Using the Common Criteria to Elicit Security Requirements with Use Cases

Michael S. Ware
Fairmont State University
Fairmont, WV 26554
mware@fairmontstate.edu

John B. Bowles
University of South Carolina
Columbia, SC 29208
bowles@engr.sc.edu

Caroline M. Eastman
University of South Carolina
Columbia, SC 29208
eastman@engr.sc.edu

Abstract

The Common Criteria is often too confusing and technical for non-security specialists to understand and therefore properly use. At the same time, it is essential that security critical IT products under development be validated according to such standards not after but rather during the software engineering process. To help address these issues, this paper presents an approach to eliciting security requirements for IT systems with use cases using Common Criteria methodologies. The approach involves using actor profiles to derive threats, mapping derived threats to security objectives, and mapping objectives to security requirements using a CC Toolbox data set. Our aim is to ensure that security issues are considered early during requirements engineering while making the Common Criteria more readily available to end-users in an understandable context. Violet, an open source UML diagram modeling tool, has been extended to implement the approach from a use case textual description perspective.

1. Introduction

To address the need for establishing a world with robust software, one recommendation by the 2004 Security Across the Software Development Lifecycle Task Force is to educate and train developers to put security at the foundation of the software development process [9]. Use case developers are uniquely positioned to adhere to this recommendation. While specifying desired services and expected user interaction in requirements, use case developers can also analyze system security by considering unanticipated user actions and unintended system behaviors.

Previous work suggests that use cases have become increasingly common during requirements engineering but offer limited support for eliciting security threats and requirements [8]. As a result, misuse cases [8], abuse cases [10], and security use cases [5] have all been proposed as methods for specifying security threats, providing assurance arguments during design and testing, and specifying security requirements, respectively. Although these approaches are certainly useful, there is still a need to integrate security into use cases using standards such as the Common Criteria (CC) [2].

The CC is an international standard to be used as a basis for evaluating the security properties of information technology (IT) products or systems. Since use cases focus on the behavior of various actors, describing a system under design in ways similar to the demands of the CC can help identify potential threats to actors and explain how to mitigate identified threats using security standards.

Thus, this paper presents a CC approach to eliciting security requirements of IT systems based on use case "actor profiles". The specification of both primary and supporting actors of a use case and completion of an actor profile allow for pre-defined threats to be derived and mapped to security objectives and requirements based on a data set used by the CC Toolbox [3]. Violet [15], an open source UML diagram modeling tool, has been extended to implement the approach from a use case textual description perspective.

Such an approach will not be easy since CC standards by themselves are often too confusing and technical for non-security specialists to understand and therefore utilize [8]. It has also been concluded that CC protection levels are rarely used in practice [9]. However, the number of software vulnerabilities reported to the CERT Coordination Center reached 4, 129 in 2002, 3, 784 in 2003, 3,780 in 2004, and a record high 5,222 in 2005 [4]. These statistics show that software is currently being built and deployed with vulnerabilities. The presence of vulnerabilities in software can ultimately lead to attacks, and the failure or misuse of IT systems is simply not an option due to society's increased dependency on them. The approach presented here is an effort aimed at combating these problems from the ground up by integrating IT security seamlessly into the software development life cycle (SDLC). We also hope our approach will aid in making the CC methodology more readily available to non-security specialists in an understandable context.

2. Related Work

Sindre and Opdahl extended regular use cases both in UML diagram and textual template form with misuse cases, which specify behaviors not wanted in a system [8]. They propose labeling misusers and misuse cases along with normal actors and use cases to represent threat and mitigation in a diagram. Similar to our approach, misuse

cases are driven by threats. However, Sindre and Opdahl focus mainly on providing method guidelines for helping individuals describe misuse cases textually whereas our approach systematically derives threats and maps them to security objectives and requirements based on the CC. Nevertheless, Sindre and Opdahl recognize the possibility of integrating misuse cases with the CC [8].

McDermott and Fox have proposed abuse cases, which are very similar in nature to misuse cases, to capture security requirements and provide help during the requirements, design, and testing phases of a security engineering process [11]. In later work, McDermott further describes how abuse cases can be extended for providing a lightweight means of increasing assurance in security relevant software [10].

A key difference between abuse-case based approaches and our approach is that abuse cases clearly indicate actual harm to a system resource, stakeholder, or the system itself. Instead of describing actual harm in terms of abuse, our approach is driven by the potential for system threats and therefore the possibility for harm to occur. Furthermore, our approach utilizes the CC and does not provide any assurance arguments or methodologies for giving such arguments.

Claiming that misuse cases are highly effective ways of analyzing security threats but are inappropriate for the analysis and specification of security requirements, Firesmith has proposed security use cases [5]. Security use cases represent countermeasures that mitigate threats and are driven by misuse cases. Firesmith also shows how security use cases can be incorporated in UML diagrams and further detailed in template form. Similar to our approach, the goal of security use cases is to specify security requirements that protect assets from harm realized by threats. However, unlike security use cases, our approach aims to counter threats using CC methodologies to specify objectives and requirements.

3. Common Criteria Approach

3.1. Overview

Beginning July 1, 2002, any U.S. Government acquisition of IT systems dealing with information security must pass a CC evaluation or equivalent [12]. In general, the CC presents requirements for the IT security of a product or system under the distinct categories of functional and assurance requirements [2]. The CC functional requirements define desired system behavior. The CC assurance requirements are used to provide confidence that desired security measures are effective and implemented correctly. Our approach focuses solely on using CC functional requirements, which fall under the following eleven categories: security audit, communication, cryptographic support, user data protection, identification and authentication, security management, privacy, protection of security functions, resource utilization, target of evaluation access, and trusted path/channels [2].

A target of evaluation (TOE) is the product or system being evaluated. One of the most important documents required to be written before a CC evaluation can take place is a TOE security target. One purpose of a security target is to describe the TOE environment by identifying threats, establishing a set of security objectives to counter identified threats, and specifying security functional requirements to meet each identified objective.

To help developers prepare for a CC evaluation, a program called the CC Toolbox is freely available currently as unsupported software [3]. The CC Toolbox guides developers through the process of creating a security target and is packaged with a pre-defined environment data set that contains a listing of threats, objectives, and CC functional requirements which may be used when describing the TOE environment.

Our approach involves describing a system being designed in ways similar to how a TOE is required to be described in a security target. More specifically, our approach first identifies threats to the actors of a use case and then uses a portion of the CC Toolbox data set to map objectives to threats and requirements to objectives.

3.2. Correlating the Common Criteria

To correlate use cases with the CC, there is a need to specify both the primary and supporting actors of a use case and to describe each actor more formally than is commonly done in traditional practices. To clarify, consider a use case *check grades* that has a primary actor student as shown in Figure 1. Implied in the successful completion of *check grades* is the interaction with an academic database management system (DBMS) that contains grade data. To analyze *check grades* from a security standpoint, the DBMS interaction needs to be explicitly stated, and the relationship between the student and the DBMS actors needs to be described. As shown in Figure 1, the primary goal of the student actor is to read grades, and the primary goal of the academic DBMS actor is to retrieve grades. Furthermore, *check grades* is described as a private exchange since private information is flowing between the

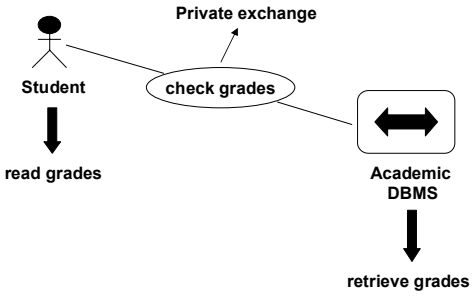

Figure 1: check grades use case

two actors.

To fulfill the need for representing use cases in this manner, our approach requires a use case creator to complete an actor profile for each actor involved in a use case. An actor profile has seven fields declaring the actor's type, location, use case association, and whether or not the use case involves exchanging private and secret information. Table 1 shows an example of an actor profile for the academic DBMS supporting actor in the previously mentioned *check grades* use case.

Table 1. Academic database actor profile.

Actor:	Academic DBMS
Use Case:	check grades
Type:	cooperative
Location:	local
Private Exchange:	true
Secret Exchange:	false
Association:	REQUEST = retrieve

3.3. Description of Actor Profiles

In line with Robertson and Robertson [13], an actor is considered to be one of three types: human, cooperative, or autonomous. In Table 1, note that the actor is considered to be of type *cooperative* since the DMBS must cooperate with the primary actor of *check grades* by a request-response dialog. Human actors are active entities that interact directly with the information flow of a use case, and autonomous actors act independently of the information but have connections to it [13]. Thus, an actor can be a human actor or person, cooperative actor such as a DBMS or server, or an autonomous actor such as standalone computational software. To continue describing an actor profile, the location field specifies the actor's physical location with respect to the system and can either have a value of local or remote. An actor should be classified as remote if it is possible for the actor to interact with the system from a remote location even if the actor may also perform operations from a local location (e.g., an employee can access a corporate database both at work in the office and at home using a personal computer). The value of the private field is either true or false depending on whether or not information flowing to and from the actor should remain private. Likewise, the value of the secret field is either true or false depending on whether or not confidentiality of the flowing information needs to be ensured.

Depending on the actor's type, the use case creator must select the actor's association with the use case from a set of pre-defined categories. The association value reflects the actor's overall goal in successfully completing the use case. For example, if the actor's type is cooperative, then the association is a "request", and it can hold any one of the following values: retrieve, store, retrieve_store, send, or receive. The value of the request association depends on whether information is only being retrieved, stored, retrieved and stored, sent, or received by the cooperative actor. Furthermore, the supporting actor's association needs to be validated with the primary actor's association for use case coherence, and vice versa. In Figure 1, since the value of the academic DBMS association is *REQUEST = retrieve*, the student association can only be *ACTION = read*. This relationship claims that the student checks her grades by reading them and the DBMS only needs to retrieve them. Similar valid relationships exist for all the possible combinations of actor types and associations. Table 2 shows all associations for each of the three actor types.

3.4. Actor threats

Actor profiles provide the foundation that allows for threats to be derived and associated with an actor based on the relationships between the actors of a use case. Use case associations are assigned threats from a predefined set of 12 different threat categories encompassing data modification, data interception, data disclosure, privacy violations, auditing, denial of service, repudiation, and several others. The threat categories were partly compiled based on terminology from a threat listing used by the ECMA Public Business Class Protection Profile [6], which was based on the CC, and partly on the predefined environment data set provided by the CC Toolbox [3]. Only those threats determined to be applicable to the majority of all IT systems were included in the 12 categories. Currently, the categories of threats used by our approach are as follows:

- T.Change_Data
- T.Data_Theft
- T.Deny_Service
- T.Disclose_Data
- T.Impersonate
- T.Insider

Table 2. Actor type use case associations.

Human Actor	Cooperative Actor	Autonomous Actor
• ACTION=read • ACTION=write • ACTION=read_write • ACTION=ask • ACTION=answer • ACTION=ask_answer	• REQUEST=retrieve • REQUEST=store • REQUEST=retrieve_store • REQUEST=receive • REQUEST=send	• FUNCTION=display • FUNCTION=update • FUNCTION=display_update • FUNCTION=receive_orig • FUNCTION=send_new • FUNCTION=no_change

- T.Outsider
- T.Privacy_Violated
- T.Repudiate_Receive
- T.Repudiate_Send
- T.Spoofing
- T.Social_Engineer

Adhering to the recommendations of the CC, threats are prefixed with a capital letter (T) followed by a period. Some threats, such as T.Change_Data, T.Data_Theft, and T.Deny_Service, have sub-threat categories that better refine the threat and apply it to a more specific situation. For example, T.Data_Theft has a sub-threat type called Intercept for describing eavesdropping that occurs on communication lines and a second sub-threat type called User_Collect for describing situations when a user abuses authorization to collect data.

3.5. Deriving Actor Threats

Using the above threat categories, threats are associated with an actor based on an evaluation of the actor's profile. For example, Table 3 outlines the threats that would be associated with a human actor, which is specified to be interacting with a cooperative or autonomous actor in the actor's profile, based on the value of the human actor's "action" field. The threats applied to the *ACTION=read_write* association would simply be a combination of the threats applied to both *ACTION=read* and *ACTION=write*.

To further explain the threat derivation process, consider the previously mentioned *check grades* use case that has a student primary actor and an academic DBMS supporting actor as declared in Table 1. The *check grades* use case involves a private information exchange between a remote, human actor and a local, cooperative actor. After comparing both of the actor profiles, Table 4 shows that four threats were derived for the student actor, and five threats were derived for the academic DBMS actor.

Once actor threats have been identified, objectives must be established to counter each threat, and requirements must be specified to satisfy each objective. This information is almost entirely provided by data taken from the pre-defined environment data set used by the CC Toolbox. The CC Toolbox data set carries out the CC rationale process by mapping threats to security objectives and objectives to CC functional components (i.e. security requirements). For those threats, such as T.Insider, T.Outsider, and T.Privacy_Violated, not present in the CC Toolbox data set, we have assigned appropriate security objectives defined by the data set to them.

4. Tool Support

To implement our approach, an open source UML diagram modeling tool called Violet has been extended to support use case textual descriptions. To incorporate the true power of use cases, the extensions give Violet users the capability to create, save, and open a "use case bundle", which we define as a collection of use case textual descriptions that all relate to the same system under design.

Table 3. Human actor interaction threats.

ACTION = read	ACTION = write

T.Impersonate
T.Repudiate_Receive
 	
if (Private Exchange)
  T.Privacy_Violated
 	
if (Secret Exchange)
  T.Data_Theft
 	
if (Location == local)
  T.Insider
else
  T.Outsider
	T.Change_Data
	T.Impersonate
	T.Repudiate_Send
	
	if (Private Exchange)
	  T.Privacy_Violated
	
	if (Secret Exchange)
	  T.Disclose_Data
	
	if (Location == local)
	  T.Insider
	else
	  T.Outsider

The dialog used to create and edit use cases has seven tab regions allowing for data entry of the basic fields of Cockburn's template [1] as illustrated in Figure 2. In addition, a new field called "Threats" has been added to the template. The threats tab allows for derived threats to be added to an actor tree. The addition of the threats field to the template is similar to the approach taken by Sindre and Opdahl for specifying lightweight misuse case descriptions [8]. However, instead of describing threats in scenarios, our approach uses the threats field to display threats, security objectives, and security requirements retrieved from the tool's knowledge base.

To demonstrate the practicality of the Violet extensions, consider a simplified student enrollment system that has the previously described *check grades* use case with student and academic DBMS actors. The actors tab allows for the use case creator to specify both the primary and supporting actors and complete their corresponding actor profiles. Although not depicted in Figure 2, by clicking the *View Profile* button for the DBMS actor, the actor profile dialog is populated and shown as in Figure 3. Using the actor profile interface, the creator can specify the type of the actor, its location, whether or not the actor is exchanging private or secret information, and finally the use case association. Although also not shown in Figure 2, another button located on the actors tab titled *Validate Actors*

Table 4. Check grades actor threats.

Student Threats	Academic DBMS Threats
1. T.Impersonate 2. T.Outsider 3. T.Privacy_Violated 4. T.Repudiate_Receive	1. T.Deny_Service 2. T.Privacy_Violated 3. T.Repudiate_Receive 4. T.Repudiate_Send 5. T.Spoofing

initiates the process of validating the specified primary and supporting actor relationship for coherence as explained in Section 3.3. After actor profiles have been completed and validated, threats can be derived by clicking the *Run CC Analyzer* button located on the threats tab. Figure 2 shows the threats tab and the results after analyzing the *check grades* use case.

As shown in Figure 2, threat names begin with a *T.* prefix and are represented as child nodes for each actor in the tree, objective names are prefixed with *O.* and are child nodes of a particular threat, and CC requirements are child nodes of a specific objective. By selecting a threat, objective, or requirement from the tree, a description is provided in the right hand text area. Data taken from the CC Toolbox pre-defined environment data set is used to provide for almost all of the threat, objective, and requirement descriptions.

Also as depicted in Figure 2, right clicking an actor in the tree displays a menu that allows for the creation of new threats not presently stored in the knowledge base. Similarly, objectives can be added to a threat, and CC requirements can be added to an objective. This important feature allows for the creation of specific threats that may only be appropriate for the system under design. Additionally, threat and objective descriptions can be modified to better explain how they impact the system under design.

As an example of extending the knowledge base, a threat called *T.Stu_Unattended_Comp* may be added to the student actor threat list of the *check grades* use case. This new threat may describe a situation when a student leaves an active computer session unattended allowing another individual to use the computer possibly in a malicious manner. In return, an objective named *O.Screen_Lock*, which may require a mechanism to be implemented whereby a user is automatically logged out after a specified time interval of inactivity has elapsed, may be created to counter one aspect of the *T.Stu_Unattended_Comp* threat. Security requirements must then be specified to satisfy the new *O.Screen_Lock* objective. This type of flexibility is necessary while team members are brainstorming use case actor threats, defining objectives to counter threats, and assigning security requirements to satisfy objectives that are not already stored in the knowledge base.

Finally, the user has three options for generating output to a table structured HTML file: general, CC rationale, and a combination of general and CC rationale. The general option generates a use case template with the threats field containing a list of threats associated with each actor. Each threat has a nested list of security requirements which are needed to satisfy the objectives that counter the threat. The CC rationale option generates a mapping of threats to objectives and a mapping of functional security components to objectives. The third option is simply the general report with the CC rationale appended to the end. The CC rationale report is aimed at producing the rationale portion of a TOE security target. Moreover, the mappings clearly show the objectives needed to counter a specific threat and the requirements needed to satisfy a specific objective.

5. Conclusion and Future Work

Our approach is aligned with two of the six best practice guidelines for software security as outlined by McGraw: one being security requirements engineering and the other being security analysis, security testing, and use of the CC 7]. Our approach also acknowledges the second priority of the President's National Strategy to Secure Cyberspace by helping to assess and secure emerging systems in order to reduce threats and related vulnerabilities [14]. In addition, it is anticipated that use of our approach will help ensure the following:

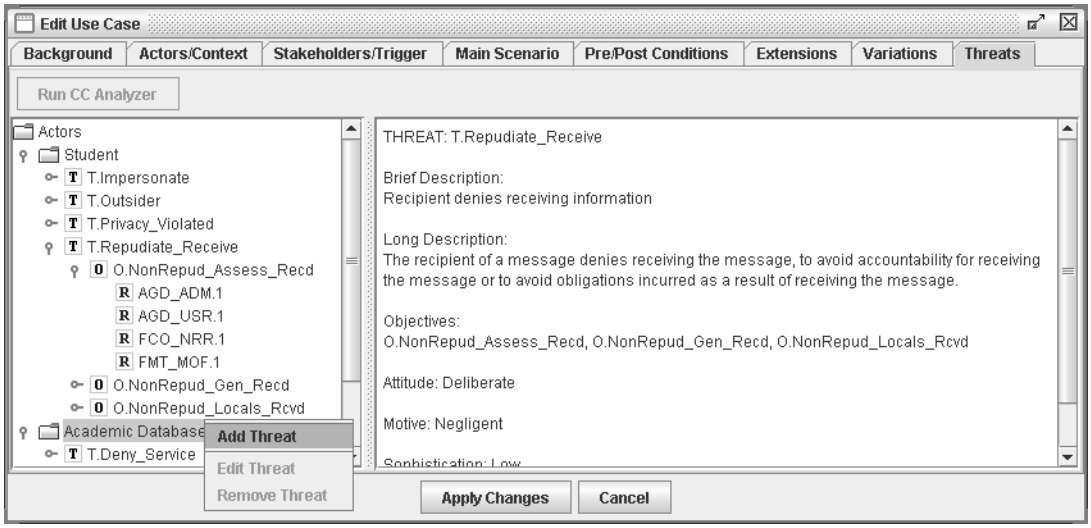

Figure 2: Threats tab.

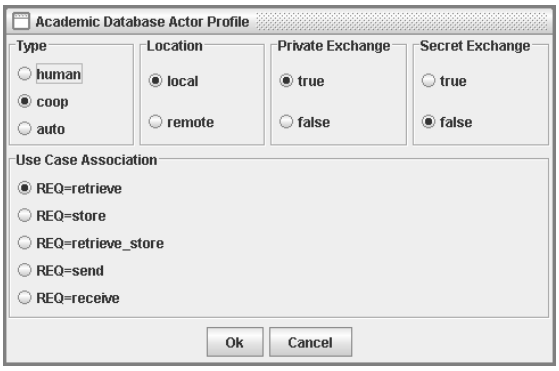

Figure 3: Actor Profile Dialog.

- consideration of security issues, most notably actor threat analysis, early during the SDLC;
- CC made more readily available to non-security individuals, such as end-users and developers, in an understandable context;
- providing aid in determining a more complete set of security requirements for a system under design.

Current work is on going to investigate ways to incorporate our approach into use case diagrams for those individuals who prefer a visual context view of the system. Also, while our approach focuses primarily on threats, it may be more beneficial to further describe the system environment by stating policies and assumptions aligned with the CC. Finally, it has yet to be studied how our approach can be extended to specify CC security assurance requirements.

Our approach is not intended to make a complete determination of all possible threats and security requirements for a system under design. Rather, it is intended to jump-start the security requirements engineering process as early as possible in the SDLC while utilizing the CC in an understandable manner.

6. Acknowledgements

This work was done at the University of South Carolina as part of the Research Experiences for Undergraduates in Multidisciplinary Computing project supported in part by National Science Foundation Award # 0353637.

7. References

[1] A. Cockburn, "Writing Effective Use Cases", Addison-Wesley, Boston, 2001.

[2] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, Version 2.2, CCIMB-2004-01-001, January 2004.

[3] Common Criteria Toolbox Version 6, SPARTA, Inc., February 2003, Retrieved June 15, 2005, from http://cctoolbox.sparta.com/.

[4] Computer Emergency Response Team (CERT) Coordination Center, CERT/CC Statistics 1988-2005, Vulnerabilities report, Retrieved September 5, 2005, from http://www.cert.org/stats/cert_stats.html.

[5] D. Firesmith, "Security Use Cases", *Journal of Object Technology,* http://www.jot.fm/issues/issue_2003_05/column6, May-June 2003, vol. 2, no.3, pp. 53-64.

[6] European Computer Manufacturers Association (ECMA) International, ECMA protection profile: E-COFC public business class. Technical report, TR/78, Geneva, Switzerland, 1999.

[7] G. McGraw, "Building Security In: Software Security", *IEEE Security and Privacy*, IEEE Computer Society, March/April 2004, pp. 80-83.

[8] G. Sindre and A.L. Opdahl, "Eliciting security requirements with misuse cases", *Requirements Engineering 10*, Springer-Verlag London Ltd, January 2005, pp. 34–44.

[9] Improving Security Across the Software Development Life Cycle. Task Force Report, April 1, 2004. Retrieved June 10, 2005, from http://www.cyberpartnership.org/SDLCFULL.pdf.

[10] J. McDermott, "Abuse-case-based assurance arguments", *Proceedings of the 17th annual computer security applications conference (ACSAC'01)*, New Orleans, Los Angeles, 2001.

[11] J. McDermott and C. Fox, "Using abuse-case models for security requirements analysis", *Proceedings of the 15th annual computer security applications conference (ACSAC'99)*, Phoenix, Arizona, 1999.

[12] National Security Telecommunications and Information Systems Security Policy No. 11, Revised Fact Sheet, National Information Assurance Acquisition Policy, Retrieved June 21, 2005, from http://niap.nist.gov/cc-scheme/nstissp_11_revised_factsheet.pdf.

[13] S. Robertson and J. Robertson, "Mastering the Requirements Process", Addison-Wesley, London, 1999.

[14] The National Strategy to Secure Cyberspace, February 2003. Retrieved June 3, 2005, from http://www.uscert.gov/reading_room/cyberspace_strategy.pdf.

[15] Violet: Very Intuitive Object Layout Editing Tool, Retrieved June 12, 2005, from http://www.horstmann.com/violet/.