Support for Security Analysis of Design Models based on Traceability

Hirokazu Yatsu\(^1\), Masaru Matsunami\(^2\), Toshimi Sawada\(^3\), Go Hirakawa\(^4\), Atsushi Noda\(^4\), Naoya Obata\(^4\), Takahiro Ando\(^1\), Kenji Hisazumi\(^1\), Weiqiang Kong\(^5\), and Akira Fukuda\(^1\)

\(^1\)Kyushu University, Fukuoka, Japan
\(^2\)Sony Digital Network Applications, Inc., Tokyo, Japan
\(^3\)Kosakusha, Ltd., Tokyo, Japan
\(^4\)Network Application Engineering Laboratories, Ltd., Fukuoka, Japan
\(^5\)Dalian University of Technology, Dalian, China

Abstract - Software systems embedded into the foundation of information society is required to be secure. Requirements for the system to be secure should be properly recognized in the upper process of system development, and accurately reflected in their specifications and designs. However, security analysis to decide whether systems are secure or not is usually done at the implementation phase of system development or later. In this paper, we propose a universal approach to support security analysis at the design phase. Our approach is to detect vulnerable parts of systems based on traceability established among SysML diagrams, security threats and countermeasures against threats using SMT solvers.

Keywords: Security Analysis, Traceability, SysML, FTA, GSN, SMT solver

1 Introduction

Software systems embedded into the foundation of information society is required to be secure. Requirements for the systems to be secure (in this paper, this is abbreviated to security requirements) should be properly recognized in the upper process of system development, and accurately reflected in their specifications and designs. However, in the current system development, security analysis to decide whether systems are secure or not is mainly performed on code created at the system implementation phase with tools, such as vulnerability scanning on binary code or static analysis on source code. It might be necessary to correct the specifications and design of the systems to remove the security vulnerabilities found through security analysis at the implementation phase or later. In that case, the overhead to correct the specifications and design would be large. Therefore, security analysis on the system specifications and design (in this paper, this is abbreviated to security design analysis) is necessary. In security design analysis, we have to grasp how countermeasures against security threats are prepared in the systems. Unfortunately, the necessity and importance of security design analysis have not been recognized yet.

In this paper, we introduce the attempt of security design analysis in Sony Digital Network Applications, Inc. (hereinafter abbreviated to SDNA Inc.) [1] and our tool to support the security design analysis by detecting vulnerable parts of the systems using SMT solvers.

2 Security design analysis

In this section, as an example of the security design analysis, we will introduce the attempt of SDNA Inc. [1]. This attempt refers to the threat modeling [2]. The subject of threat modeling is security threats with which a system is facing, whereas the subject in [1] is situations where assets of a system such as confidential materials, password, etc., are protected from security threats.

2.1 Threat modeling

The threat modeling draws attack trees, whose structure is same as fault trees drawn in Fault Tree Analysis (FTA for short) (Figure 1). However, representation of the security threats in attack trees varies from person to person. It would be difficult even for practitioners rich in security knowledge to draw attack trees.

![An example of attack tree][3]

2.2 Security design analysis

As security requirements, [1] describes a desirable situation where assets of a system are protected from security

---

[3]: [http://example.com/attack_tree.png]
threats. The assets include information and functions stored in the system. An example of the situation is that a third person cannot read/call such information/functions through any security attack. In [1], security threats that cannot be overlooked in a system are identified as the following two actions of a third person.

1. Access to assets

2. Prevention of user's access to assets

Security requirements of a system assume that assets of the system are always protected from any security attacks, that is, any third person cannot access assets of the system and cannot prevent user's access to the assets. In [1], a security threat is not directly described, but a security requirement is described as a dual (i.e., negation) of existence of some security threat.

Same as the case of drawing attack trees, representation of security requirements are likely to vary from person to person. So, [1] provides a template (Table 1) for description of security requirements. This template provides practitioners a uniform description style for security requirements. So it enables them to represent security requirements similarly to other practitioners.

<table>
<thead>
<tr>
<th>Asset</th>
<th>Action</th>
<th>Security requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>read</td>
<td>&lt;Attacker&gt; cannot read &lt;Information&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Attacker&gt; cannot prevent &lt;Users&gt; from reading &lt;Information&gt;</td>
</tr>
<tr>
<td></td>
<td>write</td>
<td>&lt;Attacker&gt; cannot write &lt;Information&gt;</td>
</tr>
<tr>
<td>Function</td>
<td>execute</td>
<td>&lt;Attacker&gt; cannot write &lt;Information&gt;</td>
</tr>
<tr>
<td>Function</td>
<td>execute</td>
<td>&lt;Attacker&gt; cannot prevent &lt;Users&gt; from executing &lt;Function&gt;</td>
</tr>
</tbody>
</table>

2.3 Decomposition of security requirements

Security requirements can be decomposed according to the configuration of a system. Figure 2 shows an example of a system which manages prescriptions assigned to patients. This system is called “Medication Notebooks System”[1]. In the system, all prescription records registered through receipt computers in pharmacies are gathered in a server. Dotted lines indicate the flow of prescription records in the system. Numbered constituents such as server, receipt computer, tablet, etc. indicate locations where prescription records exist. One of the security requirements of this system is “prescription records cannot be read by a third person in the system”.

That security requirement is equivalent to “prescription records cannot be read by a third person at any location of the system where they exist”, which is represented as a conjunction indicated in the Figure 3.

3 Support of security design analysis base on traceability

A security requirement for a system that the system is secure can only be guaranteed by indicating that assets in the system can be protected from security threats already found. As described in the previous section, security requirements derived from the security design analysis can be decomposed in accordance with the configuration of the system. As a security requirement is a dual of some security threat, the threat can also be decomposed in accordance with the configuration of the system. So security requirements and security threats should be recognized in association with the configuration of a system that is a target of security design analysis. Appropriate traceability should be established among them.
Figure 4 An example of established traceability

Figure 4 shows an example of traceability established among configuration of a system, security threats and security requirements. Configuration of a system is represented by SysML[4] block definition diagrams and requirement diagrams. Security threats are described as constituents which appear in a fault tree (or an attack tree). Security requirements are described in Goal Structuring Notation (GSN for short)[5]. The established traceability is interpreted as in Table 2.

Table 2 The interpretation of traceability

<table>
<thead>
<tr>
<th>Traceability</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>between</td>
<td></td>
</tr>
<tr>
<td>block B</td>
<td>threat T</td>
</tr>
<tr>
<td>threat T</td>
<td>security requirement M</td>
</tr>
<tr>
<td>security requirement M</td>
<td>requirement (in SysML) R</td>
</tr>
<tr>
<td>requirement R</td>
<td>block B</td>
</tr>
<tr>
<td>Interpretation</td>
<td>extracted proposition</td>
</tr>
<tr>
<td>block B is faced with threat T</td>
<td>B implies T</td>
</tr>
<tr>
<td>If security requirement M is satisfied, then threat T does not occur</td>
<td>M implies not T</td>
</tr>
<tr>
<td>If requirement R is satisfied, security requirement M is also satisfied</td>
<td>R implies T</td>
</tr>
<tr>
<td>block B satisfies requirement R</td>
<td>B implies R</td>
</tr>
</tbody>
</table>

3.1 Detection of vulnerable parts in a system

We define that a vulnerable part of a system is a block which does not satisfy enough security requirements to prevent security threats from occurring on the block. In this section, we describe a mechanism to detect vulnerable parts in a system.

Suppose that traceability is established among a block B, a threat T, a security requirement M and a requirement R (see Figure 5).

Figure 5 An example of detection

Then, some propositions can be derived using the interpretation of traceability in Table 2, same propositions can be extracted (Table 3).

Table 3 Extraction of propositions

<table>
<thead>
<tr>
<th>Interpretation</th>
<th>extracted proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>block B is faced with threat T</td>
<td>B implies T</td>
</tr>
<tr>
<td>If security requirement M is satisfied, then threat T does not occur</td>
<td>M implies not T</td>
</tr>
<tr>
<td>If requirement R is satisfied, security requirement M is also satisfied</td>
<td>R implies T</td>
</tr>
<tr>
<td>block B satisfies requirement R</td>
<td>B implies R</td>
</tr>
</tbody>
</table>

B, T, M and R in the extracted propositions are propositional variables which denote ‘existence of block B’, ‘occurrence of threat T’, ‘satisfaction of security requirement M’ and ‘satisfaction of requirement R’, respectively.

If block B is not vulnerable, there is no assignment to the propositional variables B, T, M and R. From the last three propositions in Table 3, we can deduce that B implies not T, which contradicts the first proposition B implies T. If block B is vulnerable, there is some assignment F to B, T, M and R such that F(T) = true. We can easily decide whether there is such assignment or not using SMT solvers, for example Z3[6] and Yices[7].

3.2 An assistant to detect vulnerable parts

In this section, we introduce our tool which assists detection of vulnerable parts in a system. This tool extracts propositions from SysML diagrams (block definition diagram and requirement diagram), fault/attack trees of security threats and GSN expression of security requirements. Propositions extracted by the tool are written in SMT-LIBv2[8] which is a standard input language for SMT solvers.
3.2.1 Inheritance relation between blocks
Suppose that block $B_2$ inherits block $B_1$ (Figure 7).

Then the following proposition (1) is extracted.

$$ \text{B}_2 \Rightarrow \text{B}_1 \quad (1) $$

3.2.2 Traceability between a block and a threat
Suppose that traceability is established between a block $B$ and a threat $T$ (Figure 8).

Then the following proposition (2) is extracted.

$$ \text{B} \Rightarrow \text{T} \quad (2) $$

3.2.3 Causation relation between security threats
Suppose that threat $T$ is caused from threats $T_1$ and … and $T_n$ (Figure 9).

Then the following proposition (3) is extracted.

$$ T \Rightarrow (T_1 \wedge \ldots \wedge T_n) \quad (3) $$

Suppose that threat $T$ is caused from threat $T_1$ or … or $T_n$ (Figure 10).

Then the following proposition (4) is extracted.

$$ T \Rightarrow (T_1 \vee \ldots \vee T_n) \quad (4) $$

3.2.4 Traceability between a threat and a security requirement
Suppose that traceability is established between threat $T$ and security requirement $M$ (Figure 11).

Then, the following proposition (5) is extracted.

$$ M \Rightarrow \neg T \quad (5) $$
3.2.5 Constitution of a security requirement
Suppose that security requirement M is decomposed to sub requirements M₁, … , Mₙ under the premise P (Figure 12).

Figure 12 Constitution of a security requirement

Then, the following proposition (6) is extracted.

\[(P \land M₁ \land \ldots \land Mₙ) \Rightarrow M\]  \hspace{1cm} (6)

3.2.6 Traceability between a security requirement and a requirement
Suppose that traceability is established between security requirement M and requirement R (Figure 13).

Figure 13 Traceability between a security requirement and a requirement

Then the following proposition (7) is extracted.

\[R \Rightarrow M\]  \hspace{1cm} (7)

3.2.7 Relation between requirements
Suppose that requirement R is derived from requirement R₁ (Figure 14).

Figure 14 Derivation relation between requirements

Then the following proposition (8) is extracted.

\[R₁ \Rightarrow R\]  \hspace{1cm} (8)

And suppose that requirement R is composed of R₁, …, Rₙ (Figure 15).

Figure 15 Composition of a requirement

Then the following proposition (9) is extracted.

\[R \Rightarrow (R₁ \land \ldots \land Rₙ)\]  \hspace{1cm} (9)

3.2.8 Traceability between a block and a requirement
Suppose that traceability is established between block B and requirement R (Figure 16).

Figure 16 Satisfaction relation between a block and a requirement

Then the following proposition (10) is extracted.

\[B \Rightarrow R\]  \hspace{1cm} (10)

4 Example
The validity of the proposed mechanism is checked by several examples. Figure 17 indicates one of such examples. This example treats the medication notebook system[1]. Pharmacies and mobile terminals in the system are instances of the block Client. In the example, a security threat is that prescription records can be read during a communication on the internet by a third person. A security requirement against the threat consists of three sub-requirements. In the following figure, dotted lines indicate traceability established in the system.
According to the proposed mechanism introduced in the section 3, there is no vulnerable part in the example because each block satisfies enough security requirements to prevent the security threat. However, when some traceability among the threat, the security requirements and the requirements (in SysML requirement diagram) is removed, we can realize that the threat can occur on the system, because SMT solvers find a assignment where a propositional variable which represents the occurrence of the threat to be true.

5 Conclusions

We proposed a mechanism and introduced a tool to support security design analysis by detecting vulnerable parts of a system using SMT solvers. We are going to apply the mechanism to some systems of the real world to validate the usefulness of the proposed mechanism for security design analysis.

6 Acknowledgement

This research was supported by the Ministry of Internal Affairs and Communications SCOPE( Strategic Information and Communications R&D Promotion Programme) No. 142310011.

7 References


