and tables to document the architecture of the application, including subsystems, trust boundaries, and data flow.

3) Decompose the application: Decompose the architecture of the application, including the underlying network and host infrastructure design, identify potential vulnerabilities in the design, implementation, or deployment configuration of your application.

4) Identify the threats: Keeping the goals of an attacker in mind, and with knowledge of the architecture and potential vulnerabilities of the application, identify the threats that could affect the application. Throughout this step, each threat is categorized as follows: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, and **E**levation of privilege (STRIDE).

5) Document the threats: Document each threat using a common threat template that defines a core set of attributes to capture the threat.

6) Rank the threats: Rank the threats to prioritize and address the most significant threats first. These threats present the biggest risk. The ranking process weighs the probability of the threat against damage that could result should an attack occur. It might turn out that certain threats do not warrant any action when the risks posed by the threat are compared with the resulting mitigation costs. Each threat is then ranked based on **D**amage potential, **R**eproducibility, **E**xploitability, **A**ffected users, and **D**iscoverability (D.R.E.A.D.).

Microsoft has developed a free tool to perform the threat modeling process. The latest version of the tool is Microsoft Threat Modeling Tool 2014. The Threat Modeling process starts with creating a data flow diagram (DFD) model for the software system. A DFD includes the following elements: data flows, data stores, processes, interactors, and trust boundaries. Figure 1 shows an example DFD drawn in Microsoft Threat Modeling Tool 2014. Each highlighted number in Figure 1 represents: 1) Interactor with the system – a human or another software component. 2) Trust Boundary – defense design/implemented to mitigate malicious attacks. 3) Data Flow – shows the direction of data in the system. 4) Process – login or any other functionalities. 5) Data Store – where information is stored in the system.

Each of the DFD elements is susceptible to different STRIDE of threat types. The elements of a DFD and the STRIDE threat types that affect the elements are described in Table 1 [7, 8].

The DFD elements are susceptible to threats when a trust boundary goes across a data flow. If the trust boundaries are not placed where they should be, then they may generate wrong potential threats.

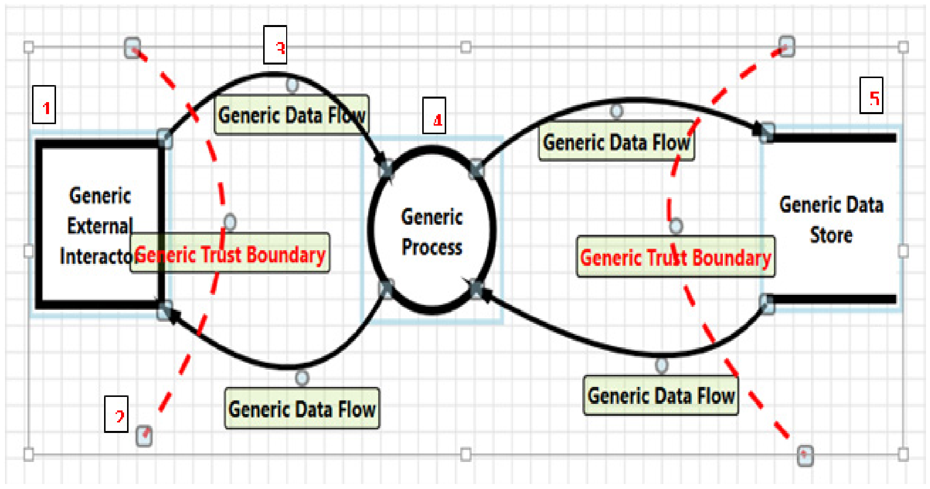


Figure 1. An example of a DFD in Microsoft Threat Modeling Tool 2014

STRIDE Types / DFD Elements	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Data Flows		✓		✓	✓	
Data Stores		✓		✓	✓	
Process	✓	✓	✓	✓	✓	✓
Interactors	✓		✓			

Table 1. STRIDE Threat types affecting DFD elements in Figure 1

3. Evaluating the Effectiveness of Microsoft's Threat Modeling Tool

Secure Software Engineering (COMP727) is a graduate level online course that teaches students how to incorporate security throughout the SDLC. The main topics include: threats to software, software vulnerabilities, risk management, security requirements, secure design principles and patterns, an overview of secure programming and security testing. The prescribe textbook for the course is Software Security: Building Security In [9]. Chapter 5 of the textbook (Architectural Risk Analysis) [9], was provided as reading material for the students. Furthermore, Chapter 5 introduces students to architectural risk analysis approaches, that would help them to understand why architectural risk analysis is important, and informs them about how to rank risks/threats and mitigate the risks/threats. Three architectural risk analysis approaches were introduced in Chapter 5, namely: Attack Resistance Analysis, Ambiguity Analysis and Weakness Analysis. Of the three risk analysis approaches, Microsoft Threat Modeling can be considered as the Attack Resistance Analysis Approach [9].

After the students read Chapter 5 of [9], they were introduced to Microsoft Threat Modeling [7, 8]. Students were then given an assignment to perform threat modeling. The objectives of the assignment are for students to identify threats, evaluate the impact of potential threats, and come up with mitigation strategies. In order to evaluate the effectiveness of the Microsoft Threat Modeling Tool with assisting non-experts (students in this case), the assignment was given in two parts. The first part of the assignment, Part A, was to perform manual threat modelling on a mock online shopping web application (tunestore) that was already developed. Part A of the assignment was focused on students identifying potential threats without any step-by-step guidance. The second part of the assignment, Part B, was carried out using the Microsoft Threat Modeling Tool 2014 that was used to compare to the results of the Part A. In Part B, students had a more guided approach to identifying threats. Additionally, in both parts of the assignment, the students were modeling threats using STRIDE as described in section 2, however, steps 6 in the process [7, 8] was omitted from this assignment.

To minimize the chance of students using the Microsoft Threat Modeling Tool 2014 for part A of the assignment, Part B was given to the students after they completed and submitted Part A. Additionally, there are two assumptions as to why students were given Manual Threat Modelling: 1) it is assumed that students would do better in Part B than Part A, since Part B is a guided approach, 2) the results in Part A would help to validate Part B results.

The main set of functionalities of tunestore are : Customer logs in, Customer buys CDs, Customer comments about CDs, Customer adds friends, Customer views friends, Customer adds a balance to their account using their credit card number, Customer logs out, and Customer sends CD as a gift to a friend.

A developed and functional mock web application was given to the students in this assignment. Our assumption was that having developed software would allow students to interact with the software, which would aid them in identifying potential threats more easily. Also, since the students are in a learning environment, a university course, the developed application would help with learning the activities threat modeling. These activities includes dissecting functional components, analyzing the components at every point of entry, and trace how data move through each functionality to identify any security flaws.

3.1 Part A: The Manual Threat Modeling Process

In Part A, the students were given the tunestore web application and instructed to come up with 10 potential threats that could affect tunestore. For each of the threats identified, the students were also asked to give a brief description of the threat, describe the impact (business and technical) of the threat, and describe how to mitigate the potential threats. To carry out this part of the assignment, they were given specific instructions that did not involve the assistance of a tool. Students were given preparatory reading materials such as information about DFD elements, information about each threat type in the Microsoft STRIDE model and reference materials to understand Threat Modeling [7, 8].

Once the students were finished with part A, they were also asked to describe the challenges they found with completing this assignment (i.e., conducting manual threat modeling process).

3.2 Part B: Microsoft Threat Modeling Tool 2014

In Part B, the students were asked to conduct threat modeling using the Microsoft Threat Modeling Tool 2014 on tunestore application. Before starting Part B, specific instructions were given to the students on how to use the Microsoft Threat Modeling Tool 2014, such as a demo video about the tool by Microsoft [10] was provided to help the students understand how to use the tool, instructions about how to download the tool to their a Windows computers, and additional instructions to use the tool for the threat modeling. Similar to Part A, students were asked to come up with 10 potential threats that could affect tunestore. However, since the tool would give the list of potential threats, students were asked to list their top ten potentials threats. For each of the threats identified, the students were also asked to give a brief description of the threat, describe the impact (business and technical) of the threat, and describe how to mitigate the threat. Students were also asked to compare the threats they found in Part A and Part B. Additionally, the students were asked to evaluate Microsoft SDL threat modeling tool by answering the following questions:

1) What challenges did you find with using the Microsoft SDL Threat Modeling Tool 2014?

2) What do you like about Microsoft SDL Threat Modeling Tool 2014?

3) What could be done about Microsoft SDL Threat Modeling Tool 2014 to make it better?

4. Evaluation of Results

This section provides, the students' work from part A and part B are compared, and the feedback from students on the manual process of threat modeling and Microsoft Threat Modeling Tool 2014. The next section discussed the results.

4.1 Comparison of students' work from Part A and Part B

Comparing students' work from Part A and Part B, i.e., the threats identified and ranked in Part A, and Part B, the following can be observed:

1) Most of the students identified threats of each STRIDE threat type and were able to give a good description of the potential threats to tunestore in Part A and B.

2) For the potential threats identified, students gave good mitigation methods to prevent the threat from materializing in Part A and B.

3) When it comes to the impacts of the threats, most of the students mentioned technical impacts instead of business impacts in Part A and B.

4) The students' work showed some of the more general threats that could affect tunestore in Part A compared to part B. In Part B some students mentioned that their potential threats in Part A were a subset of those identified in Part B. From observation of the students' work in Part B, a subset of the potential threats generated by the tool were also present in Part A. However, since the students were asked to only identify 10 potential threats (due to time constraints on the project), the students might not have reported all the threats that were likely to materialize. Most students mentioned information disclosure for the authentication process in Part A, but only one student was able to identify that password reset process/functionality information disclosure in the Part B. The password reset process was present in that student's DFDs for both Parts A and B. Another student was able to identify cross-site scripting in Part B that was not stated in Part A of the same student's work.

5) Some of the DFDs created for the tunestore application were at different levels. Students had difficulties coming up with the DFD at the right level. If the DFD is not at the right level, the list of potential threats may vary and possibly be incorrect.

6) Some of the students work in Part A did not have all of the potential threats related to the STRIDE types, but the students were able to identify all the STRIDE threat types when they used the tool in Part B. Additionally, one student mentioned that the tool helped to reassured some of the potential threats that were identified in Part A, and helped to clarify which STRIDE types the threats to tunestore belong to base on the functionality. A reassure for the reassurance of the STRIDE types could be how the tool enumerated threats on each DFD element.

7) One student mentioned that a previous project (Applying Risk Management Framework) helped with determining the impact of each threat found, and determining which threats should be mitigated.

8) Students were able to identify the main functionalities of tunestore. However, students generated different DFDs because they each have a different understanding of the functionalities of the tunestore application, and how data flows through tunestore application.

9) Overall, the students did better in Part B than Part A. Since the tool enumerated potential threats, students were able to identify threats that were related to each STRIDE type. For example, the 'purchase CD' and 'CD as a Gift' functionalities are separate data flows going to the same data store. The tool would enumerate the same potential threats for them, such as repudiation and escalation of privileges. The name a student gives to a process does not affect the potential threats that would be listed by the tool. The potential threats are given based on the interaction between the DFD elements. So, if students look through the potential threats enumerated by the tool for each data element, they may find threats that they were not able to find in Part A. In this example, repudiation was identified in Part B and not in Part A.

4.2 Students' Feedback on the Manual process of threat modeling

Most students were not able to completely understand the DFD elements to be used. Also, many of the students had difficulties using the STRIDE technique to model threats because of the lack of prior knowledge on the STRIDE technique, DFDs, anticipated threats, impact analysis and the required level of complexity of the DFD.

4.3 Students' Comparison of Microsoft Threat Modeling Tool 2014 and manual process

All students noted that the SDL tool found threats that they had not identified as well as those that they did.

4.4 Students' Feedback on Microsoft SDL Threat Modeling Tool 2014

4.4.1 Challenges students had with the tool

A significant challenge for most of the students at the outset was learning DFD symbols/stencils, or knowing which DFD element to choose. However one student mentioned that the components available to choose from were limited and "*generic*". Another issue for some was that there were not enough specific instructions about which processes and data flows to use.

4.4.2 Features that students liked about the tool

The students liked that the tool gave error that guided them when drawing their diagrams. The students also liked that the tool described and categorized the threats, stated design flaws and even gave mitigation suggestions.

4.4.3 Students suggestions for improving the tool

The students came up with the following suggestions to improve the tool: 1) The tool could provide better descriptions of the stencils (DFD elements) 2) The tool could have the functionality to help users to prioritize the threats identified and 3) The tool could provide more hints on the DFD diagram errors.

5. Discussion

The results from the students' work showed a collective improvement in Part B over Part A. However, this tool evaluation study has the following limitations:

1) The number of students that participated in the study was small. There were twenty students that completed the assignment.

2) Students came up with different DFDs. It is not clear whether the variations of DFDs could have had a significant effect on the potential threats found in part A and part B. Therefore it may be better to give students one DFD, and ask them to conduct threat modeling in Part A and B.

3) Students completed part B after completing part A. So their work in Part B may be influenced by their work in Part A.

The level of the DFD can affect the amount of potential threats identified. As the DFD gets more detailed, the number of potential threats are likely to increase. Students/developers may have to go through a long list of potential threats before deciding on the ones to mitigate.

The tool does not provide a ranking for each potential threat found. Since it is assumed that the students are novices to the Threat Modeling Process, it may be difficult for students to select the threats that are more likely to materialize and rank them appropriately.

Some of the DFD elements in the Microsoft Threat Modeling Tool 2014 give different potential threats. As a result, if students select different DFD symbols from each DFD symbol set, their results could be different. For example, using the generic data store instead of the SQL Database data store will give different results. The generic data store will have less potential threats compared to the SQL Database data store. The description of the potential threats using the SQL Database data store is more detailed than the generic data store. This could have a negative effect on how well a student determines the impact of the threat and the mitigation solutions for the threat.

Based on our evaluation results, we propose to do the following in future teaching and evaluation of the Microsoft Threat Modeling tool:

1) In the course, students were first given a project on Applying Risk Management. This project asked students to analyze risks, mitigate risks, and validate mitigation strategy for a proposed business/project following the Risk Management Framework proposed by Cigital [9]. This topic is presented in the Chapter 2 of the textbook [9]. After some other topics were introduced to the students, and some other projects were given to the students, the topic of architectural risk analysis was introduced to the students, and the project on threat modeling was given to the students. It may be beneficial to introduce the students to the topics of threat modeling right after the topic of Risk Management Framework was introduced. Students could be given the project of threat modeling right after the Risk Management Framework project. This way, the students will be more familiar with the steps for risk analysis. It will also help students apply the knowledge gained from Risk Management Framework project to the threat modeling project.

2) It has been observed that some of the DFDs created by students in both part A and part B of the threat modeling project did not represented the entire information flows of the tunestore application. Therefore it would be necessary to provide more information on how to draw DFDs. Also, where trust boundaries were placed in a DFD, the list of potentials threats generated by the tool was affected. Therefore, information should be provided prior to the assignment to help students draw good DFDs that represent the application with trust boundaries placed at appropriate locations. Additionally, the abstraction level of the DFDs can affect the amount of potential threats generated by the tool. A DFD at a too detailed level would overwhelm students with too many potential threats. It will be good to help students to draw DFDs at a level suitable for the objectives of this assignment, i.e. a level that's not too high, and yet not too detailed.

3) Instead of limiting the number of potential threats to 10, students could be asked to list all the potential threats that would be applicable to the application. This way we could get an even better understanding of how well the tool could assist the students with threat modeling compared with manual process.

4) The instructions of the assignment could be improved by making them more specific. Some students grouped the threats by the functionalities of the application in part A of the project. For example, the login process – potential threats: spoofing, tampering, information disclosure and elevation of privilege. The instructions of the assignment could explicitly ask students to identify threats of STRIDE types according to functionality.

5)In a software security testing course in the Fall 2014 semester, students did security testing on the tunestore application and found different type of vulnerabilities. Some of these vulnerabilities were: cross-site scripting in the comment on CD field, cross-site scripting in the URL parameter, SQL Injection in the login page, register customer, and password reset. Very few of these potential attacks showed up in the threat modeling results in COMP727 in Spring 2015. Based on this observation, it may be beneficial to ask students to identify potential threats for all the functionalities in Part A and Part B of the threat modeling assignment, based on suggestion 4. This may help us better understand whether students could come up with more specific threat types compared with using manual process.

6) Since Microsoft threat modeling tool 2014 does not assist with the ranking of potential threats, and students had difficulty with this task, students could be instructed to rank the threats using the D.R.E.A.D approach.

7) Add instructions on how to select the appropriate DFD elements in the Microsoft threat modeling 2014 tool since the DFD element could affect the potential threats generated by the tool.

6. Conclusion

It is important to integrate best practices for developing secure software into the software development lifecycle, so that software continues to function correctly under malicious attacks. Students in software security need a guided approach that would help them become more aware of the software security threat that should be addressed early in the software development lifecycle through threat modeling.

Risk analysis including architectural risk analysis, threat modeling are best practice for developing secure software. Microsoft threat modeling is an architectural risk analysis method and was introduced to students in a secure software engineering course at our university. Microsoft's SDL threat modeling tool 2014 was developed to assist software engineers to conduct architectural risk analysis. Through a threat modeling assignment, we evaluated the effectiveness of Microsoft SDL threat modeling tool 2014 in assisting software engineers, particularly non-experts or students to conduct threat modeling by comparing it with threat modeling without using the tool.

Our study shows that students as a whole improved their work on threat modeling with the tool compared with not using the tool. The abstraction level of the DFD affects the amount of potential threats identified. Selecting different DFD elements in the SDL threat modeling tool will also generate different threats. Future work includes improving the teaching and learning of threat modeling by improving the assignment instruction, and providing instruction on drawing DFD. Future evaluation of Microsoft's SDL threat modeling tool could be improved by providing a common DFD diagram for the students, and not limiting the number of threats identified by the students.

7. Acknowledgments

This work is partially supported by National Science Foundation under the grant HRD-1137516. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] McGraw, G., & Viega, J. (2001). Building Secure Software. Addison Wesley.

[2] McGraw, G. Software Security. IEEE Security & Privacy. 2004

[3] Howard, M. (2004). Building more secure software with improved development process. IEEE Security & Privacy, 2(6):63–65.

[4] Ardi, Shanai, et al. (2007). How can the developer benefit from security modeling? Availability, Reliability and Security, ARES . The Second International Conference on. IEEE.

[5] Möckel, Caroline, Ali E. Abdallah. (2010). Threat modeling approaches and tools for securing architectural designs of an e-banking application. Information Assurance and Security (IAS), Sixth International Conference on. IEEE.

[6] AlBreiki, Hamda Hasan and Qusay H. (2004). Mahmoud. Evaluation of static analysis tools for software security. In Innovations in Information Technology (INNOVATIONS), 10th International Conference on, p. 93-98. IEEE.

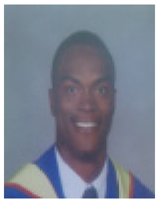
[7] Chapter 3 Threat Modeling, retrieved on March 20, 2015 from http://msdn.microsoft.com/en-us/library/ff648644.aspx

[8] Uncover Security Design Flaws Using The STRIDE Approach, retrieved on March 20, 2015 from http://msdn.microsoft.com/en-s/magazine/cc163519.aspx#S3

[9] McGraw, Gary. Software Security: Building Security In. 1st edition. Addison-Wesley Professional, 2006, pp. 150-179

[10] Threat Modeling Tool 2014 Demo, retrieved from https://www.youtube.com/watch?v=G2reie1skGg

9. Author Biographies



Imano Williams was born in St. Catherine, Jamaica. He received his B.Sc. degree in Computer Science & Electronics (Double Major) from The University of The West Indies, Jamaica in 2012. He then worked as a Software Quality Assurance Engineer before pursuing his MS in Computer Science at North Carolina Agricultural & Technical State University, Greensboro, North Carolina, United States of America in 2014. His current research interests include software security education and usable security.

Xiaohong Yuan was born in China. She received her PhD in Computer Science from Florida Atlantic University, Boca Raton, Florida, USA in 2000. She is currently a Professor in the Department of Computer Science, and Director of Center for Cyber Defense at North Carolina Agricultural and Technical State University, Greensboro, North Carolina, USA. Her research interests include software security, health informatics security and privacy, mobile security, and information assurance education.