A Metamodel for Security and Privacy Knowledge in Cloud Services

Hironori Washizaki, Sota Fukumoto, Misato Yamamoto, Masatoshi Yoshizawa, Yoshiaki Fukazawa (Waseda U.), Takehisa Kato (Toshiba), Shinpei Ogata (Shinshu U.), Haruhiko Kaiya (Kanagawa U.), Eduardo B. Fernandez (FAU), Hideyuki Kanuka, Yuki Kondo (Hitachi), Nobukazu Yoshioka (NII), Takao Okubo (IIS), Atsuo Hazeyama (Tokyo Gakugei U.)
Challenges in Cloud Security and Privacy (S&P)

• How to consistently utilize existing diverse S&P knowledge?
  ⇒ Metamodel

• How to consider S&P over different layers?
  ⇒ Layered metamodel (and knowledge-base)

Patterns

Guidelines

Practices

Cloud services

Ex.) User Authentication

Software Application

User Authorization

Platform

Secure Config. OS Hardening

Infrastructure

Electronic Access Control system
6+1 Pieces: Layered and Modularized Metamodel for Cloud

• Incorporating existing metamodels [Fer][Hazeyama][Kalloniatis][Tesoriero] and reference architectures [NIST]

• Target: Goals and domains
• Bridge: Relation between problems and solutions

Overview of 6+1 Pieces

Problem

Bridge

Solution

Target

SaaS (Application)

PaaS (Platform)

IaaS (Infrastructure)
Problem

- Specify common concepts for S&P problems
- Cloud-independent
Solution

- Specify common concepts for S&P solutions
- Cloud-independent
Bridge

- Specify connections between problems and solutions
- Cloud-independent
Software application and platform

- Specify common concepts for software application and platform
- Cloud-specific
• Specify common concepts for infrastructure
• Cloud-specific
Possible usecases

• UC1. Categorizing knowledge
  – Knowledge-base
  – OWL, RDF?

• UC2. Representing, sharing and utilizing individual knowledge

• UC3. Representing, sharing and utilizing result of knowledge application
Initial knowledge-base

**Misuse pattern/SQL Injection**

- **Name**: SQL Injection
- **Pattern**: Misuse pattern
- **Cloud Service**: SSL
- **Vulnerability**: 
- **Explanation**: This attack exploits target software that constructs SQL statements based on user input. An attacker crafts input strings so that when the target software constructs intended SQL injection results from failure of the application to appropriately validate input. When specially crafted user-controlled input consisting of SQL syntax is not envisaged during application design, depending upon the database and the design of the application, it may also be possible to leverage injection to have the attacker bypass the application completely. Successful injection can cause information disclosure as well as ability to add or modify data in the database.
- **Solution**: Strong input validation - All user-controllable input must be validated and filtered for illegal characters as well as SQL content. Keywords such as UNION, SELECT or other context in which they appear.
- **Goal**: Any attacker is able to write past the boundaries of allocated buffer regions in memory, causing a program crash or potentially redirection of execution as per the attacker's desire. In addition, it may also be possible to leverage injection to have the attacker gain access as well as ability to add or modify data in the database.
- **Document**: CAPEC
- **Related patterns**

**Related Pattern**
Case Study: Modeling Patterns

• Misuse: Session Hijacking Attack Pattern
• Solution: Security Session Pattern
Conclusion and Discussion

• Contribution
  – Metamodel for addressing S&P in cloud services and its simple case study
  – Simple case study to show how metamodel is used for modeling patterns

• Discussion
  – Missing any important concepts?
  – Does the metamodel contribute to utilizing knowledge across layers?
  – How can we build useful knowledge-base upon the metamodel?
  – Are some packages reusable for any platform?
  – How about complex cases needing various patterns, guidelines and practices?