Eliciting Security Requirements for an Information System using Asset Flows and Processor Deployment

Haruhiko Kaiya, Department of Computer Science, Shinshu University, Nagano, Japan
Junya Sakai, Department of Computer Science, Shinshu University, Nagano, Japan
Shinpei Ogata, Department of Computer Science, Shinshu University, Nagano, Japan
Kenji Kaijiri, Department of Computer Science, Shinshu University, Nagano, Japan

ABSTRACT

The authors cannot comprehensively determine all of the vulnerabilities to an attack only from requirements descriptions. To resolve the problem, the authors propose a method for eliciting security requirements using the information about system architecture. The authors convert a use-case description into a variation of a data flow diagram called an asset-flow diagram (AFD). The authors then refine the AFDs based on a processor deployment diagram (PDD), which gives information about a system architecture. By using vulnerabilities patterns to an attack, the authors distinguish vulnerabilities to the attack that can be identifiable in AFDs from remaining vulnerabilities to the attack. To prohibit the former vulnerabilities, security requirements are defined as countermeasures and/or modification of existing requirements. To prevent the latter vulnerabilities, security requirements are defined as design and implementation constraints. Through an evaluation of a web application, the authors show that our method enables us to elicit security requirements against several different attacks in different system architectures.

Keywords: Asset, Countermeasure, Information Security, Security Prohibitive and Preventive Requirements, System Architecture, Use-Case Model, Vulnerability

1. INTRODUCTION

Information security means protection of information (ISO Standard, 2005), and information is a kind of asset, i.e., that has value to an organization (ISO Standard, 2004). It is important to take information security into account in the early stages of system development because security issues have a big impact on development costs, system performance, and even the fundamental functional requirements of users. Requirements engineering focuses to a large

DOI: 10.4018/jsse.2013070103
extent on stakeholders, especially customers and
users, and the viewpoints of the asset holders
and the attackers are dealt with in the security
requirements analysis (Sindre & Opdahl, 2005),
(Lamsweerde, 2004), (Liu, Yu, & Mylopoulos,
2003), (Giorgini, Massacci, Mylopoulos, &
Zannone, 2005). However, the question of how
to handle assets in an information system has
rarely been discussed in security requirements
engineering. One of the reasons is that many
researchers in this field think this question is out
of the scope of requirements analysis because it
is about the design and/or architecture of
the system. Another reason is that well-known
notations such as the use-case model and goal
model are not good at handling many kinds of
assets. However, security requirements analysis
cannot be effective without considering the
design and architecture to be used because the
functions and attributes of the implemented
system potentially threaten the assets.

Recently, several researchers tried to handle
security requirements and design/architectural
issues together (Heyman, Yskout, Scandariato,
Schmidt, & Yu, 2011; Okubo, Kaiya, & Yosh-
oki, 2012). In this paper, we also propose a
method for eliciting security requirements by
taking design and architectural issues into ac-
count. We call the method the Cause-Oriented
Vulnerability Analysis (COVA) because remov-
ing and preventing vulnerabilities is the main
reason for protecting assets. People entrust a lot
of assets to computer systems to facilitate their
daily activities. This is one of the main reasons
why we develop the system. For example, some
people may entrust their healthcare data to a
healthcare information system for the sake of ef-
ficient healthcare, or delegate the right of dealing
in stocks to an automatic stock dealing system.
We think that the most fundamental cause of
threats to assets is how we trust and delegate
assets to computers. This means we have to
focus on the flows of the assets in a computer
system. The parts of such flows that can be used
in an attack can be regarded as vulnerabilities.
The role of security requirements is to specify
functions and constraints that will prohibit or
prevent such vulnerabilities. However, not all
vulnerabilities can be identified from funda-
mental requirements such as the necessary
functions for users. Moreover, distinguishing
vulnerabilities that have already been embed-
ded in requirements from vulnerabilities that
will emerge in the later development phases is
important because their security requirements
differ; i.e., the former should be prohibited
and the latter should be prevented. COVA has
been designed with the above rationale in mind.
The main contributions of COVA to eliciting
security requirements are as follows. First, we
can systematically narrow down the suspected
vulnerabilities by determining which attacks
pose threats. Second, we can systematically
elicit security requirements for prohibiting
vulnerabilities caused by existing requirements.
Third, we can systematically elicit security
requirements for preventing vulnerabilities
designed and/or implemented in the system.

The rest of the paper is organized as fol-
lows. In the next section, we briefly describe
a few well-known attacks that will be used to
explain our method. In Section 3, we explain
COVA. In Section 4, we evaluate COVA as to
whether it can thwart these attacks in different
system architectures. We review related work
in Section 5, and we summarize our results and
outline future issues in Section 6.

2. SOME WELL-
KNOWN ATTACKS

The following attacks on Web applications were
taken from a web site on Web application risks
(OWASP, 2010):

- **Injection:** An attacker sends commands
to the interpreter in a system that he has
no permission to access. A lot of web ap-
plications use interpreters for database
operations such as SQL commands. If a
user can input characters freely and the
interpreter directly interprets the input as
a command, he or she can operate on the
Improvement of Estimation of Objective Scores of Answer Statements Posted at Q&A Sites
Yuya Yokoyama, Teruhisa Hochin and Hiroki Nomiya (2013). International Journal of Software Innovation (pp. 16-30).
www.igi-global.com/article/improvement-of-estimation-of-objective-scores-of-answer-statements-posted-at-qa-sites/105629?camid=4v1a

Efficient Software Quality Assurance Approaches Oriented to UML Models in Real Life
www.igi-global.com/chapter/efficient-software-quality-assurance-approaches/30757?camid=4v1a