Goal-Based Requirement Engineering for Fault Tolerant Security-Critical Systems

Davoud Mougouei

Department of Information System
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia, 43400 Serdang,
Selangor, Malaysia
dmougouei@gmail.com

Abstract

Large amount of security faults existing in software systems could be complex and hard to identify during the fault analysis. Therefore, it is not always possible to fully mitigate the internal or external security faults (vulnerabilities or threats) within the system. On the other hand, existence of faults in the system may eventually lead to a security failure. To avoid security failure of the target system it is required to make the system flexible and tolerant in the presence of security faults. This paper proposes a goal-based modeling approach to develop security requirements of Security-Critical Systems (SCSs) through explicitly factoring the faults into the requirement engineering process. Our approach establishes the Security Requirement Model (SRM) of the system based on its respective Security Fault Model (SFM). We incorporate fault tolerance into the SRM through considering the partial satisfaction of security goals. The proposed approach factors this partiality into the goals by using proper mitigation techniques during the refinement process. This approach eventually contributes to a fault tolerant model for security requirements of the target system.

Keywords: vulnerability, intrusion tolerance, threat, security fault, security goal

1. Introduction

Security is one of the most important concerns of any software system. But, today’s software development techniques do not pay enough attention to engineering of security into the early development stages [1, 2]. Lack of a traceable methodology [3] for performing the security analysis in early stages of the software development is also another issue to be addressed by practical security engineering approaches. On the other hand, not all of the security faults existing in software systems are identifiable during the fault analysis [4]. Due to the unavailability of complete fault prevention we have to remove security faults. Fault removal also may introduce new fault(s) into the system. Hence, exhaustive fault analysis is not possible and elicited security requirements may not mitigate all of the security faults [5]. Consequently a security failure may occur. To avoid security failure of the system we need to make it tolerant in the presence of unpreventable security faults [6]. Nonetheless recently reported approaches toward fault tolerance (intrusion tolerance) of SCSs have been mostly focused on applying fault tolerance techniques to the system’s architecture and design [6] rather than caring for fault tolerance in the security requirements of the system. In this paper we have presented a goal-oriented approach toward modeling and specification of security requirements and faults for fault tolerant security critical systems. Our proposed
approach incorporates fault tolerance into the security requirements of the SCS through accepting partial satisfaction of security goals. Although probabilistic logic is also previously used in [7] to care for partiality of the goal satisfaction but it does not properly capture the partiality [8]. We choose fuzzy semantic of RELAX [9] over probability since in the probability logic, the goal is either satisfied or not while in the fuzzy logic, the goal may be “roughly” satisfied [9]. We employ RELAXation [8] technique and temporal fuzzy requirement engineering language of RELAX [10] incorporated into the KAOS [9, 11] to accept and formally describe this partiality in security requirements.

This paper, therefore, has three main contributions. Firstly, it gives a process for elicitation and formal description of security requirements and faults. Secondly it factors the security faults into the requirement and fault model of SCS’s [9] and finally the paper incorporates fault tolerance into the requirement model of the system through partial satisfaction [8] of security goals. We illustrate our approach by applying it to a typical online banking system (OBS), a security-critical system providing banking services. The remainder of the paper is organized as follows. Section 2 discusses related works. Section 3 presents our modeling approach. Section 4 introduces OBS as our running example and describes the details of applying the approach to the OBS. Finally, in Section 5, we present conclusions and discuss future works.

2. Related Works

Attack trees are widely used to model how the attacks can be performed [12-14]. Kenneth S. Edge in [15] proposes the concept of protection trees to determine which countermeasures are required and where to place them in the system. The work [16] has employed first order logic expressions to resolve goal satisfaction difficulties through precise description of security goals and anti-goals. However, none of the mentioned works address partial satisfaction of security goals to tolerate the violations. The work [7] has tried to resolve this problem by propagation of degrees of satisfaction within the goal refinement process. The technique uses probability to care for partiality. However as discussed before, probability does not properly capture the partiality [8]. Current trends toward fault tolerance of security critical systems have been mostly focused on applying fault tolerance techniques to the systems’ architecture and design [6] rather than caring for fault tolerance in the security requirements. The applied techniques include replication, redundancy, indirection, reactive techniques, and reconfiguration [17]. One of the most primitive works in this area [18] proposed intrusion tolerances techniques for storing files. Intrusion tolerance has also been successfully applied to authentication and authorization servers [19] as well as data processing area[20]. Merideth et al., [21] introduced “Thema” which is Byzantine Fault-Tolerance middleware system in order to execute the Byzantine Fault-Tolerance by capturing all requests and responses. Aghaie et al., [22] have proposed client transparent fault tolerance model for web services which applies N-version and active replication techniques to identify server errors and redirect requests to reserved backup server in order to reduce the service failures. Santos et al., [23] proposed fault tolerance infrastructure by which it adds an extra layer acting as proxy between client requests and service provider’s response to ensure client transparent faults tolerance. Another important work is [6] in which the author describes some of the strategies such as reconfigurable operations, Recoverable operations, fail-safe behavior to be considered
in the architect of intrusion tolerant systems. However the paper does not specify how to apply these strategies during the security analysis phase.

3. Modeling Approach

Our approach to security requirement gives a step by step process for modeling and formal description of security requirements and faults. We incorporate fault tolerance into the security requirement model of the target system by caring for partial satisfaction of security goals. Based on top-level security goals and through the process of goal development [8] we develop the goal (requirement) model of the system. We use KAOS [16] to explain security goals and faults in our models. For this purpose we use the Maintain and Avoid keywords to specify “always” goals and the Achieve keyword to specify “eventually” goals [24]. We accept and describe partial satisfaction of the security goals through assigning proper RELAX [9, 10] statements to the “relaxed” attribute of the corresponding security goal. In a similar spirit, we use our goal development process to develop the fault model of the system with respect to the possible security faults. In this section, we specify how to formally describe the SRM and SFM based on KAOS and vocabulary of RELAX. The attributes used for description of security goals and faults are introduced in Table 1.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Values (Sample)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifies the requirement or fault</td>
<td>R1.2.3, F1.2.3</td>
</tr>
<tr>
<td>ACTOR</td>
<td>Responsible human or software agent</td>
<td>Agent’s name</td>
</tr>
<tr>
<td>MODAL</td>
<td>Modality of the requirement or fault in terms of RELAX’s shall statements</td>
<td>shall, may</td>
</tr>
<tr>
<td>STATE</td>
<td>Temporal logic operator</td>
<td>Finally, Generally</td>
</tr>
<tr>
<td>PHRASE</td>
<td>KAOS statements</td>
<td>Achieve, Avoid, Maintain</td>
</tr>
<tr>
<td>RELAXED</td>
<td>RELAX statements that addresses faults in SRM</td>
<td>As close as possible to value</td>
</tr>
</tbody>
</table>

3.1. Modeling Steps

Our proposed goal-based modeling approach can be applied through six steps as outlined in Figure 1. Each step leads the requirement engineer to finally achieve a fault tolerant SRM for the SCS. For a given system specification, firstly we identify the system assets [25]. To protect the system assets we develop the security goals. Then, security experts develop the security artifacts representing the possible threats against the system [26]. Throughout the subsequent steps we develop and refine the SFM and SRM to mitigate (tolerate) the security faults. The refinement process continues till the SRM properly mitigates (tolerates) the SFM. This will be determined by security experts.
Figure 1. Steps to Apply the Proposed Approach

3.1.1. **Step 1: Develop Use Case-Misuse Case Diagram.** Before starting the requirement elicitation process, we develop use case-misuse case diagram of the system to identify the key (misuse) scenarios of the system and the relations among them.

3.1.2. **Step 2: Identify Assets and Security Goals.** This step is to identify the valuable system assets and security goals based on use case-misuse case diagram and business concerns of stakeholders [25]. We initiate the SRM based on identified security goals. This contributes to efficiency of the resulting protection model [7].

3.1.3. **Step 3: Develop SFM.** In this step we refine the SFM based on system’s security artifacts and security goals specified in SRM. The security artifacts include all kinds of artifacts such as attack scenarios, misuse cases and attack trees which are developed by security experts [25] to identify security faults in the target system. The initial SFM as depicted in Figure 3 will be developed through inverting the initial SRM. By inverting we mean interchanging all ‘OR’ and ‘AND’ operators [16] and also inverting the description of operand goals as explained in [7]. This leads the security experts to develop only the faults threatening the system assets.

3.1.4. **Step 4: SFM Analysis and Pruning.** In this step, security experts evaluate the developed SFM and reduce the SFM to the feasible threats. An attack path will be pruned if the satisfaction cost involved is higher than the value of the target asset.
3.1.5. Step 5: Develop SRM and Mitigate Faults. In this step we develop the security requirement model of the system to mitigate the threats modeled in SFM. In the following we introduce the techniques we have employed to mitigate (tolerate) the faults.

i. Add low level (sub) goals. For a threatened security goal, the first and the least costly mitigation technique [8] is to mitigate the threat throughout adding a new requirement as a sub goal.

ii. RELAX goal. If adding new sub goals does not mitigate the fault and partial satisfaction of the security goal is acceptable, then we tolerate the security threat through RELAXation. We RELAX the security goal by assigning a proper value to its ‘relaxed’ attribute. Fuzzy semantic of RELAX statements properly reflects partial satisfaction of the security goals. As an example for RELAXation, consider the following goal:

"R4: OBS shall generally maintain [service][availability]"

The security goal R4 may not be satisfiable all the times; for example when a denial of service (DOS) attack occurs. However if a short interruption in service providing is tolerable, we can accept partial satisfaction of the goal through its RELAXation. For this purpose we use a temporal RELAX operator as follows:

"R4: OBS shall generally maintain [service][availability] as close as possible to optimumAvailability"

As depicted in the relaxed R4, the ‘relaxed’ attribute has the value of ‘as close as possible to optimumAvailability’ where the ‘optimumAvailability’ represents the optimal value for availability of the system. The optimal value maximizes the membership function of the fuzzy set. However optimal value is not necessarily the maximum value. This is explained in terms of fuzzy nature of RELAX semantic.

The fuzzy membership function \( M \) in Equation (1), reflects the degree of satisfaction of R4 based on the difference between current availability and the ‘optimumAvailability’. The membership function \( M \) has its maximum value (i.e., 1) at the availability of optimum and decreases for other availability values.

\[
\Delta(R4) = \text{Measured value for availability of the system} \\
M(\Delta(R4) - \text{optimumAvailability}) = 1, \text{if } \Delta(R4) = \text{optimumAvailability} \\
M(x) = \text{MembershipFunction of fuzzy set of } S \mid M(x) \rightarrow S, M(0) = 1 \\
S = \{(x_i, M(x_i)) | M(x_i) \in [0,1], x \in \mathbb{R}\} \\
(1)
\]

of R4 based on the difference between current availability and the ‘optimumAvailability’. The membership function \( M \) has its maximum value (i.e., 1) at the availability of optimum and decreases for other availability values.

iii. Add High Level Goal. For invariant goals [9] which mitigation is not possible through adding low level goals and partial violation of the goal cannot be tolerated, we need to add a high-level goal to mitigate the threat. Adding a high level goal is quiet costly due to the radical effects it imposes on the behavior of the target system [8].

3.1.6. Step 6: SRM Analysis and Pruning. In this step security experts evaluate the degree of security in SRM with respect to the security factors such as cost, impact and flexibility of the security goals [27]. As the consequent, SRM will be pruned based on
the evaluation results. We also may need to refine the newly added high-level goals by performing steps 3 to 6. When to stop the refinement process is left to security experts to decide.

4. Application of Modeling Approach

To demonstrate the validity of our approach, we applied our proposed approach to the case study provided in [15] describing an online banking system (OBS) as a security critical system. An excerpt of the case study is presented here to serve as a running example:

OBS provides some regular banking services like money transfer over the internet. The bank accounts are alluring targets for criminals. Hence, OBS transactions must be protected to keep financial losses to a minimum. The availability of OBS services is as important as the confidentiality and integrity. Financial transactions in OBS might be done via Debit Cards so proper security requirements should be considered to protect them from any possible abuse. The OBS also has a server which should be protected as well. An attacker might exploit the OBS’ internal communication network to threaten the transactions. OBS in addition should prevent unauthorized online access to the system. Hence, it supports user authentication by checking the user name and password. However, the attacker still can guess either user name or password but it is supposed to be difficult. OBS must provide reasonable assurance that their customers’ accounts are secure. The main threat that concerns banks, with respect to online banking, is that an attacker will transfer money out of their customers’ accounts.

In this section we represent the results of applying our proposed approach to the OBS. We give examples of the threat mitigation techniques applied to mitigate (tolerate) the threats.

4.1. Step 1: Develop Use Case-Misuse Case Diagram. In this step we create the use case-misuse case diagram of OBS as depicted in Figure 2. Security use cases represent security functionalities of the system. Actor might be a good actor or an attacker (insider) [26].

4.2. Step 2: Identify Assets and Security Goals. Step 2 is to identify the OBS assets and top-level security goals. Given in the scenario, OBS’s assets include bank accounts, private information of bank and user information (financial transactions and personal information). As a service provider, OBS services also should be reliable and available to the users. To protect the identified assets we initiate the SRM by refining the high-level goal “maintain [OBS][security]” to goals R1 to R4 as depicted in Figure 3. The inverse of ‘maintain [OBS][security]’ goal will be represented by ‘achieve [OBS][insecurity]’ goal [16] in SFM. We develop the initial SFM by simply inverting the SRM. It is also represented in terms of nodes F1 to F4 in Figure 3. The upper part of the graph represents SRM, while the lower part represents SFM that act to confound SRM. The top-level goal in SRM is to maintain security of the OBS. As shown in the Figure 4, security requirements and their corresponding security faults are represented in terms of XML entities. Each requirement or fault entity is describes through assigning proper values to its attributes.
Since, in the initial model of SRM, the security goals are expected to be fully satisfied, ‘relaxed’ attributes have given the null value. Formal description of SRM and SFM will be updated based on the changes made by refinement process.

Figure 2. OBS’ Use Case-misuse Case Diagram

Figure 3. OBS’ Initial SRM and SFM. Junction Points Represent Logical AND while their Absence Means Logical OR
4.3. Step 3: Develop the SFM. In this step we develop the SFM based on the OBS security artifacts. We have depicted an excerpt of the resulting SFM in Figure 5. We only refine of the SFM branches to demonstrate the refinement of the goals threatening the accounts (i.e., achieve [bank accounts][insecurity]). The branch has been refined to the vulnerabilities depicted by red borders on the satisfaction path of fault F1. This means the bank accounts would be threatened if the vulnerabilities specified in the low levels of the SFM are not mitigated. However due to the existence of the unknown vulnerabilities, complete threat mitigation is not possible. Hence we need to care for partial satisfaction of security goals during the development of SRM. After finishing the development of SFM, we need to update the corresponding model description.

4.4. Step 4: SFM Analysis and Pruning. In this step, we identify risks of security faults from the attacker’s point of view. Then we prune the SFM till only the feasible threats are remained.

4.5. Step 5: Develop SRM and Mitigate Faults. In this step we develop the SRM to mitigate the threats reflected in SFM. We start the process by inverting the initial SFM as we explained in Section 3. Resulting model is the SRM developed based on the existing SFM. For example the requirement R1.1.3 in SRM is refined through inverting the fault F1.1.3 (achieve unauthorized online access to account) in SFM. Then we develop the SRM by refining the related sub goals to mitigate the security threats.
Step 5(i). Add low level (sub) goals. We start the mitigation by adding a new sub goal to mitigate the fault. As an example for adding low level goals, the security fault F1.1.1.2 (‘achieve [guess ID]’) in Figure 5, is mitigated in Figure 6 by adding a new goal achieve [generate random ID] indicated by the arrow 5(i) in Figure 6. In other words, we can reduce the likelihood of guessing user ID, through randomizing the generated IDs. In a similar way, we have added a sub goal R1.1.3.2.2.1.1 (achieve [trial limitation]) to mitigate the threat of guessing password by brut forcing. For this purpose, we limit the number of password trials to prevent attackers from brute forcing.

Step 5(ii). RELAX goal. We apply this technique when threats can be partially mitigated. Generating random IDs and implementing password policy and password encryption in OBS can partially mitigate the dictionary attack. However absolute satisfaction of the goal R1.1.3.2 cannot be guaranteed and consequently a dictionary attack may succeed. To avoid security failure of the OBS, the threat should be mitigated (tolerated). In this example, we accept existence of this violence and try to tolerate it through RELAXation. Figure 6 shows the result of RELAXation of requirement R1.1.3.2. We have RELAXed it by assigning the RELAX statement of ‘as close as possible to hardToGuess’ to the ‘relaxed’ attribute of the requirement R1.1.3.2 (as it’s depicted by the arrow 5(ii)). Consequently we update the description of goal R1.1.3.2 as follows:

“R1.1.3.2: OBS shall generally avoid [ID and Password to Guess] as close as possible to hardToGuess”
Figure 6. An Excerpt of the Developed SRM

The value ‘hardToGuess’ is a constant value representing the optimum value for difficulty of guessing password and ID. Note the optimum is not necessarily the maximum value. This is explained in terms of fuzzy semantic of the goal:

“AG ((A (avoid ID and Password to Guess) – hardToGuess) ∈ S)”

Where S is a fuzzy set whose membership function has value 1 at zero (m(0) = 1) and decreases continuously around zero. “A (avoid ID and Password to Guess)” represents the hardness of guessing the ID and password which will be compared to ‘hardToGuess’. As a consequence of such a description, the system designer in the subsequent stages would be aware of partial satisfaction of R1.1.3.2. Hence he can choose proper fault tolerant design strategies to care for this partiality and make the target OBS, tolerant in the presence of unavoidable security faults. However to estimate the value of “A (avoid ID and Password to Guess)”, we need to apply proper metrics for measuring the strength of the password and IDs used. The work [28] has introduced a technique to measure strength of the password through simulating the password cracking algorithms. Degree of satisfaction of goal R1.1.3.2 is left to the OBS designer to determine.

Step 5(iii). Add High Level Goal. Consider a situation in which the ID and Password are guessed by the attacker and the OBS cannot accept such security violence. In this
case, we have to add a high-level security goal such as a supplementary authentication mechanism like challenge-response to mitigate the threat. Consequently, system can still avoid unauthorized access to accounts in case of violation of R1.1.3.1 or R1.1.3.2. However as we mentioned, this newly added high-level goal introduces a new behavior to the system and the closer to the top-level goal it is, the greater the cost of implementation would be. The new goal is ORed with the other high level goals. Challenge-response is a secure and efficient authentication mechanism [29] which tries to identify the user by exchanging some confidential information in a secure way between system and user.

**Step 6: SRM Analysis and Pruning.** In this step we analysis the SRM and evaluate the degree of security in this model with respect to the security factors such as cost, impact and flexibility of the security goals. We prune the SRM based on the analysis results. These results also will be used by security experts to determine the end of refinement process.

4. Conclusion

The large amount of security faults existing in software systems could be complex and hard to identify during the fault analysis and consequently it is not always possible to absolutely satisfy the security goals. Therefore a security failure may occur as a result of this partiality. To avoid security failure of the system we need to care for partiality of the security and tolerating the security threats from the very beginning of the development process. However there are three major problems with most of the current security engineering approaches. Firstly, they fail to properly incorporate the security analysis results into the security requirements of the system. Secondly, they do not take into consideration the existence of unpreventable and accidental security faults. Lack of an applicable methodology for engineering the security in early stages of the software development is also another problem with the current methods. In this paper we have presented a step by step approach toward modeling and specification of security requirements and security faults for fault tolerant security critical systems. The method is a goal-based approach for elicitation of security requirements and construction of security requirement model and security fault model. Our approach employs RELAX to explicitly factor the security faults into the security requirements and address partial satisfaction of the security goals which eventually leads to a fault tolerant model for security requirements of the system. We have illustrated our approach including the process for developing the requirement model by applying it to a typical online banking system (OBS) as our running example. There are several ways to continue this work. When to stop the refinement process must be specified based on the result of risk analysis and evaluation of security in the security requirement model of the system. Measuring the degree of security in the security requirement model of the system is an open issue that we have overlooked it in this paper. To perform such a measurement we need to expand the vocabulary of RELAX to reflect attributes such as cost, impact and flexibility of the security goal. Due to the partiality of the RELAXed goals we also need to have an applicable criterions for measuring the degree of satisfaction.
References


