Eliciting Security Requirements through Misuse Activities

Fabricio A. Braz  
Electrical Eng.  
University of Brasilia  
fabraz@unb.br

Eduardo B. Fernandez  
Computer Science and Eng.  
Florida Atlantic University  
ed@cse.fau.edu

Michael VanHilst  
Computer Science and Eng.  
Florida Atlantic University  
mike@cse.fau.edu

Abstract

In previous work we introduced an approach for finding security requirements based on misuse activities (actions). This method starts from the activity diagram of a use case (or a sequence of use cases). Each activity is analyzed to see how it could be subverted to produce a misuse of information. This analysis results in a set of threats. We then consider which policies can stop or mitigate these threats. We now extend that approach to consider in the analysis the type of misuse (confidentiality, integrity ...) that can happen in each activity, the role of the attacker, and the context for the threat. This extended analysis results in a finer and more systematic way to find threats and we can identify now more threats. We also improve the way to find the policies to control these threats and we consider how to map the corresponding policies to security patterns. The information in each pattern helps in the selection of an optimal (or good) set of policies. Our extended approach can be conveniently incorporated in a methodology to build secure systems.

1. Introduction

In spite of the large amount of research on security [15], several indexes have shown an increase of system security weaknesses, vulnerabilities, and incidents [3].

Security must be addressed in all activities in all phases of the software lifecycle [15]. For complex applications, a comprehensive approach to security – top-to-bottom, beginning-to-end, and everywhere-in-between – is a basic requirement. Until now, no single software engineering methodology exists that can ensure security in the development of large-scale software systems [15].

The complexity of software security requirements starts with their definition. In the Software Assurance Common Body of Knowledge [15], security requirements are defined in terms of system security policies, which state constraints on functionality. Haley et al. [9] follow the same idea, defining security requirements as constraints on functional requirements. We agree that in several cases security is a matter of constraints, but in many others security is a more general need, and therefore it cannot be treated as a simply set of restrictions.

The role of security requirements is to provide information about the actual needs of a system or application with respect to security in order to accomplish its business goals. Security requirements differ significantly from one system to another. Typically, systems need some combination of authentication, authorization, transaction integrity, accountability, and message secrecy. Many studies attempt to enumerate the vulnerabilities of existing platforms.
We believe that we should look at the system from a higher level perspective. An attacker has an objective or goal that he wants to accomplish, e.g., steal the identity of a customer, transfer money to his own account, etc. Security requirements should define the needs of the system without committing to specific mechanisms, by focusing on the potential attack or threat.

Several approaches to deal with security requirements have been proposed. We review some of them in Section 4. We have presented an approach to enumerate threats by considering each activity in each use case and seeing how it can be subverted by an internal or external attacker. This paper extends these results to make the method easier to apply and more systematic in identifying threats. We also consider the transition from requirements to analysis.

This paper is organized as follows: in Section 2 we describe the misuse activity approach, while in Section 3 we show the proposed improvements to the method. Related work is discussed in Section 4, and finally, conclusions and future research are given in Section 5.

2. Finding Threats through Misuse Activities

The misuse activities approach consists of a systematic way to identify system threats, and determining policies to stop and/or mitigate their effects. To do that, two activities are employed. The first activity is an analysis of the flow of events in a use case or a group of use cases, in which each activity is analyzed to uncover related threats. This analysis should be performed for all the system uses cases. The second activity entails a selection of appropriate security policies which can stop and/or mitigate the identified threats.

The analysis of the flow of events in a use case implies a detailed investigation of each activity from each use case, aiming to figure out any possible ways to subvert it by either insiders or outsiders. All misuse action-related information is documented on UML
activity diagrams, whose notation has been extended as follows. Dashed rounded rectangles denote misuse activities or threats. Dashed connecting lines represent misuse related control flows, which associate misuse activity and their related objects. An extended activity diagram example is shown in Figure 1, in which several threats are identified, denoted by T1 to T8, from internal/external sources, as shown on section a) in Figure 2. These threats are obtained by the analyst using his knowledge of the application; the approach is systematic in that all activities in all use cases are analyzed. For example, T4 indicates that after the customer has provided personal information, the manager could illegally disseminate this information to others.

Once misuse actions have been uncovered, appropriate security policies must be selected to stop and/or mitigate them. This activity could be aided by some comprehensive security policy list [8]. However, this selection should result in a minimum set of mechanisms instead of mechanisms piled up because they might be useful. For example, to mitigate the threats T5 and T6 shown in Figure 2, we can choose the policy “separation of duties”. Other examples can also be found in Figure 2.

Although the misuse activities approach shows a systematic way to identify and diagram threats and to select suitable policies, it lacks details which could aid in applying the method. It becomes evident when executing its two main tasks that some basic questions might appear:

- How should the analysis in each activity be performed in order to elicit all (or most) of the threats?
- How to get from the misuse action to the appropriate system policy or policies in a simple way?

The next section addresses these questions.

3. Extending the Misuse Activities Approach

High level or institution security policies are general security directives or guidelines with which any service, practices or product of the organization must comply [2]. They are defined according to the appropriate business drivers, including institution vision, marketing objectives, and related regulations and standards, such as PCI [13]. Ideally, any organization should have these policies well defined, and clearly communicated to its employees. Indeed, without them it’s impossible to figure out how secure should the company systems be, to fit its needs, and what effort and money to spend in providing this security.

In order to uncover the associated threats, three aspects will be considered when performing this analysis. The first refers to the use cases, in which it is expected to find all system scenarios. Sometimes, it is necessary to analyze a sequence de uses cases which represents a critical business workflow, therefore the activity diagram should depict not only a specific use case flow, but a business workflow. Let’s take the
The second aspect is the set of security attributes. Every activity from the activity diagram should be analyzed according to the standard main security attributes: confidentiality, integrity, availability and accountability. All this means that we must analyze whether the activity could be subverted in terms of confidentiality: traffic analysis, covert channels, inference, and information disclosure; integrity: unauthorized data modification, transactional integrity, deception, masquerading, spoofing, impersonation; for availability: denial of service, disruption; and accountability: repudiation, track erasing.

As the third aspect, we have the source of the threat, which relates to the privileges that the threat agent has to execute the attack. This aspect has been increasing its relevance after the release of reports such as [14]. Here, we can divide the attackers into few groups: the outsider, people with no authorization to access any part of the system; the authorized insider, people who can access the system and to execute the activity; and the unauthorized insider, who has system access authorization, but not to the activity being analyzed [4].

Another important aspect not explicitly considered here, which affect the analysis is the system context, represented by environmental characteristics, including people, operation dynamics, and other systems already in place. An example of this influence is shown below.

As before, an activity diagram is the starting point to perform the analysis, which could be done in parallel. We must scrutinize the activities keeping in mind the following statement “What misuse could be done in <activity> by <source> which compromises <security attribute> from the <asset>”. In this statement the activity refers to each activity from the activity diagram, the source refers to the third aspect (source of threat), the security attribute refers to the second (type of misuse), and the asset refers to the flow object [4], e.g. in Figure 1, one of the threatened assets is the object “Account2”.

Figure 3 shows a table which results of applying this approach to the activity diagram in Figure 1. This table summarizes the results after applying the misuse activities approach to uncover threats. In order to make it easy to notice the differences, the threats already uncovered, shown in Figure 2, are in black (their numbers have been changed). As presented in Figure 3, several more misuse activities have been uncovered, compared to the direct application of misuse actions, such as “T11 - Changes the customer credit info to get more clients”.
more clients”. In this case, the analysis of integrity violation performed by some authorized internal user leads to consider this threat scenario. Such outcome is tied with the main goal of the misuse activities approach, which is to support the security requirements analysis in a such way that we would not forget any obvious threat. Additionally, when we give a first class treatment to the (misuse) activities performed by internal users, we can deal with the biggest source of threats in a proper way. It is still feasible that some threats might not be found by this approach, but in despite of this, the new approach appears to be more effective and useful than the earlier approach.

The context aspect can lead to some different conclusions in terms of the relevancy and suitability of some threats. In the example in Figure 1, if we consider that it is not possible to open accounts remotely, it means that any customer who wants to open an account must go to a bank agency to do it. In this case, some threats might be disabled, e.g. “T3 - Eavesdropping” and “T4 - Uncovers customer relationship with the institution” by trying to create new account with his info”.

Once we have the possible threats, we need to find policies to stop or mitigate them. That is, the threats represent the security problem space and the policies the solution. We may need a combination of policies (called here a “response”) to control each threat. For example, Figure 4 shows that policies $P_1$ and $P_2$ can be combined into response $R_1$ to control $T_1$. In concrete terms, in order to mitigate the threat $T_{11}$ in Figure 3, we may need a response, called here “stop corrupt manager”, which combines “Separation of Duties” and “Logging” policies. Obviously, the same response can mitigate more than one threat.

After defining the needed combination of responses, a combination of security patterns can be used to enforce such policies (called here “realization”). As an example, in Figure 4 the response $r_1$ can be realized by $R_1$, which includes patterns $P_1$ and $P_2$, or $R_2$, which includes pattern $P_4$. Note that a response may require an AND of security patterns. In order to realize the response “stop corrupt manager”, as shown above, we need to implement two policies as patterns, we could use the combination of “role-based access control”, including constraints for separation of duties, and “security event logging” patterns. Additionally, Figure 4 highlights a complete decision path which addresses a hypothetical set of threats.

4. Related Work

As mentioned earlier, there are many approaches to elicit software security requirements. In [11] they explore asset identification, but extracting assets from the activity diagram is much more effective and systematic.

Another proposed approach is the abuse/misuse case approach [1, 18]. Misuse cases are independent use cases initiated by external attackers to the system. While this approach can help in a high level elicitation of security requirements, it does not show how to associate the misuse cases to legitimate behavior and concrete assets; therefore, it is not clear what misuse case should be considered, nor in what context.

Mouratidis et al. in [12] present a method that applies “security constraints” to relationships in the domain and architecture. While a security constraint specifies a type of protection, it does not address why the protection is needed, nor, more importantly, what happens when the constraint is violated.

By comparison, our approach provides a rational basis for choosing among security alternatives and supports backup mitigation, recovery, and forensics as part of the design.

The threat modeling approach [5, 19], on one hand gives a clear idea of how to elicit, classify, prioritize and mitigate threats, but on the other hand it demands information available only at late design time, which drives the security design to start in the middle of the software development life cycle. The misuse actions approach can be applied from the very beginning, since it only requires as precondition to already have use cases.

Michael Jackson’s problem frames have been used as an alternative to elicit and analyze software security
requirements [9,10]. Hatebur et al. [10] describe a security engineering process to develop security systems based on problem frames, and a collection of security patterns\(^1\), plus components as the way to deal with the solution. While problem frames appear useful in some cases, they are not as useful for a complete design as UML. Also, they are not so widespread.

5. Conclusions

In the past, we developed an approach for software requirement elicitation based on the misuse activities that focuses on analyzing the activities from an activity diagram representing a use case, from the attackers’ perspective.

In order to make the misuse activities more systematic, we have presented here an improvement on its threat analysis by adding two more aspects to be taken in consideration when performing such analysis, the security attribute subverted (confidentiality, integrity, availability, and accountability), and the source of threats.

As the preliminary results show, by applying these improvements the approach became more effective, since several more threats have been found. Those threats should be analyzed by likelihood and impact, before going into the design stage.

We have also introduced an idea to realize policies as security patterns, which is the focus of our current work.

References


\(^1\) The patterns mentioned in that paper have no relation to standard security patterns [16].