SSE Workshop @ NII, Mar 22-23, 2016

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.)

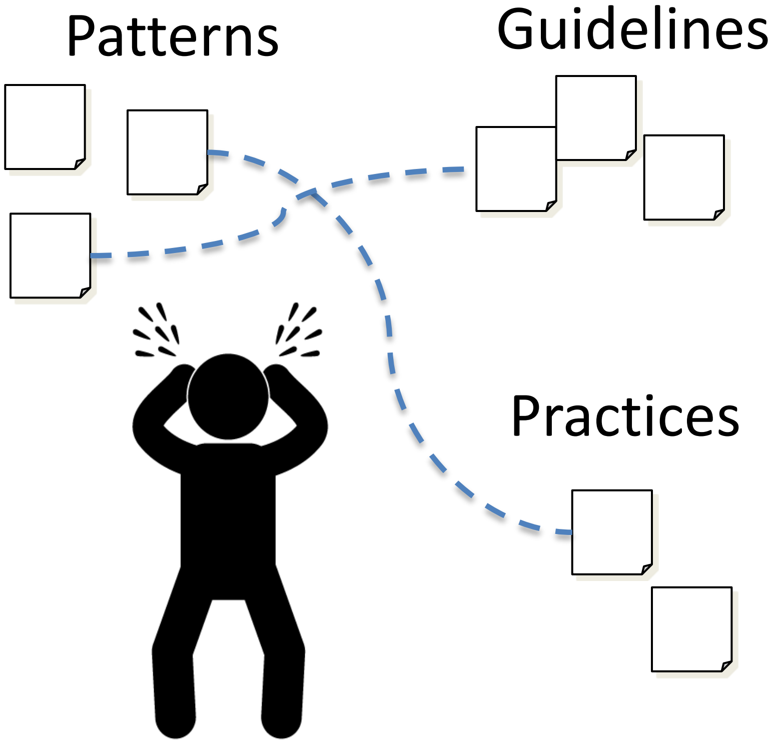
Challenges in Cloud Security and Privacy (S&P)

- How to consistently utilize existing diverse S&P knowledge?

⇒ Metamodel

- How to consider S&P over different layers?

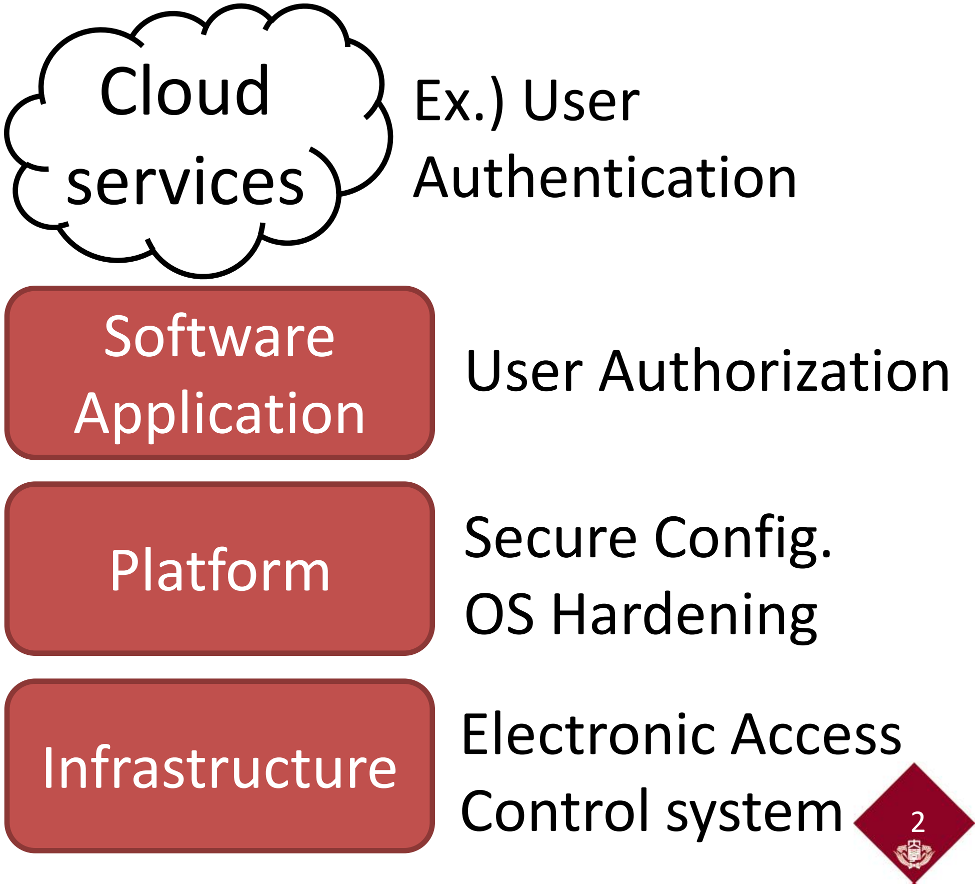
⇒ Layered metamodel (and knowledge-base)



Patterns

Guidelines

Practices



Cloud services

Ex.) User Authentication

Software Application

User Authorization

Platform

Secure Config.
OS Hardening

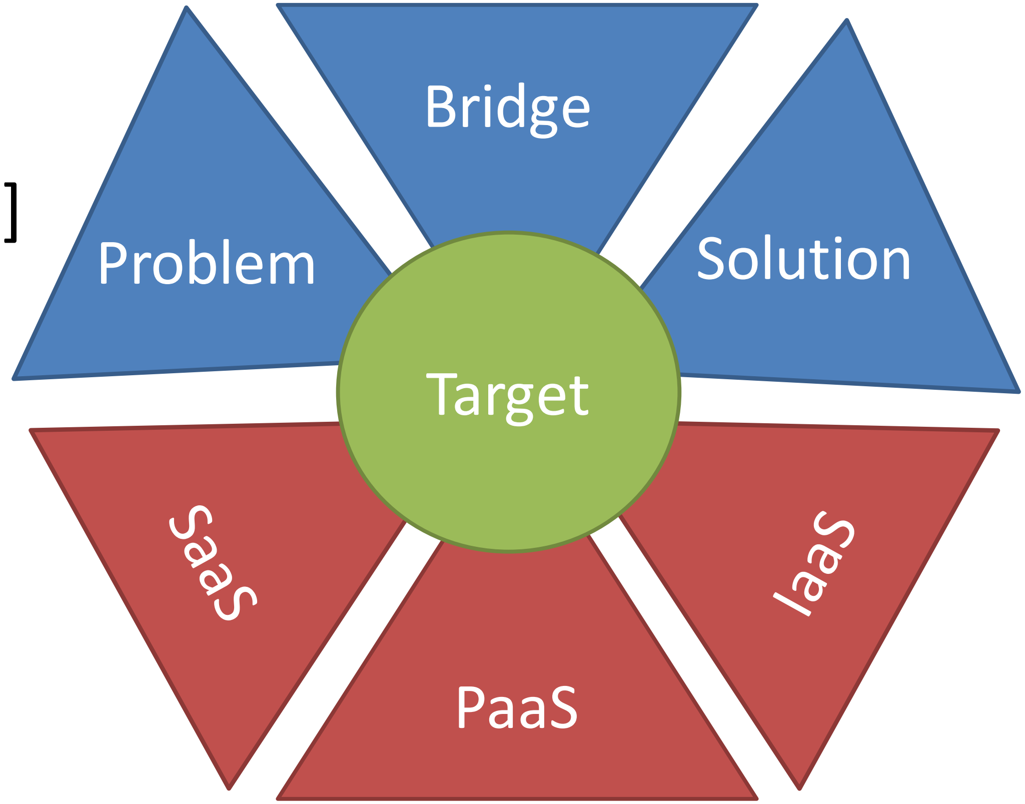
Infrastructure

Electronic Access
Control system

6+1 Pieces: Layered and Modularized Metamodel for Cloud

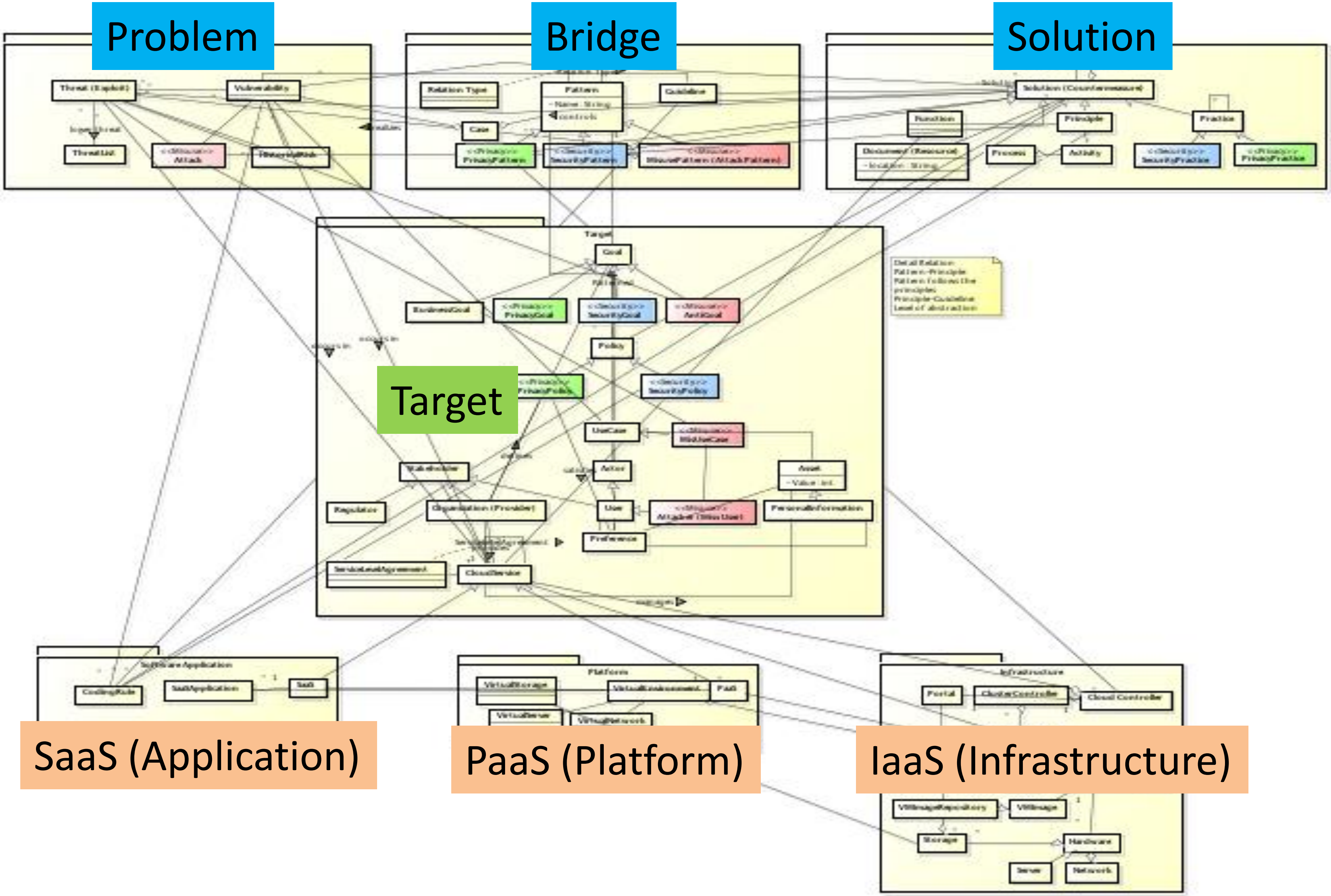
- Incorporating existing metamodels [Fer][Hazeyama][Kalloniatis][Tesoriero] and reference architectures [NIST]

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



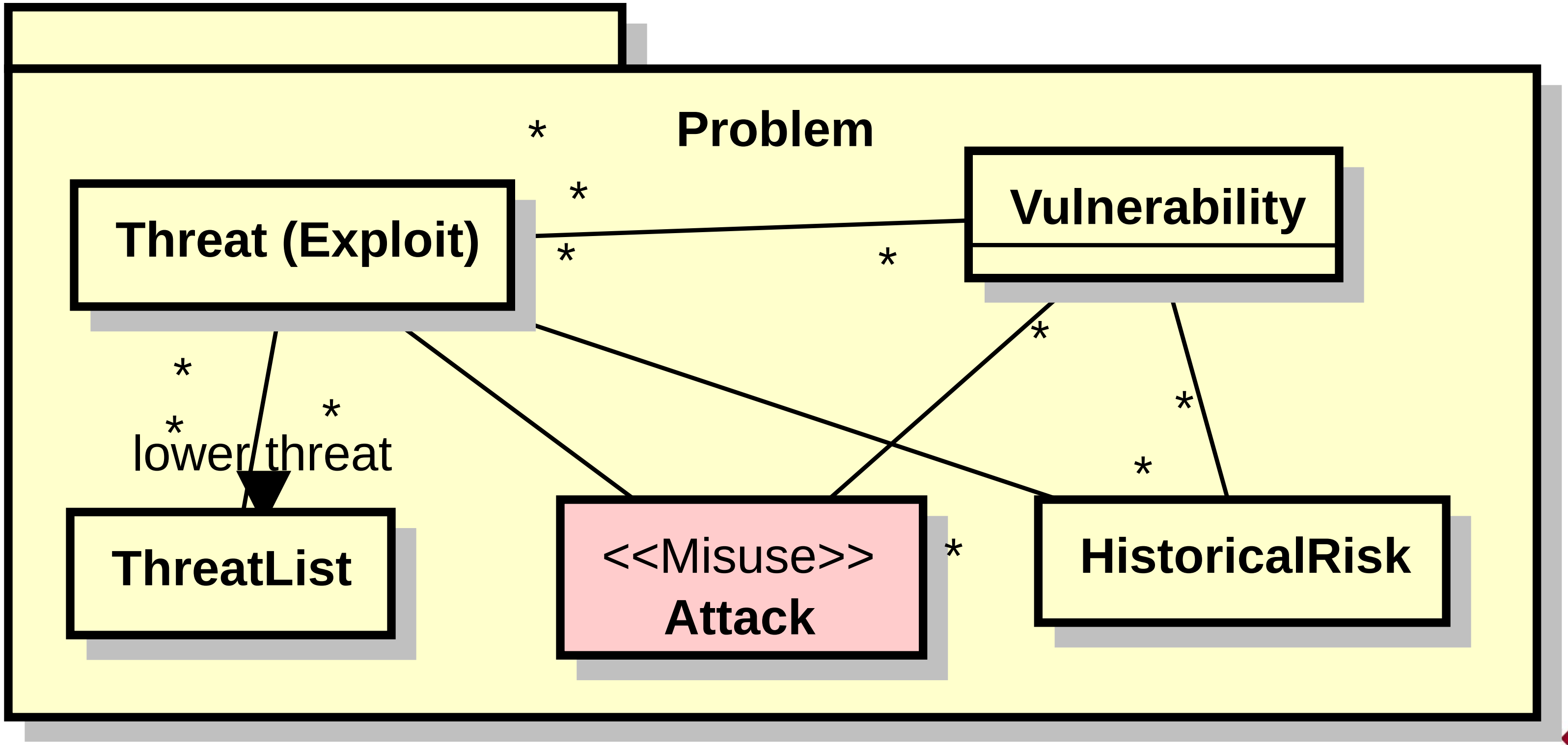
E. B. Fernandez, et al, "Building a security reference architecture for cloud systems," Requirements Engineering, 2015
A. Hazeyama, “Survey on Body of Knowledge Regarding Software Security,” SNPD 2012
NIST Cloud Computing Security WG, "Cloud computing security reference architecture," 2013
C. Kalloniatis, et al., “Addressing privacy requirements in system design: the pris method,” Requirements Engineering, 13(3), 2008
R. Tesoriero, et al. “Model-Driven Privacy and Security in Multi-modal Social Media Uis,” MSM 2011

Overview of 6+1 Pieces



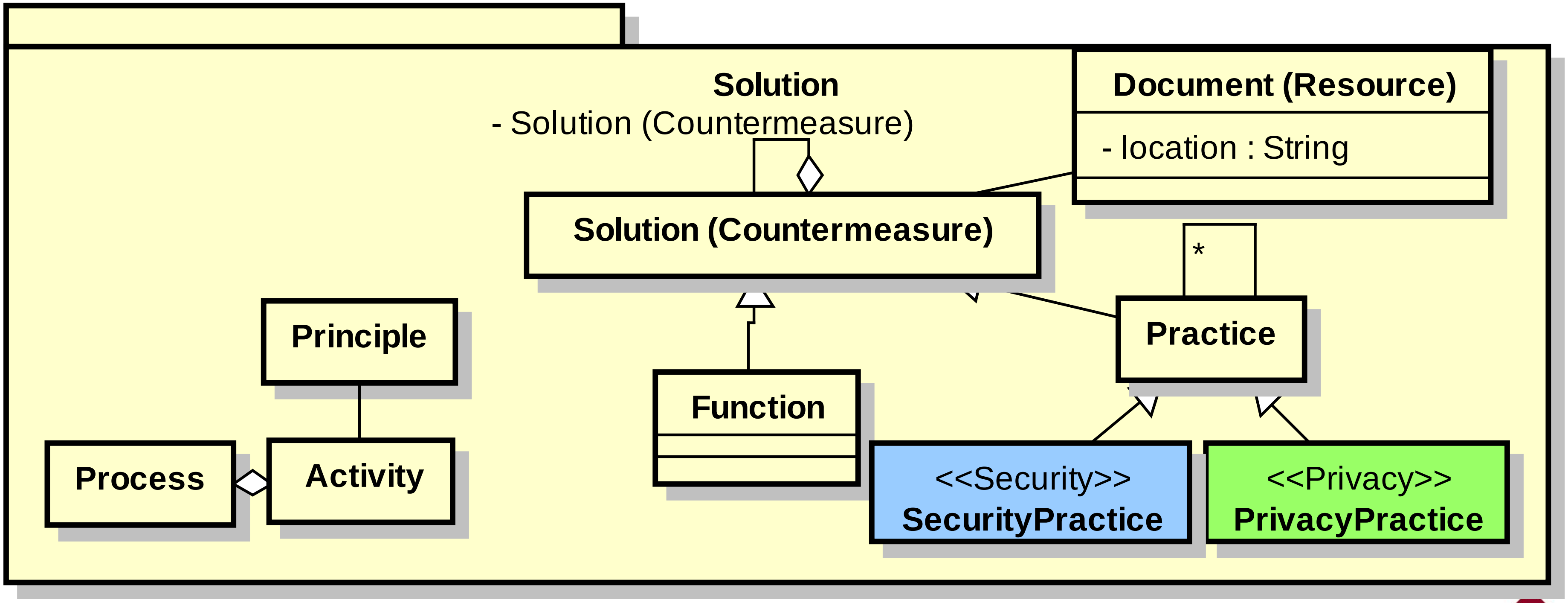
Problem

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



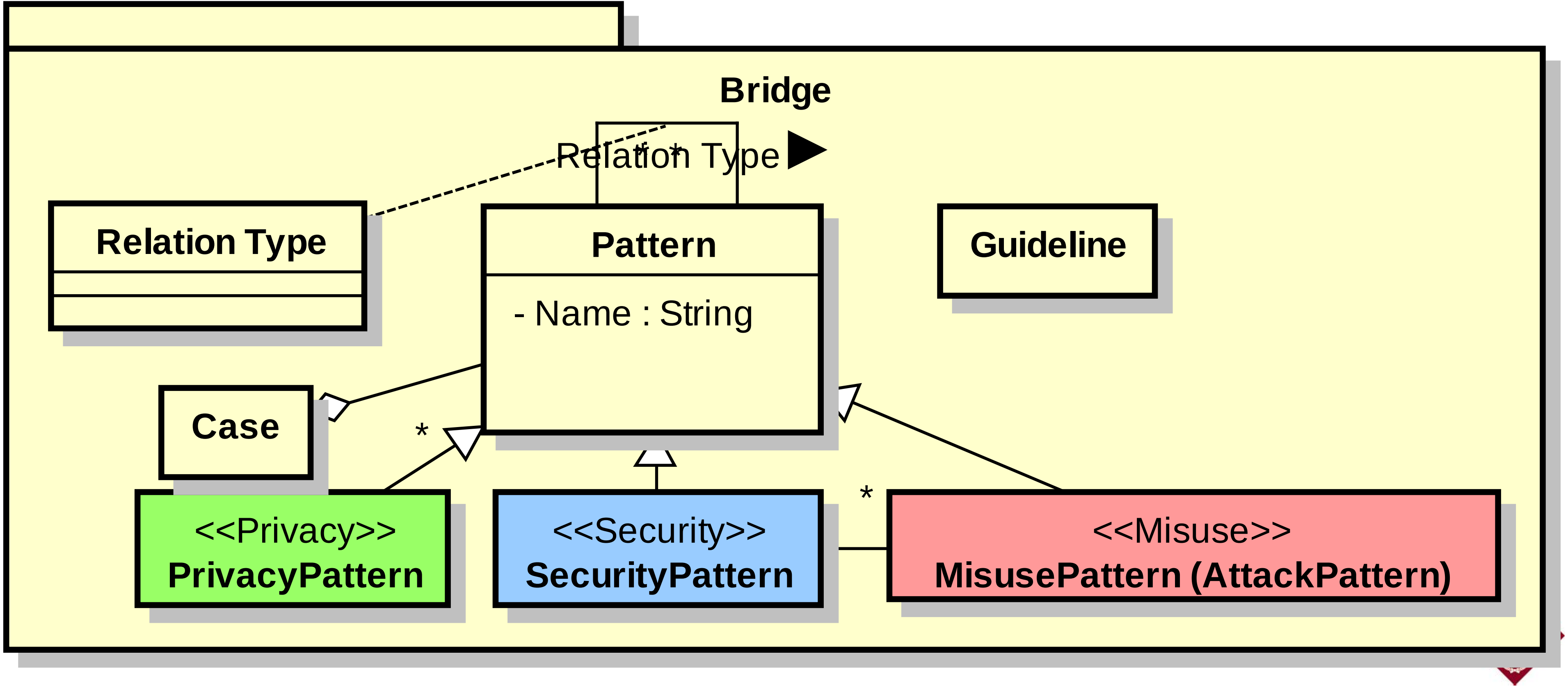
Solution

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



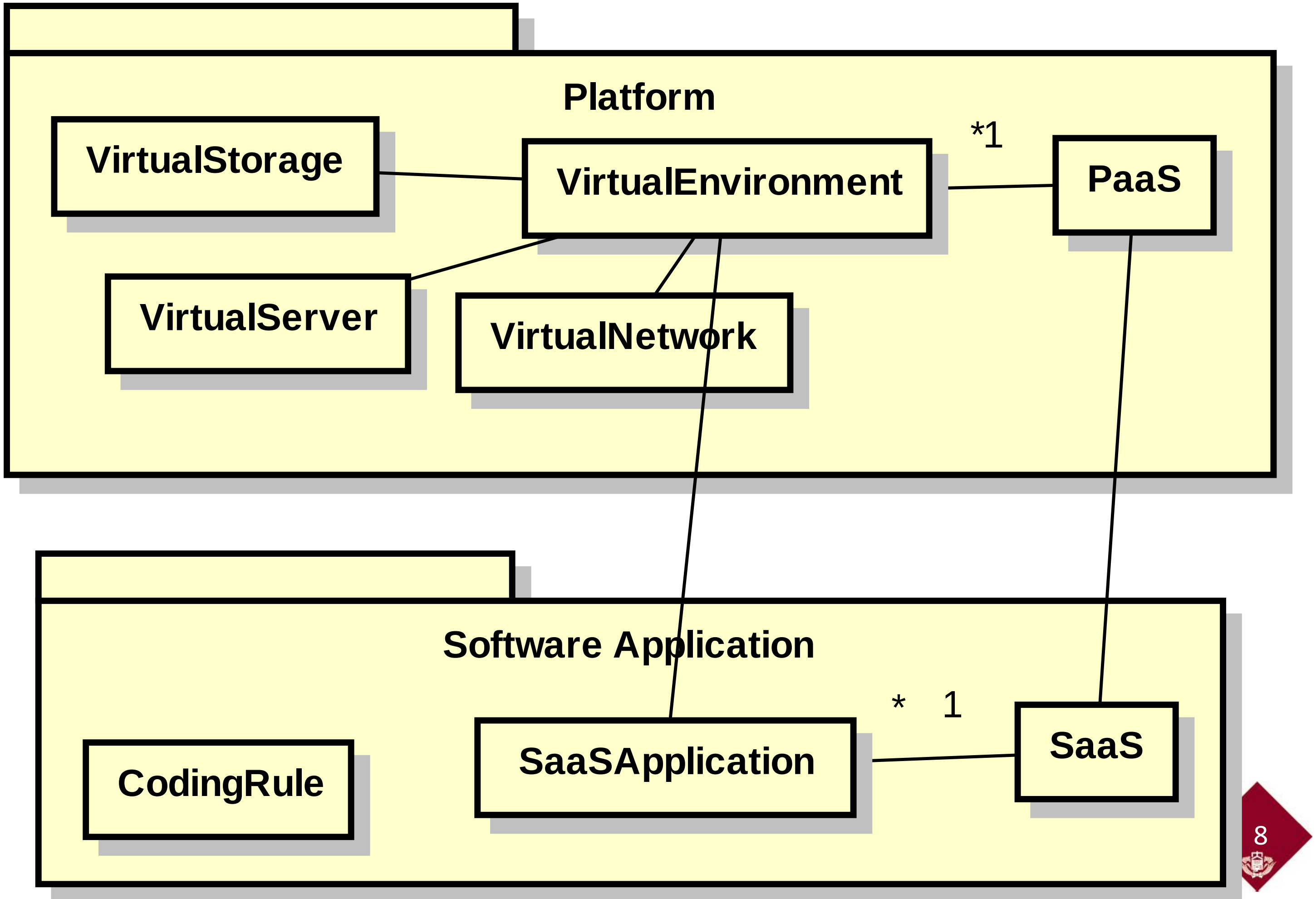
Bridge

- Specify connections between problems and solutions
- Cloud-independent



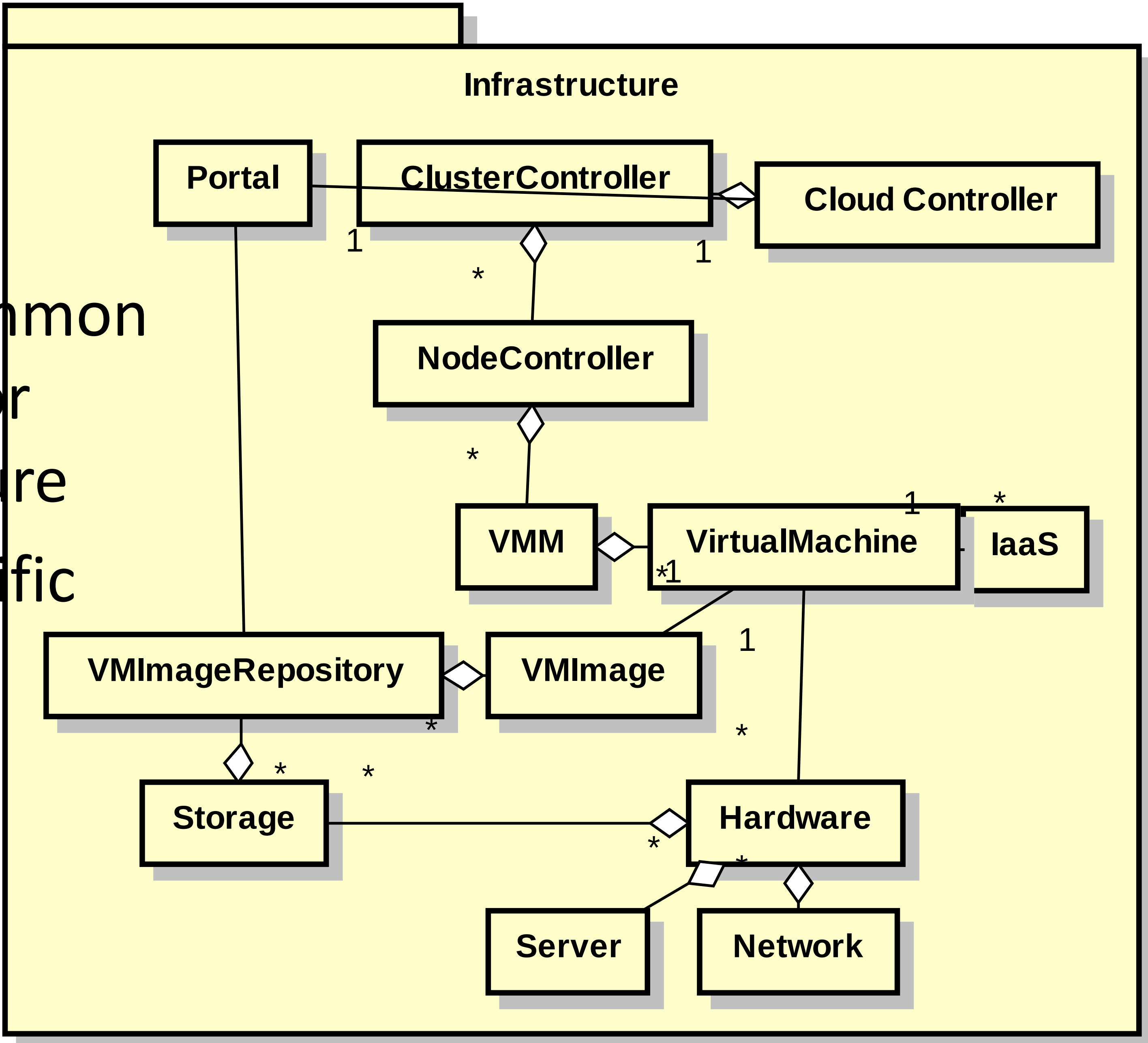
Software application and platform

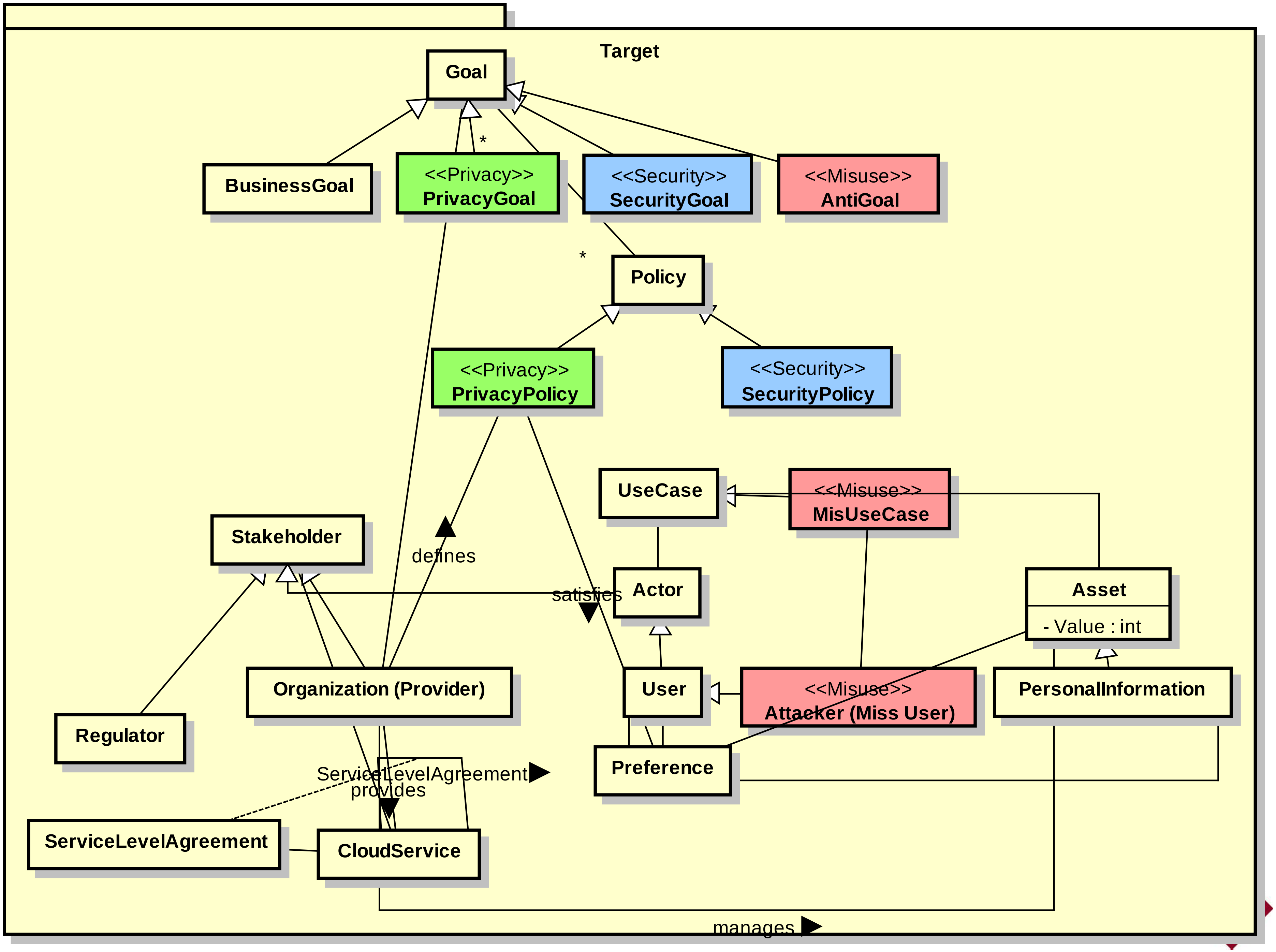
- Specify common concepts for software application and platform
- Cloud-specific



Infrastructure

- Specify common concepts for infrastructure
- Cloud-specific

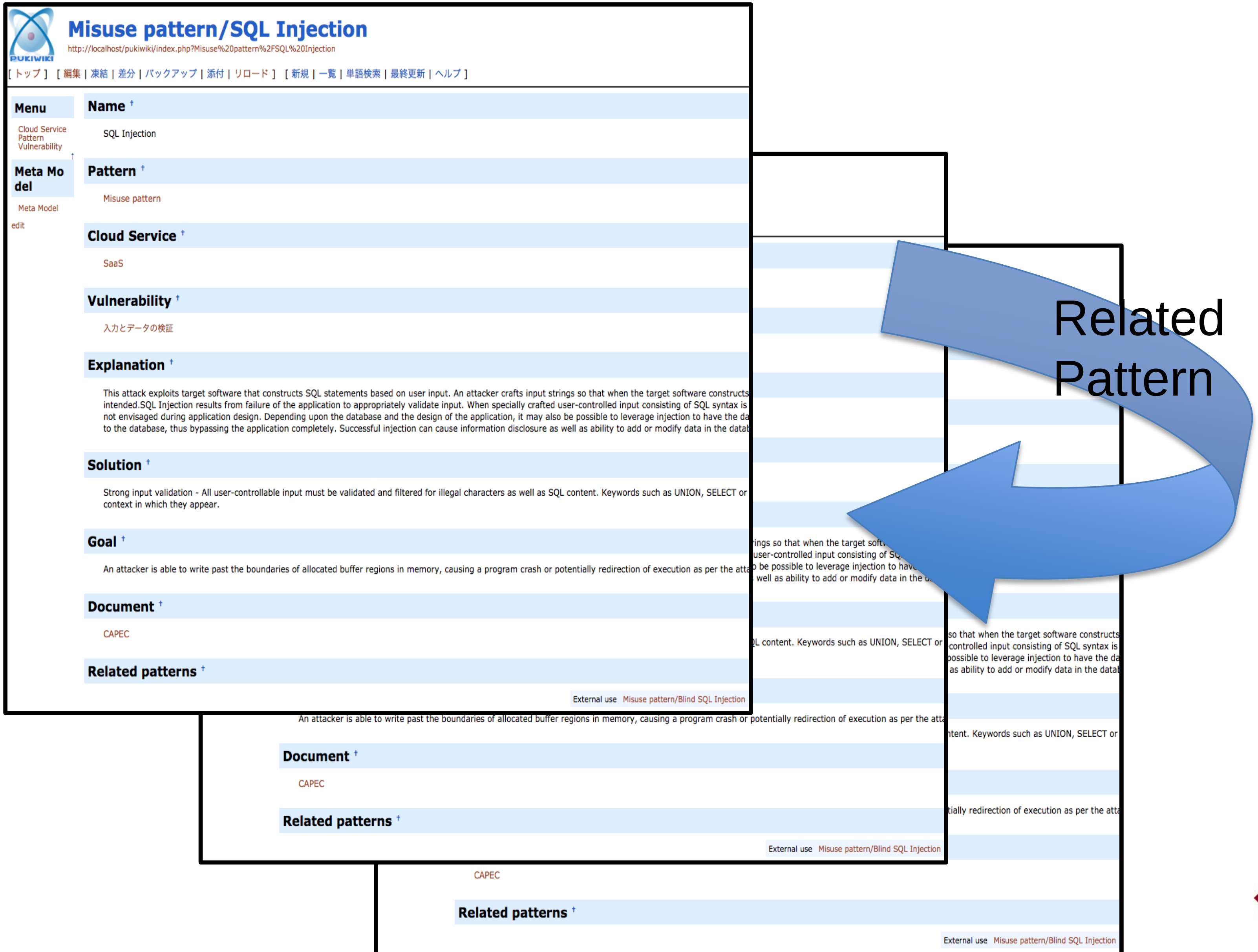



Target
Goal
BusinessGoal
<<Privacy>>
PrivacyGoal
<<Security>>
SecurityGoal
<<Misuse>>
AntiGoal
*
*
Policy
<<Privacy>>
PrivacyPolicy
<<Security>>
SecurityPolicy
UseCase
<<Misuse>>
MisUseCase
Stakeholder
defines
satisfies
Actor
Asset
- Value : int
Organization (Provider)
User
<<Misuse>>
Attacker (Miss User)
PersonalInformation
Regulator
Preference
ServiceLevelAgreement
provides
ServiceLevelAgreement
CloudService
manages

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

http://localhost/pukiwiki/index.php?Misuse%20pattern%2FSQL%20Injection

[トップ] [編集 | 凍結 | 差分 | バックアップ | 添付 | リロード] [新規 | 一覧 | 単語検索 | 最終更新 | ヘルプ]

Menu

Cloud Service
Pattern
Vulnerability

Meta Model

Meta Model

edit

Name †

SQL Injection

Pattern †

Misuse pattern

Cloud Service †

SaaS

Vulnerability †

入力とデータの検証

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 da to the database, thus bypassing the application completely. Successful injection can cause information disclosure as well as ability to add or modify data in the datab

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 context in which they appear.

Goal †

An 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 atta

Document †

CAPEC

Related patterns †

External use Misuse pattern/Blind SQL Injection

An 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 atta

Document †

CAPEC

Related patterns †

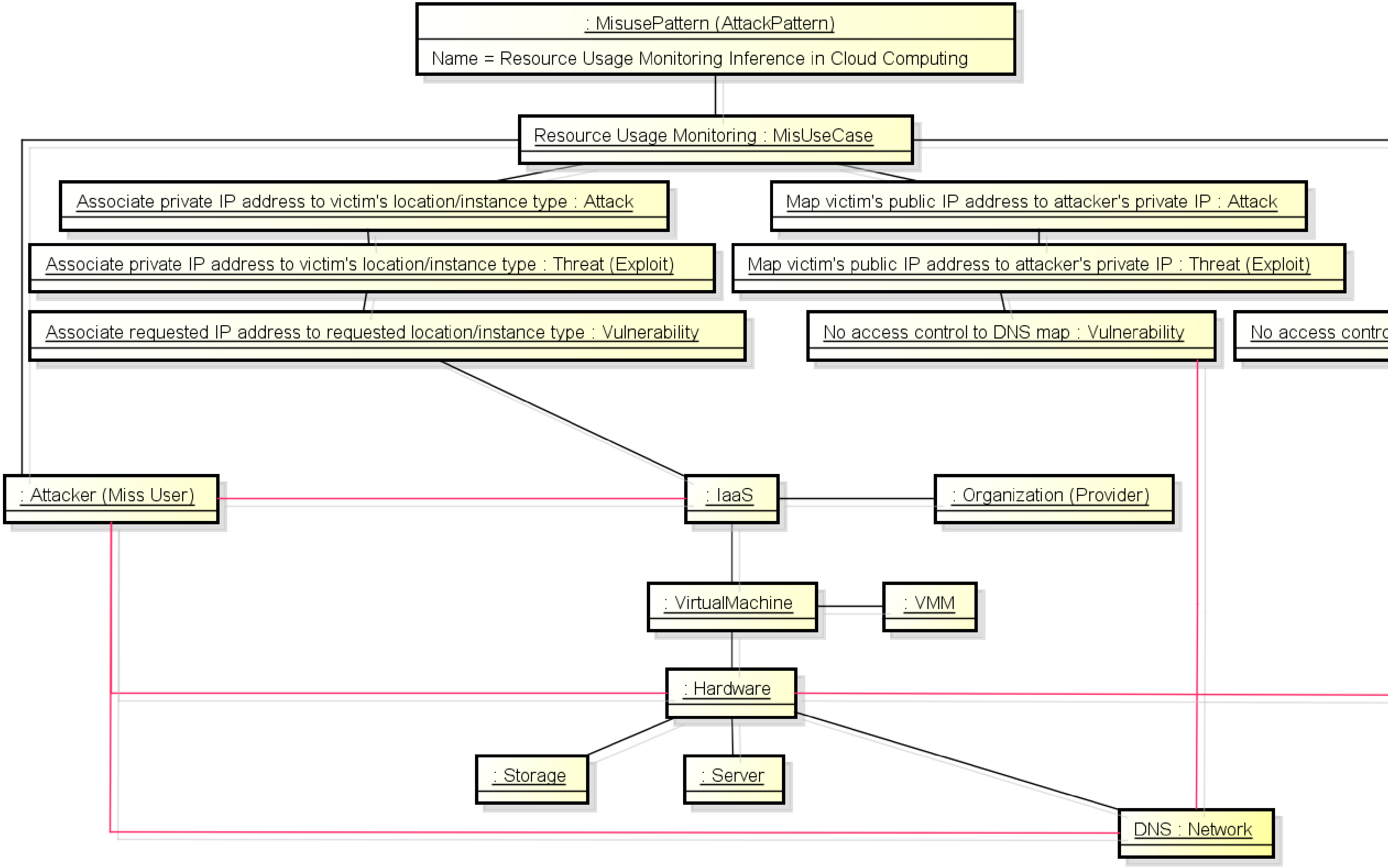
External use Misuse pattern/Blind SQL Injection

CAPEC

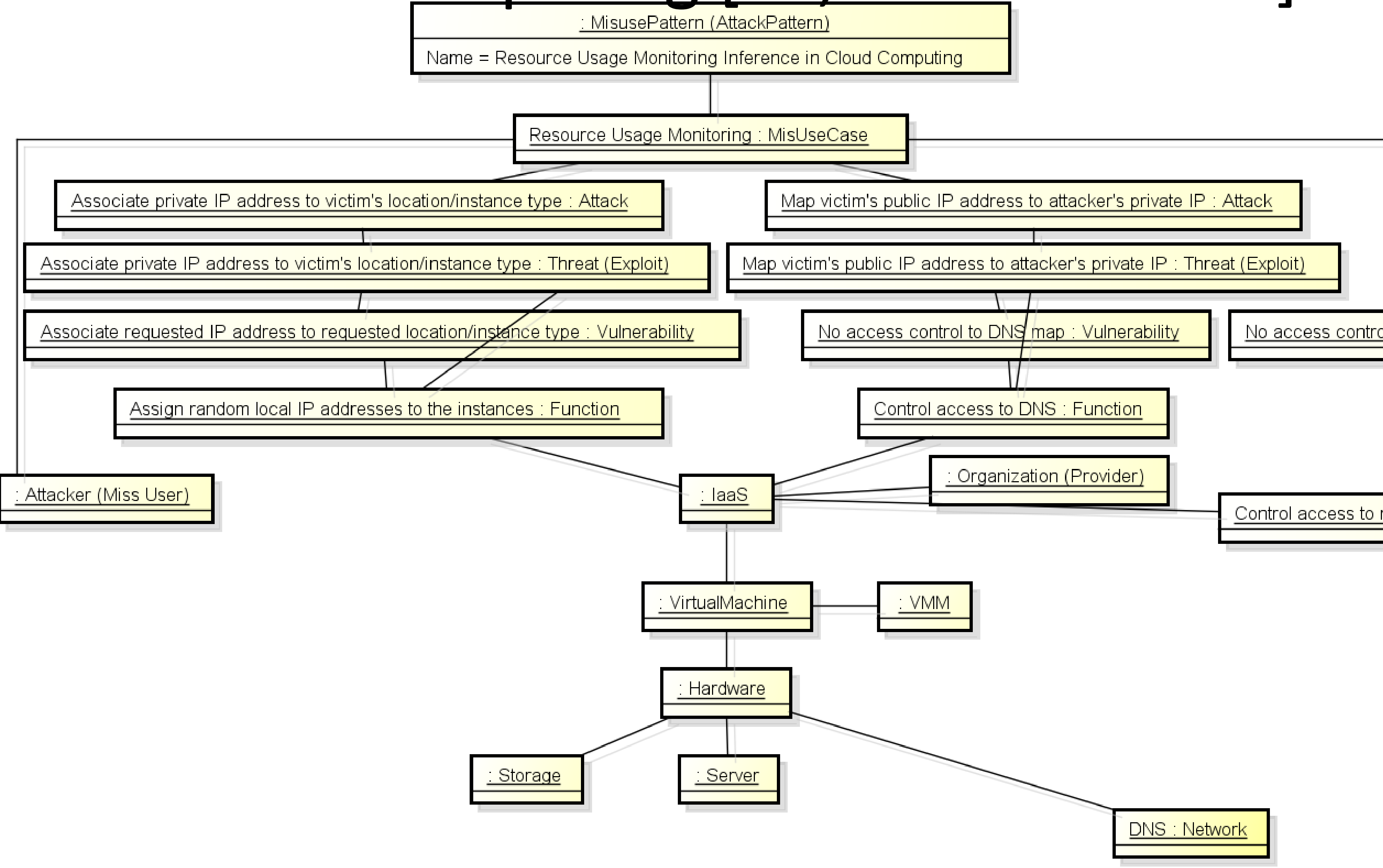
Related patterns †

External use Misuse pattern/Blind SQL Injection

Resource Usage Monitoring Inference in Cloud Computing [Ed, AsianPLoP'11]


: MisusePattern (AttackPattern)
Name = Resource Usage Monitoring Inference in Cloud Computing
Resource Usage Monitoring : MisUseCase
Associate private IP address to victim's location/instance type : Attack
Map victim's public IP address to attacker's private IP : Attack
Associate private IP address to victim's location/instance type : Threat (Exploit)
Map victim's public IP address to attacker's private IP : Threat (Exploit)
Associate requested IP address to requested location/instance type : Vulnerability
No access control to DNS map : Vulnerability
No access contro
: Attacker (Miss User)
: IaaS
: Organization (Provider)
: VirtualMachine
: VMM
: Hardware
: Storage
: Server
DNS : Network

Resource Usage Monitoring Inference in Cloud Computing [Ed, AsianPLoP'11]



: MisusePattern (AttackPattern)

Name = Resource Usage Monitoring Inference in Cloud Computing

Resource Usage Monitoring : MisUseCase

Associate private IP address to victim's location/instance type : Attack

Map victim's public IP address to attacker's private IP : Attack

Associate private IP address to victim's location/instance type : Threat (Exploit)

Map victim's public IP address to attacker's private IP : Threat (Exploit)

Associate requested IP address to requested location/instance type : Vulnerability

No access control to DNS map : Vulnerability

No access contro

Assign random local IP addresses to the instances : Function

Control access to DNS : Function

: Attacker (Miss User)

: IaaS

: Organization (Provider)

Control access to r

: VirtualMachine

: VMM

: Hardware

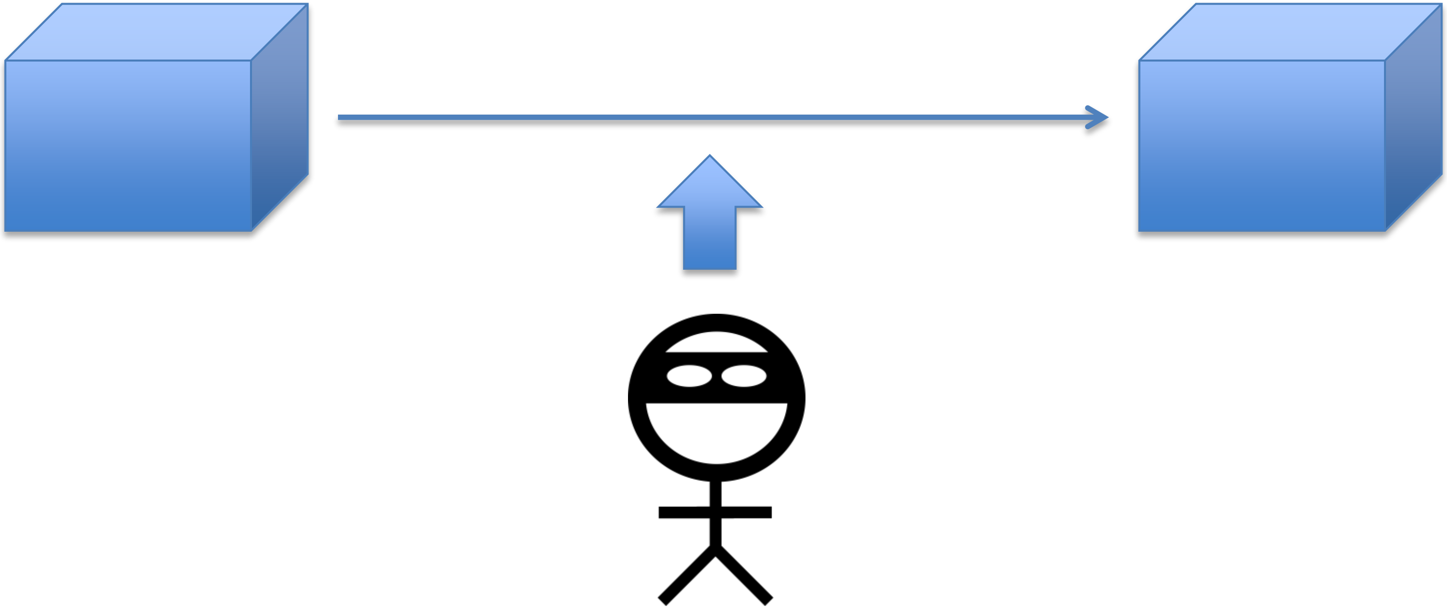
: Storage

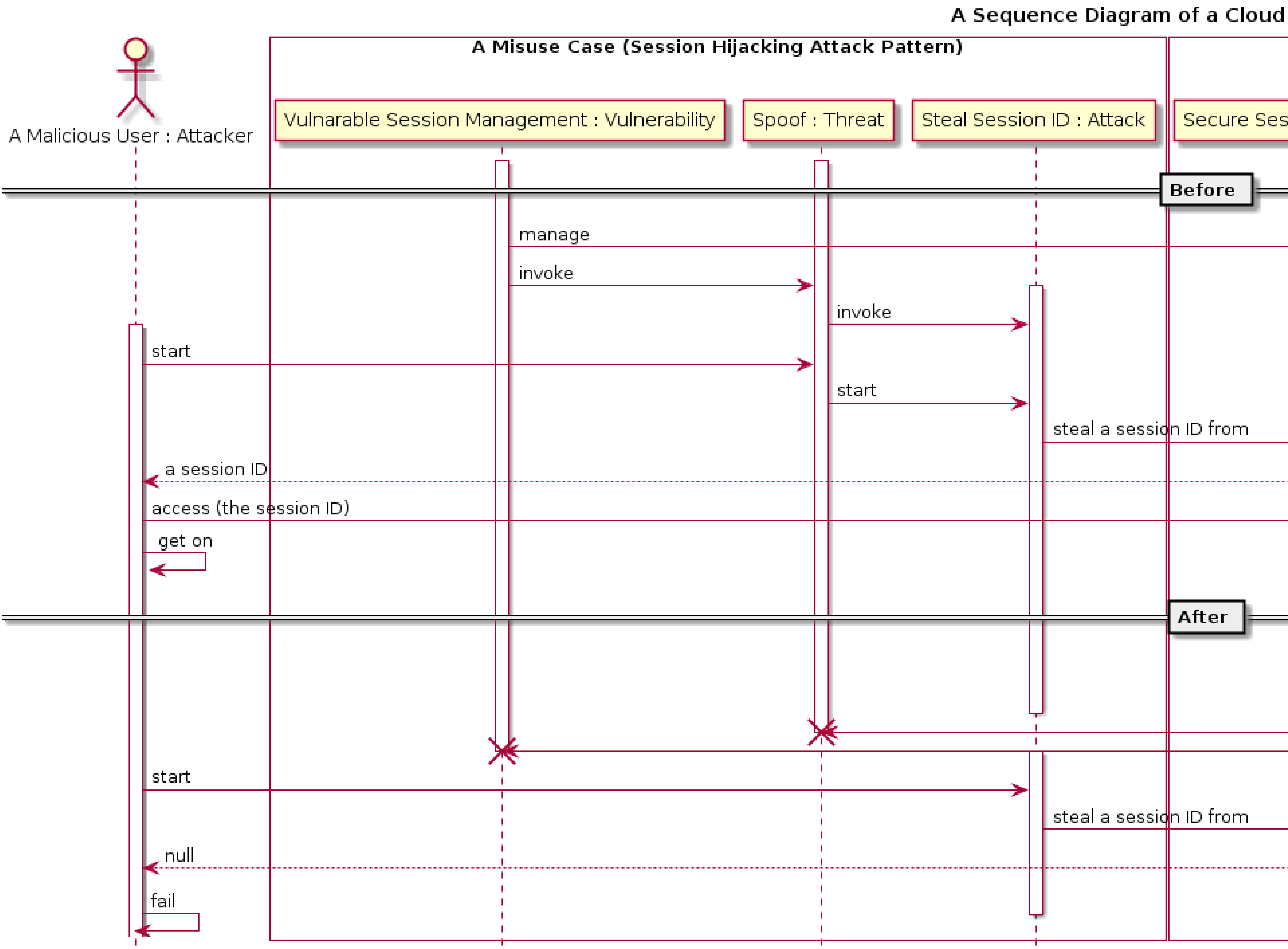
: Server

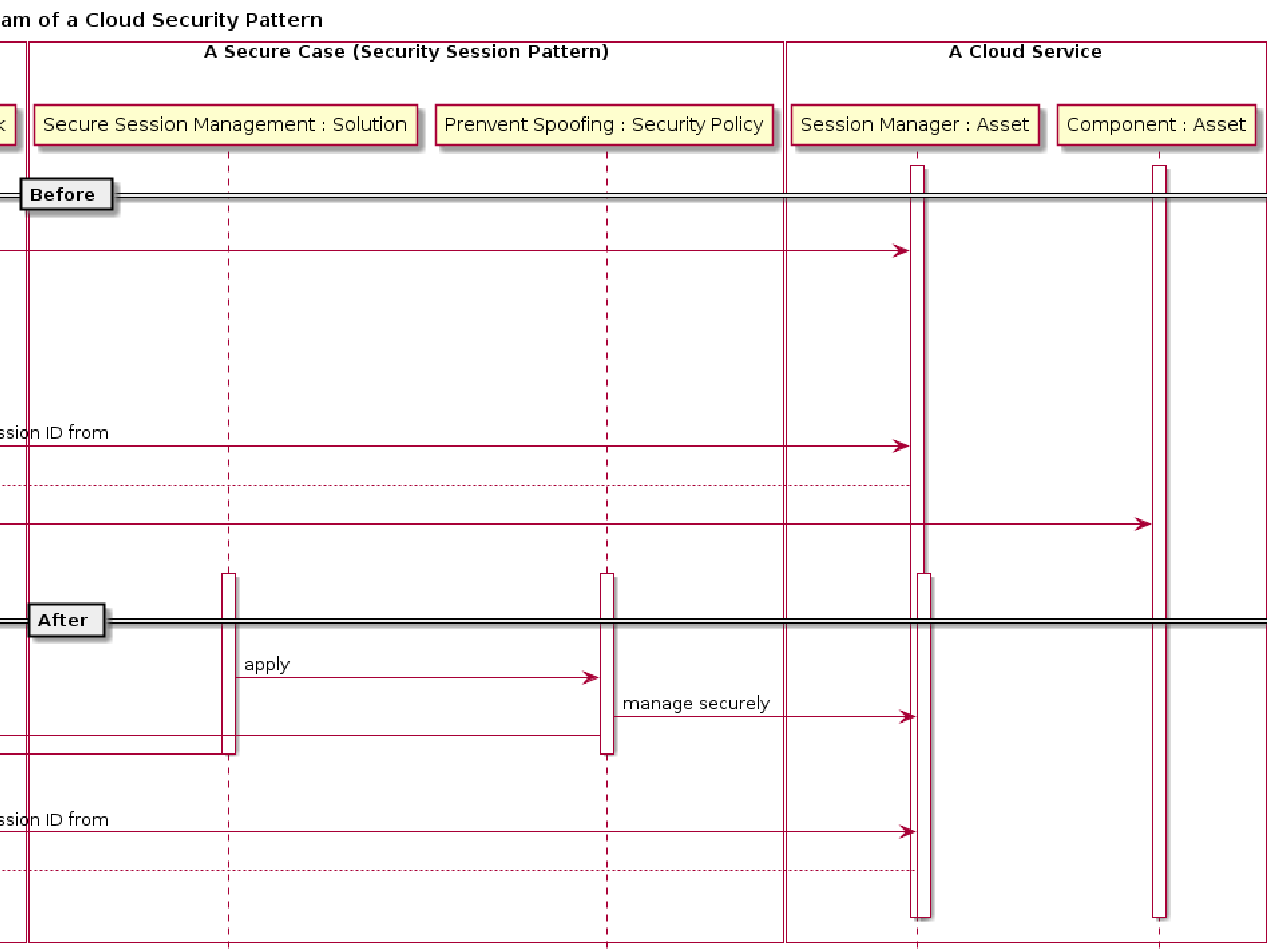
DNS : Network

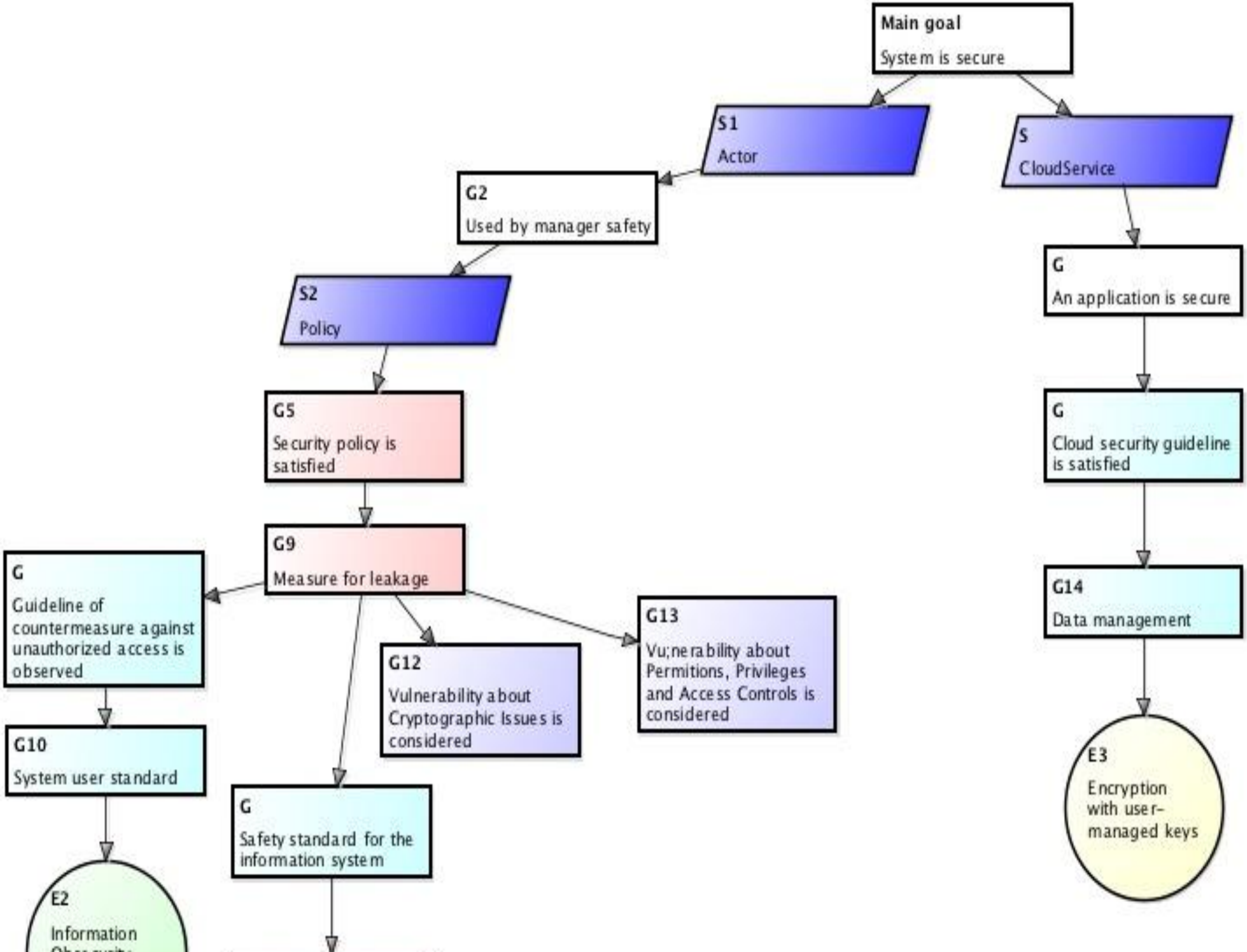
Case Study: Modeling Patterns

- Misuse: Session Hijacking Attack Pattern
- Solution: Security Session Pattern




A Sequence Diagram of a Cloud
A Misuse Case (Session Hijacking Attack Pattern)
A Malicious User : Attacker
Vulnarable Session Management : Vulnerability
Spoof : Threat
Steal Session ID : Attack
Secure Ses
Before
manage
invoke
invoke
start
start
steal a session ID from
a session ID
access (the session ID)
get on
After
start
steal a session ID from
null
fail


am of a Cloud Security Pattern
A Secure Case (Security Session Pattern)
A Cloud Service
k
Secure Session Management : Solution
Prenvent Spoofing : Security Policy
Session Manager : Asset
Component : Asset
Before
ssion ID from
After
apply
manage securely
ssion ID from


Main goal
System is secure
S1
Actor
S
CloudService
G2
Used by manager safety
S2
Policy
G
An application is secure
G5
Security policy is satisfied
G
Cloud security guideline is satisfied
G9
Measure for leakage
G
Guideline of countermeasure against unauthorized access is observed
G13
Vu;nerability about Permitions, Privileges and Access Controls is considered
G14
Data management
G12
Vulnerability about Cryptographic Issues is considered
G10
System user standard
G
Safety standard for the information system
E3
Encryption with user-managed keys
E2
Information

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?