Developing Abuse Cases Based on Threat Modeling and Attack Patterns

Xiaohong Yuan*, Emmanuel Borkor Nuakoh, Imano Williams, Huiming Yu
Department of Computer Science, North Carolina A&T State University, 1601 East Market St., Greensboro, North Carolina, USA.

* Corresponding author. Tel.: 1(336)2853700; email: xhyuan@ncat.edu.
Manuscript submitted February 20, 2015; accepted April 20, 2015.
doi: 10.17706/jsw.10.4.491-498

Abstract: Developing abuse cases help software engineers to think from the perspective of attackers, and therefore allow them to decide and document how the software should react to illegitimate use. This paper describes a method for developing abuse cases based on threat modeling and attack patterns. First potential threats are analyzed by following Microsoft's threat modeling process. Based on the identified threats, initial abuse cases are generated. Attack pattern library is searched and attack patterns relevant to the abuse cases are retrieved. The information retrieved from the attack patterns are used to extend the initial abuse cases and suggest mitigation method. Such a method has the potential to assist software engineers without high expertise in computer security to develop meaningful and useful abuse cases, and therefore reduce the security vulnerabilities in the software systems they develop.

Key words: Abuse case, threat modeling, attack patterns, secure software development.

1. Introduction

To secure cyberspace, it is critical to engineer secure software. Security-related activities and deliverables need to be integrated into each of the phases of software development life cycle [1]-[3]. One of the security-related activities is to develop abuse or misuse cases. Abuse case is a use case from an attacker perspective with the intent to harm the system [4]. An abuse case might harm an actor of the system, a stakeholder or the system itself [5]. Abuse cases threaten use cases and serve as a support for developers to elicit security requirements. Developing abuse cases allow software engineers to think from the perspective of attackers, and decide and document a priori how the software should react to illegitimate use [6]. Countermeasures can be developed to mitigate misuse cases in the form of security use cases [7].

Hope, McGraw & Anton [8] suggested that abuse cases can be created through informed brainstorming. However, high expertise and experience in security is required to produce meaningful and useful abuse cases using brainstorming method. It has also been suggested that abuse cases be developed based on a set of requirements and standard use cases, and a list of attack patterns [6]. However, specific processes for developing abuse cases are lacking. This paper describes a method for developing abuse cases based on threat modeling and attack patterns. Such a method allows software developers who do not have high expertise and experience in security to develop abuse cases by following specific steps.

The rest of the paper is organized as follows. Section 2 provides background information on threat modeling and attack patterns. Section 3 describes the proposed method for developing abuse cases based on threat modeling and attack patterns. Section 4 illustrates the proposed method with an example. Section 5 concludes

the paper.

2. Background

2.1. Threat Modeling

Threat modeling is a process proposed by Microsoft for identifying and ranking risks to architecture and design level artifacts [9], [10]. It follows the process of hypothesizing potential security threats, evaluating the threats, ranking the threats and suggesting mitigation strategies. Security threats are classified into five general categories: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege (STRIDE). The security threats identified in the system are ranked based on their Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability (DREAD) [9], [10].

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, interators, and trust boundaries. Each of the elements is susceptible to a set of threats. The elements of a DFD and the STRIDE threat types that affect the elements are listed below [10]:

1) Data Flow: Tampering, Information Disclosure, Denial of Service
2) Data Stores: Tampering, Information Disclosure, Denial of Service
3) Processes: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
4) Interactors: Spoofing, Repudiation

Based on Microsoft's threat modeling process, Microsoft Security Development Lifecycle (SDL) threat modeling tool is provided as a free tool to assist software developers to identify potential threats and suggest mitigation strategies [11], [12]. With the tool, a developer first draws a data flow diagram (DFD) model depicting the design of the software system. The tool provides guidance and feedback in drawing the model. The tool then guides the developer to analyze threats and mitigations according to STRIDE framework for each element of the DFD model. Finally the tool generates a report for the threat model.

2.2. Attack Patterns

Derived from the concept of design patterns, attack patterns describe common methods for exploiting software. Attack patterns capture and communicate the attacker's perspective, which can help software developers to think like an attacker. The following information is typically included in an attack pattern: Pattern Name and Classification, Attack Prerequisites, Description, Related Vulnerabilities or Weaknesses, Method of Attack, Attack Motivation-Consequences, Attacker Skill or Knowledge Required, Resources Required, Solutions and Mitigation, etc. [13]. Hoglund & McGraw [14] described 49 attack patterns. The Common Attack Pattern Enumeration and Classification (CAPEC) repository includes 463 publicly available attack patterns along with a comprehensive schema and classification taxonomy [15].

Sethi & Barnum [16] illustrated that attack patterns have the potential to be used in each phase of the secure software development life cycle. Gegick and Williams [17] constructed attack patterns based on existing vulnerability databases using regular expressions and used these attack patterns for identifying security vulnerabilities during software design. Pauli and Engebretson [18] proposed a software tool to retrieve related CAPEC attack patterns based on system prerequisite in order to use the mitigation strategies for the retrieved attack patterns during system design and implementation.

3. The Proposed Method for Developing Abuse Cases

We propose a method for developing abuse cases based on Microsoft's threat modeling and attack patterns as show in Fig. 1.

The steps in Fig. 1 are explained below:

1) Develop use cases for the system.

2) Draw DFD for the system which implements the use cases.
3) Identify potential threats for each element in the DFD following Microsoft's threat modeling approach. Microsoft SDL threat modeling tool can be used for threat modeling.
4) Identify abuse cases based on the threats identified in step 3). At this stage, only the names of the abuse cases, and the goal of each abuse case are identified.
5) Retrieve attack patterns that are relevant to the abuse cases. CAPEC attack patterns as well as other attack pattern sources can be utilized. Attack patterns can be retrieved based on keywords.
6) Use the retrieved attack patterns to extend the abuse cases identified in step 4). The attack execution flow information in an attack pattern can be used to derive the "abusive interaction" section of an abuse case. Relevant attack patterns may also allow developers to see different ways an abuse case can be realized, i.e., sub abuse cases can be derived.

4. Illustrating the Proposed Method with an Example

In this section, an example is used to illustrate the proposed method for developing abuse cases. The example system is a health information system (HIS) that keeps track of a patient's information, appointments, appointment findings, prescriptions, lab results, etc.

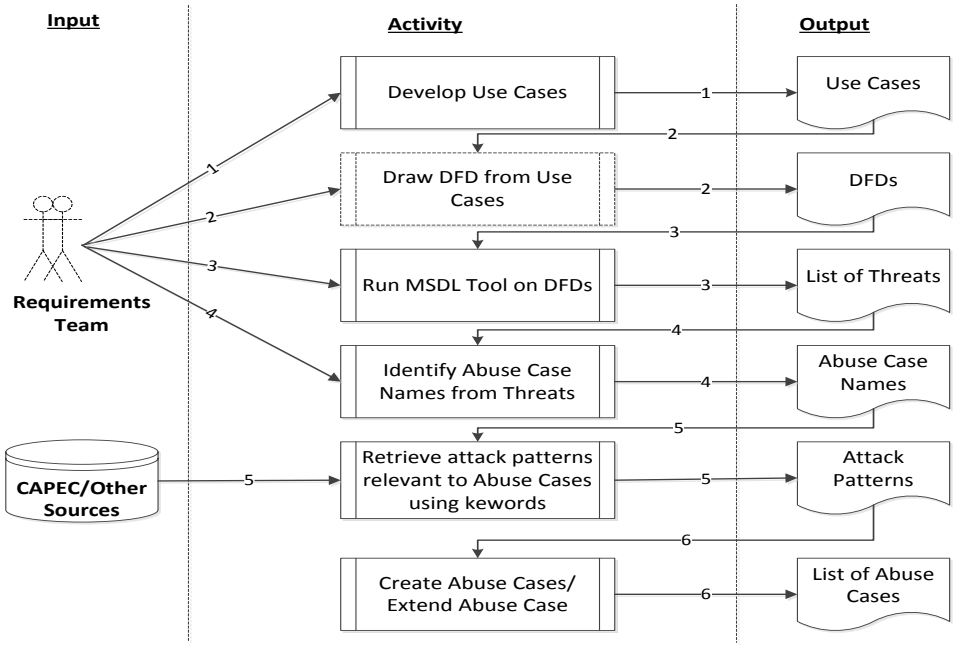

Fig. 1. The method for developing abuse cases based on Microsoft threat modeling and attack patterns.

4.1. Developing Abuse Cases

The HIS has many users which may include secretaries, nurses, doctors, pharmacists, IT personnel, business office personnel and administrative personnel [19]. This system can include the following use cases: 1) Secretary entering patient information; 2) Nurse entering preliminary appointment information; 3) Doctor entering appointment findings; 4) Doctor transmitting pharmacy orders to the pharmacy; 5) Pharmacist receiving pharmacy order. For the purpose of illustrating the proposed method, only the doctor's role of entering appointment findings is considered. Abuse cases will be developed for the use case "Enter Appointment Findings". The description of this use case is as follows:

The doctor logs into HIS server using a secure browser. The server authenticates the Doctor and opens a session for him. The Doctor enters patient appointment findings and then logs out.

4.2. Drawing DFD

Next, the DFD of the HIS is drawn. For simplicity, only the part of the DFD that implements the use case "Enter

Appointment Findings" is drawn (see Fig. 2).

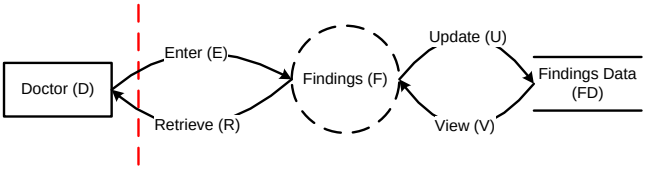

Fig. 2. DFD for the use case "enter appointment findings".

In Fig. 2, the doctor has the ability to enter or retrieve appointment findings after he is successfully authenticated and authorized (the dashed vertical line represents authentication and authorization) by the system. The Findings process takes the data from doctor, processes it and sends it to the findings data store. The arrows show the data flow between the doctor, the Findings process and the data store

4.3. Threat Modeling

Following Microsoft's threat modeling process, security threats are analyzed for each elements of the DFD. For the elements shown in Fig. 1., the threats each element is susceptible to is shown in Table 1.

Table 1. Threats Affecting Elements in Fig. 1.

Element	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Data Flows		X		X	X	
Findings Data Stores		X		X	X	
Findings Process	X	X	X	X	X	X
Doctor	X		X			

4.4. Identifying Abuse Cases

Based on the threats listed in Table 1 that affect each elements of the DFD, abuse cases can be identified. The abuse cases identified are explained below:

1) The "Finding Process" is susceptible to spoofing, that is, an attacker can implement a fake "finding process" and use it to replace the legitimate one. The doctor may be tricked to enter appointment findings to the fake "Finding Process". Therefore, a "Spoof Finding Process" abuse case is identified. The "Doctor" is also susceptible to spoofing; therefore, an "Impersonate Doctor" abuse case is identified.
2) The "Data Flows", "Findings Data Store", and the "Finding Process" are susceptible to tampering. Tampering all these elements will allow attackers to change the findings the doctor entered. Therefore a "Change Doctor's Findings" abuse case is identified.
3) The "Finding Process" and the "Doctor" are susceptible to Repudiation. A "Repudiate Entering Findings" abuse case is identified.
4) The "Data Flows", "Findings Data Store", and the "Finding Process" are susceptible to information disclosure. An "Intercept Packets" abuse case and a "View Findings without Authorization" abuse case are identified.
5) The "Data Flows", "Findings Data Store", and the "Finding Process" are susceptible to denial of service. A "Make Finding Process Unavailable" and a "Corrupt Findings Data store" abuse case can be identified.
6) The "Finding Process" is susceptible to elevation of privilege; therefore a "Non-doctor Enter Appointment Findings" abuse case is identified.

Fig. 3 shows the abuse cases and the elements of the DFD they affect.

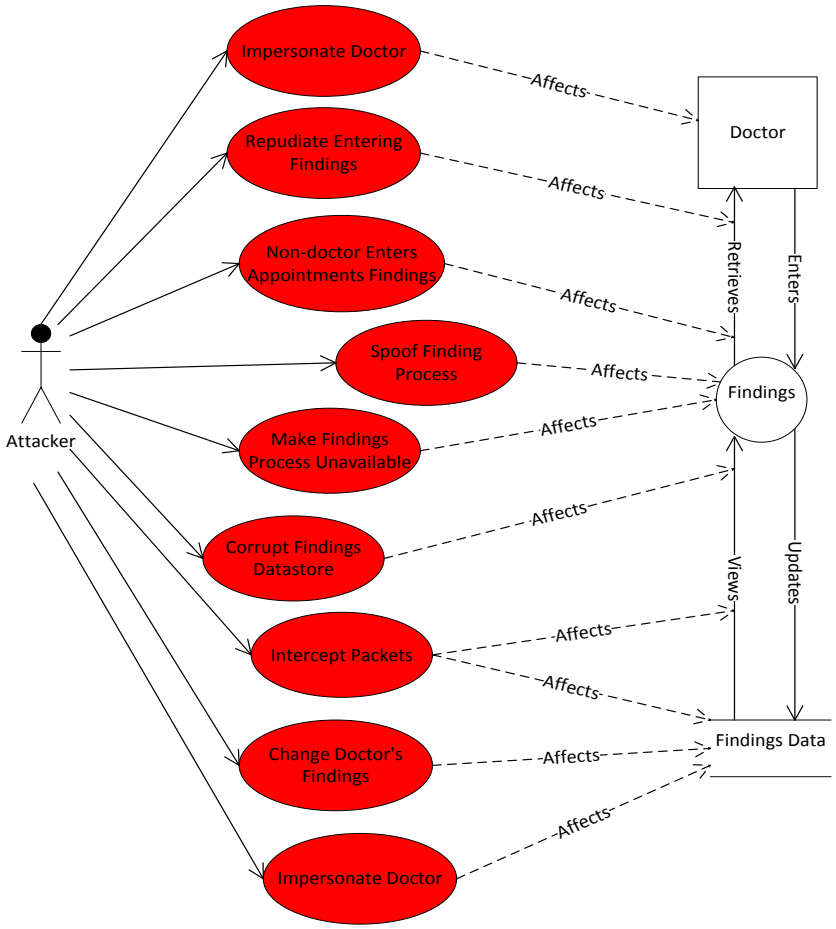

Fig. 3. Abuse cases and the elements of the DFD they affect.

4.5. Retrieving Relevant Attack Patterns

Even though abuses cases were generated based on threat models, how are these abuse cases realized is not clear. For a software engineer without high security expertise, it is challenging to describe the interactions of the actor and the system for the abuse cases. We propose to utilize the knowledge provided in attack patterns to assist in this step. Information in attack patterns relevant to the abuse case being considered will be used to develop the description of abusive interaction of the abuse case. There are different methods to realize an abuse case, these different methods can be described as sub abuse cases.

In our example, we retrieve relevant CAPEC attack patterns and use the information in these attack patterns to generate abuse case description and/or find sub abuse cases. Relevant attack patterns can be retrieved through keywords. For example, for the "Impersonate Doctor", we can use the keyword "impersonate" to search CAPEC attack pattern catalog. This search by keyword function is provided by CAPEC website. CAPEC search function returns several pages of results. The results from the first page are examined and the most relevant attack patterns are listed below:

1) CAPEC – 98: Phishing
2) CAPEC-218: Spoofing UDDI/ebXML Messages
3) CAPEC-151: Identity Spoofing
4) CAPEC-415: Pretexting via Phone
5) CAPEC-102: Session Sidejacking
6) CAPEC-21: Exploitation of Session Variables, Resources IDs and other Trusted Credentials
7) CAPEC-272: Protocol Manipulation
8) CAPEC-194: Fake the Source of Data

Reading the description of these attack patterns, it can be reasoned that CAPEC-218 is not relevant because

UDDI/ebXML Messages are not used in the example HIS system. CAPEC-415 is not relevant because it is a social engineering attack. The rest of attack patterns can be studied in detail to find whether they provide information to provide detailed description of an abuse case.

4.6. Using the Relevant Attack Patterns to Extend Abuse Cases

As an example, CAPEC-21 is selected to extend the abuse case "Impersonate Doctor". Exploitation of session variables is one way of impersonating the doctor role. Therefore a sub abuse case "Impersonate Doctor through Session Exploitation" is derived. Using the information in the "Description" and "Example Instance", the sub abuse case "Impersonate Doctor through Session Exploitation" can be described as follows:

Name: Impersonate Doctor through Session Exploitation

Objective: To impersonate the doctor and change appointment findings.

Prerequisites: Server software must rely on weak session IDs proof and/or verification schemes

Abusive Interaction:

An attacker fetches many samples of a session ID. This may be through legitimate access (logging in, legitimate connections, etc), systematic probing, or eavesdropping.

An attacker repeatedly attempting to query the system with a spoofed session header in the HTTP request.

Post Condition: Attacker assumes the identity of a doctor.

Similarly, other sub abuse cases for "Impersonate Doctor" can be found and described, for example, "Impersonate Doctor through Phishing", "Impersonate Doctor through Session Sidejacking".

The mitigation strategy information in the CAPEC attack patterns can be used to inform the development of security requirements and design of the system.

5. Conclusion

This paper describes a method for developing abuse cases based on Microsoft's threat modeling and attack patterns. An example health information system is used to illustrate the process. CAPEC attack patterns are used in the proposed method, though other attack pattern libraries can also be used in the proposed method. This method leverage the knowledge base of Microsoft threat modeling and attack patterns with the goal of enabling software engineers, especially those without high expertise in computer security to develop meaningful and useful abuse cases, and develop secure software.

The proposed method has the following limitations:

1) Based on the initial abuse cases, the software developer uses keywords to search for relevant CAPEC attack patterns. The quality of the keywords will affect the relevance of the attack patterns being retrieved. Further research needs to be done to investigate how to generate keywords that will retrieve the most relevant attack patterns.
2) When attack patterns are retrieved from CAPEC library, the software developer needs to judge which ones are most relevant. Sometimes a lot of attack patterns are retrieved, it will be time assuming to examine each one to determine which ones apply to the current situation, and can be used to extend the abuse cases. Our future work includes designing and implementing a mechanism to rank the relevance of the retrieved attack patterns.
3) The method illustrated here is mostly a manual process. Our future plan is to design and implement a tool that automates part of the process to make it easier for software engineers to use this method in their software development.
4) Though an example is used to illustrate the proposed method, the effectiveness and efficacy of the proposed method need to be further researched. We plan to ask students in a software engineering class at our university to follow the proposed method to develop abuse cases as a way to evaluate the proposed method.

Acknowledgment

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] Howard, M., & Lipner, S. (2006). *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software*, Microsoft Press.

[2] Build security in: Setting a higher standard for software assurance. Retrieved February 20, 2015, from https://buildsecurityin.us-cert.gov/bsi/home.html

[3] CERT software assurance, software engineering institute, carnegie mellon. Retrieved February 20, 2015 from http://www.cert.org/work/software_assurance.html

[4] Alexander, I. (2003). Misuse cases: Use cases with hostile intent, *Software*, 58-66.

[5] McDermott, J., & Fox, C. (1999). Using abuse case models for security requirements analysis. *Proceedings of the Conference on 15th Annual Computer Security Applications* (pp. 55- 64).

[6] Tndel, I. A., Jensen, J., & Rstad, L. (2010). Combining misuse cases with attack trees and security activity models. *Proceedings of the Conference on Availability, Reliability, and Security* (pp. 438-445).

[7] McGraw, G. (2006). *Software Security: Build Security*, Addison-Wesley Professionals.

[8] Hope, P., McGraw, G., & Anton, A. I. (2004). Misuse and abuse cases: getting past the positive, *Security and Privacy*, *2(3)*, 90 - 92.

[9] Chatper 3 threat modeling. Retrieved on February 20, 2015, from http://msdn.microsoft.com/en-us/library/ff648644.aspx

[10] Uncover security design flaws using the STRIDE approach. Retrieved on February 20, 2015, from http://msdn.microsoft.com/en-us/magazine/cc163519.aspx#S3

[11] SDL threat modeling tool. Retrieved on February 20, 2015, from http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx

[12] Microsoft threat modeling tool. Retrieved on February 20, 2015, from http://www.microsoft.com/en-us/download/details.aspx?id=42518

[13] Barnum, S., & Sethi, A. (2015). Introduction to attack patterns. Retrieved on February 20, 2015 from https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack/585-BSI.html

[14] Hoglund, G. & McGraw, G. (2004). *Exploiting Software: How to Break Code*. Boston, MA: Addison-Wesley.

[15] MITRE. (2015). Common attack pattern enumeration and classification. Retrieved on February 20, 2015, from http://capec.mitre.org/index.html

[16] Barnum, S., & Sethi, A. (2015). Attack pattern usage. Retrieved on February 20, 2015, from https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack/588-BSI.html

[17] Gegick, M., & Williams, L. (2005). Matching attack patterns to security vulnerabilities in software-intensive system designs, *Software Engineering Notes*, *30(4)*, 1-7.

[18] Pauli J. J., & Engebretson, P. H. (2008). Towards a specification prototype for hierarchy-driven attack patterns. *Proceedings of the First International Conference on Information Technology: New Generations*.

[19] Pauli, J. J., & Xu, D. (2005, April). Misuse case-based design and analysis of secure software architecture. In *Proceedings of the International Conference on Information Technology: Coding and Computing* (pp. 398-403).

Xiaohong Yuan was born in China. She received her Ph.D 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 the 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.

Emmanuel Borkor Nuakoh was born in Bibiani, Ghana. He received his B.Sc. degree in geological engineering from the University of Mines and Technology, Ghana in 2011. After which he furthered his education in the United States and received his M.S. degree in computer science from North Carolina Agricultural & Technical State University, Greensboro, North Carolina in 2014. Emmanuel is currently in the Ph.D program in the Department of Computer Science at North Carolina Agricultural and Technical State University. His current research interests include software security, information privacy and security and cloud security.

Imano Williams was born in St. Catherine, Jamaica. Imano received his B.Sc. degree in computer science and 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 degree in computer science at North Carolina Agricultural and Technical State University, Greensboro, North Carolina, United States of America in 2014. His current research interests include software security, usable security.

Huiming Yu was born in China. She received her Ph.D in computer science from Stevens Institute of Technology, Hoboken, New Jersey, USA in 1992. She is currently a professor and the director of graduate study in the Department of Computer Science, North Carolina Agricultural and Technical State University, Greensboro, North Carolina, USA. Her research interests include software engineering, visualization, web security, information security, web applications and cloud computing.