jMUCMNav: an Editor for Misuse Case Maps

Yashar Bizhanzadeh¹, Peter Karpati²

¹ Dept. of computer science and engineering - Chalmers University of Technology and University of Gothenburg, SE-412 96 Gothenburg, Sweden
bizhan@student.chalmers.se

² Dept. of Computer and Information Science, Norwegian University of Science and Technology, Sem Sælands vei 7-9, Trondheim, Norway
kpeter@idi.ntnu.no

Abstract. Business processes require a supporting technical architecture enabling their realization. The secure design of this architecture has key importance in the future of the business; however there are not many notations which consider security issues and architectures together. Misuse case map is a diagrammatical notation that supports the depiction of system architecture, together with security vulnerabilities and their mitigations. It also enables the representation of intrusion scenarios through exploit paths. The diagrams can become quickly complex with the arising details of the system and the intrusion, thus tool support is a big step forward for spreading the use of these notations. This paper introduces jMUCMNav; an editor for misuse case maps based on the well tested and widely used use case map editor.

Keywords: security requirements, misuse case maps, system architecture, editor

1 Introduction

Security is considered as a qualitative, anti-functional or non-functional requirement in software system development and is added as a part of software development life cycle right from the early requirement analysis phase. With the growing importance of secure systems, security requirement engineering (SRE) grows as a part of requirements engineering (RE). This new field feeds into RE by introducing different methods and techniques to cope with security issues, and supporting them by proper tools. Misuse case maps (MUCM) are one of these diagrammatical modeling techniques. This model connects security related considerations to the technical architecture of an enterprise system, thus providing a reliable base for business processes. This paper introduces an editor for MUCM called jMUCMNav. The following section discusses related SRE works. Section 3 introduces misuse case maps. Next, we look at different SRE supporting tools such as Surakasha, SQUARE, SeaMonster and jUCMNav as well as elaborate on our choice of the MUCM editor base. Section 5 presents and discusses the misuse case map editor. Finally, we conclude and point out future work.
2 Related Work

Elicitation methods for ordinary functional requirements usually are not enough for preparing a complete and consistent set of security requirement. Security researchers provide new methods to elicit security requirements in a systematic way. Existing SRE techniques use different approaches and focus on different viewpoints on designing secure software.

**Multilateral approaches** focus on the different security and privacy expectations of the stakeholders and try to solve the conflicts, which always happen between the expectations of these two groups. These approaches try to look at the system with the viewpoints of different stakeholders, find out the valuable asset for different group of stakeholders, make the security for all groups, and solve the security related conflict between these groups [6]. These approaches result multilateral security requirement analysis and some other methods like SQUARE (security quality requirements engineering methodology). SQUARE focuses on clear communication between stakeholders and security requirement engineers to extract security goals, develop proper artifact, and estimate the security risks by using techniques such as use cases, misuse cases and attack trees.

**UML-based approaches** utilize the notation of the Unified Modeling Language to define the security requirement. Misuse cases are introduced by Sindre and Opdahl [8] to elicit threats to use cases and the functionalities, which are not allowed to be performed by users. Lodderstedt et al. present another UML-based approach for distributed systems called SecureUML [9]. It makes a role-based access control for class components using a UML profile to fulfill the confidentiality and integrity goals. Another example of this group is UMLSec [10]. This approach focuses on the three core goals of confidentiality, integrity and availability to define the security for security critical systems.

**Other approaches** like goal oriented, problem frame-based, and risk analysis-based methods approach the security engineering from a different angle but they are less relevant to our discussion.

3 Misuse case maps

Misuse case maps (MUCMs) [2] present security issues from an architectural perspective. They combine perspectives from misuse case (MUC, see the next section) [8] and use case maps (UCM) [1, 3, 4, 5]. Misuse case maps address security requirements by focusing on vulnerabilities, threats and intrusions (inherited from MUCs), from an architectural point of view (inherited from UCMs). UCMs provide a combined overview of a software system’s architecture and its behaviour by drawing usage scenarios paths (aka use cases) as lines across boxes that represent architectural run-time components. The boxes can be nested to indicate hierarchies of components. The scenario paths are connected to the components they run across by responsibilities drawn as crosses.

Fig. 1 shows the MUCM notation. It extends the UCM notation with intrusions. The intrusions are represented by one or more exploit paths. These exploit paths cut
through vulnerable parts of the system. Each path starts with a triangle. If no damage happens, it ends in a bar or a “no entry” symbol. Otherwise, the exploit path ends in a lightning symbol. Exploit paths can be numbered as individual steps in a complex intrusion. The steps of an intrusion will usually be causally related, each one built on previous results. Responsibilities can be left out if they are not relevant for the overall intrusion.

The system may have vulnerable points (such as authentication responsibility) or components (such as components without up-to-date security patches), which are suspect to threats. Mitigations can help to counter the threats and appear in the MUCM as desired possibilities. This translates to security requirements later. Misuses are depicted by the exploit path’s crossing of a vulnerable point or part. Get, put (+) and remove (−) arrows can be used to show how an exploit path interacts with a component. An example of a put arrow is when the attacker (‘arrow starting from the exploit path’) installs (+) a sniffer program on one of the servers (‘arrow ends at the component’). (see [11] for more detail).

Fig. 1. MUCM notations.

Fig. 2 shows the first 5 steps of a bank intrusion reported in the literature [19, Chapter 7]. First, the intruder found an interesting bank by browsing a web site with organizations and IP ranges assigned. Next, he probed for further details about the IP addresses of the bank and found a server that was running Citrix MetaFrame (remote access software). He then scanned other networked computers for the remote access port to Citrix terminal services (port 1494). The attacker knew he might be able to enter the server with no password, as the default Citrix setting is “no password required”. He searched every file on the computer for the word “password” to find the clear text password for the bank’s firewall. The attacker then tried to connect to routers and found one with default password. He added a firewall rule allowing incoming connections to port 1723 (VPN).
Fig. 2. A misuse case map created by the jUCMNav editor visualizes the first half of a bank intrusion. The whole red line depicts the attacker’s footprint whereas the dashed black line shows the regular users’ activities. Arrows with plus sign show when the attacker puts/gets something from/to a component.
4 Overview of SRE Initiatives with Tool Support

The Surakasha security workbench [20] combines different methods in security requirement engineering and makes a similar mapping between the steps of functional requirement analysis and security requirement analysis as much as possible. It is an open source desktop application implemented in Java. Although it has a clean user interface, this software suffers from some serious deficiencies. It has a few bugs in saving and loading diagrams. Usability of the software is undesirable especially in working with attack trees. Since it has not sufficient technical documentation regarded the architecture, design and development extension and maintenance cost will be high for adopting this software. The quality of source code is quite poor with inappropriate packaging and insufficient comments and documentation.

The SQUARE methodology [13] was developed by cyber security lab in Carnegie Mellon University to support the nine-step Security Quality Requirement Engineering process. This process includes sub-processes and techniques to improve requirement identification, analysis, and specification. It also focuses on management issues associated with the development of good security requirements. SQUARE is a web application developed in Java, with MySQL database and Tomcat Apache server. It is not an open source project, so the source code is not available. It is free to use on local machine or online [14]. End user help documents seem perfect but there is no technical documentation for development. As a major difference compared to Surakasha, the SQUARE tool dose not prepare any facility and editor for making any special security related artifact such as use cases, misuse cases or attack trees. It should be considered as a managerial tool in security requirement engineering for increasing the quality of the process. In conclusion, it is not a fitting candidate for extension.

SeaMonster [15, 16] is a security-modeling tool initiated by SINTEF and has been developed as a part of SHIELDS project [17]. The SHIELDs project focuses on model-based detection and elimination of software vulnerabilities. SeaMonster contains two different editors for attack trees and misuse case diagrams. You can create and edit diagrams for both attack trees and misuse cases with related notations and perform related usual task such as loading and saving different diagrams. SeaMonster is an open source desktop application that is developed as an eclipse plug-in. Eclipse is an application platform, which acts as a host for different plug-ins with diverse functionalities. SeaMonster utilizes three different eclipse modeling frameworks consist of Graphical Modeling Framework (GMF), Eclipse Modeling Framework (EMF) and Graphical Editing Framework (GEF). It has a simple and easy user interface with acceptable usability and reliability supported by good end user documentation. The extension and maintenance cost is a bit unclear since it does not have enough technical document related to development. In conclusion, SeaMonster could be considered as a viable candidate.

jUCMNav (the Java Use Case Map Navigator, [18]) is developed as part of user requirement notation (URN). URN is used for elicitation, analysis, specification, and validation of the requirement. It consists of two complementary views, Goal-oriented requirement language (GRL) and use case maps (UCM). jUCMNav is simply focuses on modeling for UCMs and supports all use case maps notations. jUCMNav is an open source desktop application which developed as an eclipse plug-in. This is the
most reliable and usable tool among all. It shares the eclipse flexible user interface and professional features. The quality of the source code is accompanied by sufficient technical documentation related the architecture, used library and programming languages. On the other hand there are no low-level design documents such as class diagrams. It has different subproject under development showing its high potential in extension and grows. As the down side, we can mention the complexity of the code and libraries. In conclusion, jUCMNav is a viable candidate.

Table 1 shows the comparison of the preceding tools. Applications has been compared in some aspects which have the most effect on the decision making process to find the most suitable base platform for extension. Based on the previous discussions and the comparison, jUCMNav was chosen as the best candidate as a base for the MUCM editor.

### Table 1. Comparison of the tools in SRE.

<table>
<thead>
<tr>
<th>Application</th>
<th>Open source</th>
<th>Type of application</th>
<th>Programming language</th>
<th>Used libraries</th>
<th>Technical documents</th>
<th>Usability</th>
<th>Reliability</th>
<th>Portability</th>
<th>Code quality</th>
<th>Complexity of code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surakaha</td>
<td>Yes</td>
<td>Desktop</td>
<td>Java</td>
<td>Swing</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>SQUARE</td>
<td>No</td>
<td>web</td>
<td>Java</td>
<td></td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>SeaMaster</td>
<td>yes</td>
<td>Eclipse base</td>
<td>Java</td>
<td>EMF, GMF, GEF</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>jUCMNav</td>
<td>yes</td>
<td>Eclipse plugin</td>
<td>Java</td>
<td>EMF, GEF</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

### 5 An Editor for MUCMs

The jMUCMNav editor is a modeling tool for MUCMs in designing secure architectures for business processes. It is possible to draw an intrusion scenario using MUCM notations as drawing components. These notations are placed on the right hand side palette of the editor (see Fig. 3.) appearing as an eclipse palette after creating a new misuse case map diagram through the file menu. We categorized the notation in different drawers inside the palette based on their functionality.

One can drag and drop the elements of the notation from the palettes to the main editor screen, bind them together and move them on the screen. Every path consists of an end point, start point, linking edges and empty points. It is possible to change the shape of the path by dragging the empty point on the screen; furthermore you can attach other elements to the path in these empty points. The editor does not perform any syntactic or semantic check on how the elements are put together currently.

The jMUCMNav application [12] was developed as an eclipse plug-in so it can be installed and executed in any Eclipse. It is developed using java programming language, based on a model driven architecture. This plugin uses two main Eclipse frame-works, EMF (eclipse modeling framework) and GEF (Graphical Editing framework). EMF is used to facilitate the design and implementation of a model driven architecture. Model in jMUCMNav consists of the notations for misuse case maps [2], parent and child relationship among them and the hierarchy of the notations. The code generation facility of the EMF framework on the model gives us a good degree of the modifiability and extendibility for the software. Using this framework, we could focus on the model itself and avoid the implementation details.
jMUCMNav’s user interface inherits the professional interface of Eclipse. Eclipse is one of the most popular IDEs (integrated development environments), which have its own characteristics in user interface design. It is quite popular in its way with different useful UI pallets, views and perspectives. Since jUCMNav gets its UI features from eclipse, it inherits all these well tested features from eclipse user interface. As the first evaluation of the usability we can rely on the usability of the Eclipse and jUCMNav -which is the ancestor of jMUCMNav. Since we have changed only the notation of the jUCMNav, we can claim that we inherited all the usability features of the jUCMNav and Eclipse IDE.

6 Summary and Future Work

The paper presented an editor for the misuse case maps security-modeling technique. The technique was introduced, and the choices regarding the editor implementation elaborated. However, the usability, appropriateness and efficiency validation of jMUCMNav are missing. We plan to do attend to these matters in form of a student experiment in the future. JMUCMNav will be a part of a bigger workbench tool for the Hacker Attack Representation Method (HARM, [11]), which aims to be a security requirements engineering methodology using attacks as starting point.
References

14. https://squaretool.cylab.cmu.edu/Square/
17. http://www.shields-project.eu/
18. http://jucmnv.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome