A SECURITY REFERENCE ARCHITECTURE FOR CLOUD SYSTEMS

Eduardo B. Fernandez

Dept. of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL, USA

http://www.cse.fau.edu/~ed

ed@cse.fau.edu

About me

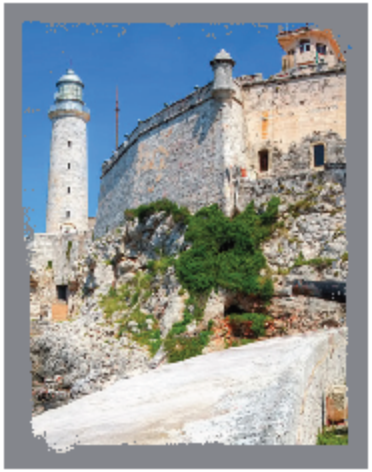
- ***Professor of Computer Science at Florida Atlantic University, Boca Raton, FL., USA***
- ***At IBM for 8 years (L.A. Scientific Center).***
- ***Wrote the first book on database security (Addison-Wesley, 1981).***
- ***Author of many research papers***
- ***Consultant to IBM, Siemens, Lucent,…***
- ***Ing Elect. UTFSM, MS EE Purdue U, PhD CS UCLA***
- ***Now a visiting professor in Chile (UTFSM)***

Markus Schumacher
Eduardo Fernandez-Buglioni
Duane Hybertson
Frank Buschmann
Peter Sommerlad

SECURITY PATTERNS

Integrating Security and Systems Engineering

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS



Eduardo Fernandez-Buglioni

SECURITY PATTERNS IN PRACTICE

Designing Secure Architectures Using Software Patterns

WILEY SERIES IN SOFTWARE DESIGN PATTERNS

Objectives

- ***Get a panorama of security patterns and their use***
- ***Consider a systematic approach to build secure systems based on patterns and UML***
- ***Building Security Reference Architectures for Clouds using patterns***

The value of information

- ***Individuals and enterprises rely on information for credit, health, professional work, business, education,…***
- ***Illegal access (reading or modification) to information can produce serious problems***
- ***Because of its value, information is a growing target of attacks***

Security objectives

- ***Confidentiality--no leakage of sensitive or private information***
- ***Integrity-- no unauthorized modification or destruction of information***
- ***Availability (No denial of service) -- annoying , costly***
- ***Accountability (Non-repudiation)-- legally significant***

Countermeasures

- ***Identification and Authentication– we must know who are you***
- ***Access control/ authorization --provide confidentiality and integrity***
- ***Information hiding (cryptography, steganography)– making information unintelligible***
- ***Auditing-- basis for prosecution or improvements to the system***
- ***Intrusion detection—attack alerting***

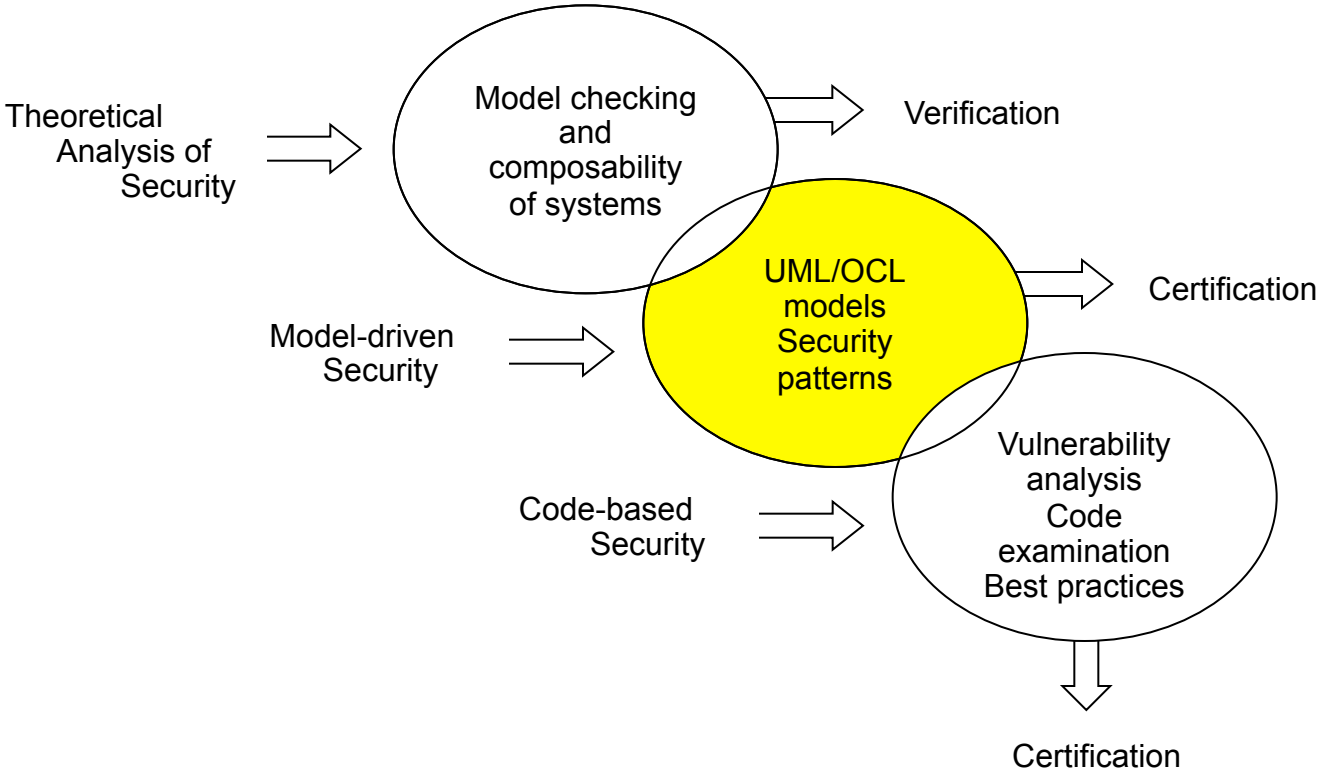
Current situation

- ***The Internet is an insecure place and attacks keep occurring***
- ***One of the main reasons is the poor quality of the software used in systems and application software***
- ***Software engineering neglected security for a long time, emphasis on development speed, no features that can be sold,…***

Remedies

- ***Help designers build secure code using a systematic approach, even if they do not know much about security***
- ***Provide units of security (packed solutions to specific problems) with catalogs and tools***
- ***Build security together with the functional part of the application***
- ***Use a model-based approach***

Approaches to security



Need for a conceptual approach I

- ***Security should be applied where the application semantics is understood***
- ***Security is an all-levels problem***
- ***We should start from high-level policies that can be mapped to the lower levels***
- ***We need precise models to guide system development***
- ***Consider a layered architecture***

Need for conceptual structure II

- ***A unified system is easier to understand: better design, better administration***
- ***Easier to analyze effect of new hardware or software***
- ***Start from policies and models***
- ***Apply security throughout the lifecycle***

Patterns

- *A pattern is a solution to a recurrent problem in a specific context*
- *Idea comes from architecture of buildings (C. Alexander)*
- *Applied initially to software and then extended to other domains*
- *Appeared in 1994 and are slowly being accepted by industry*

Value

- ***Reusable solutions, but maybe not directly, usually require tailoring***
- ***Encapsulate experience and knowledge of designers (best practices)***
- ***Free of errors after a while***
- ***Need to be catalogued to be useful***
- ***Used as guidelines for design***
- ***Good to evaluate systems and standards***
- ***Useful for teaching***

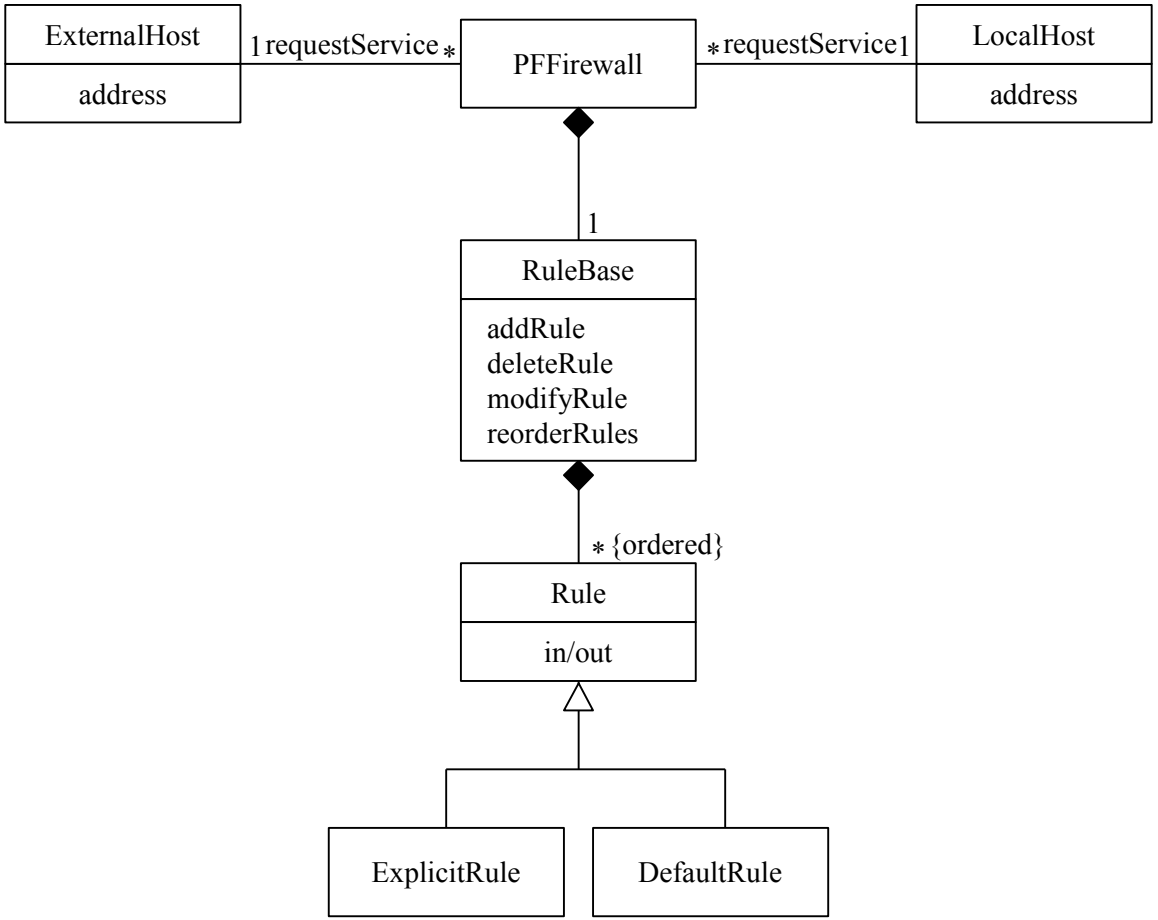
Value of security patterns

- *Can describe security principles (Single Point of Access) or security mechanisms (Firewalls)*
- *Can guide the design and implementation of the security mechanism itself*
- *Can guide the use of security mechanisms in an application (stop specific threats)*
- *Can help understanding and use of complex standards (XACML, WiMax)*
- *Good for teaching security principles and mechanisms*

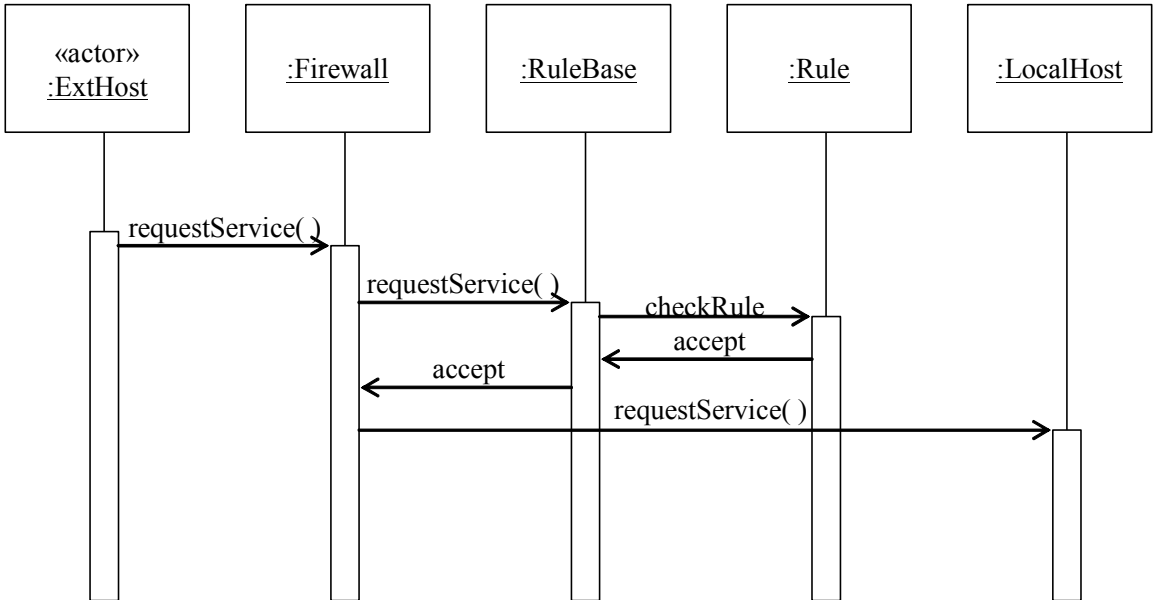
POSA template

- ***Intent (thumbnail)***
- ***Example***
- ***Context***
- ***Problem and forces***
- ***Solution: in words, UML models (static and dynamic)***
- ***Implementation***
- ***Example resolved***
- ***Known uses***
- ***Consequences***
- ***See also (related patterns)***

Structure of the solution



Filtering a client's request



Using the patterns

- ***Catalogs of patterns are not enough, designers must be given guidance in their use***
- ***There are many patterns (growing in number) and the task of selecting them gets harder***
- ***A first approach is to classify the patterns according to some criteria***

We can use patterns at all levels

- ***Patterns for models define the highest level***
- ***At each lower level we refine the patterns at the previous level to consider the specific aspects of each level***
- ***We'll analyze some patterns from each layer***

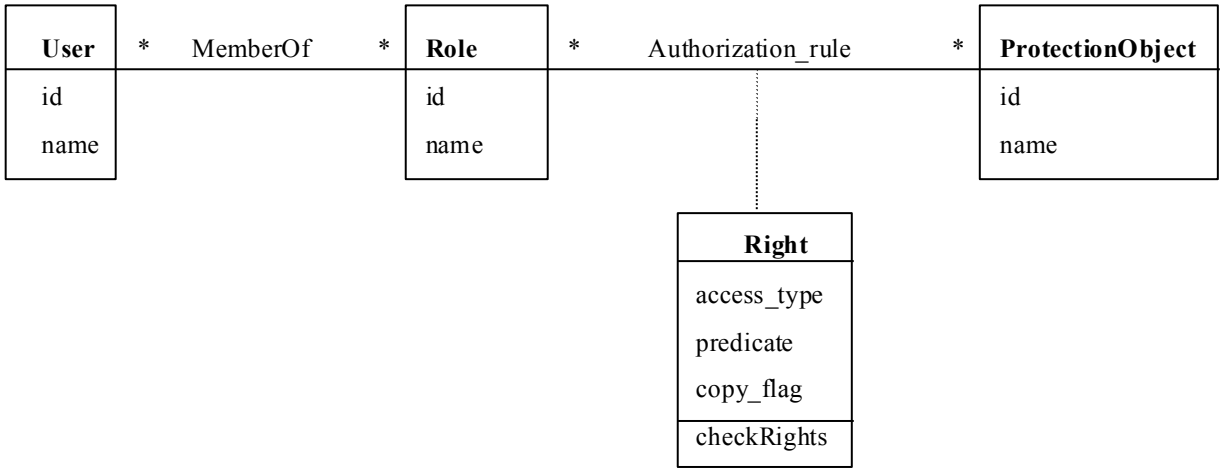
Applic. Layer: Access control models

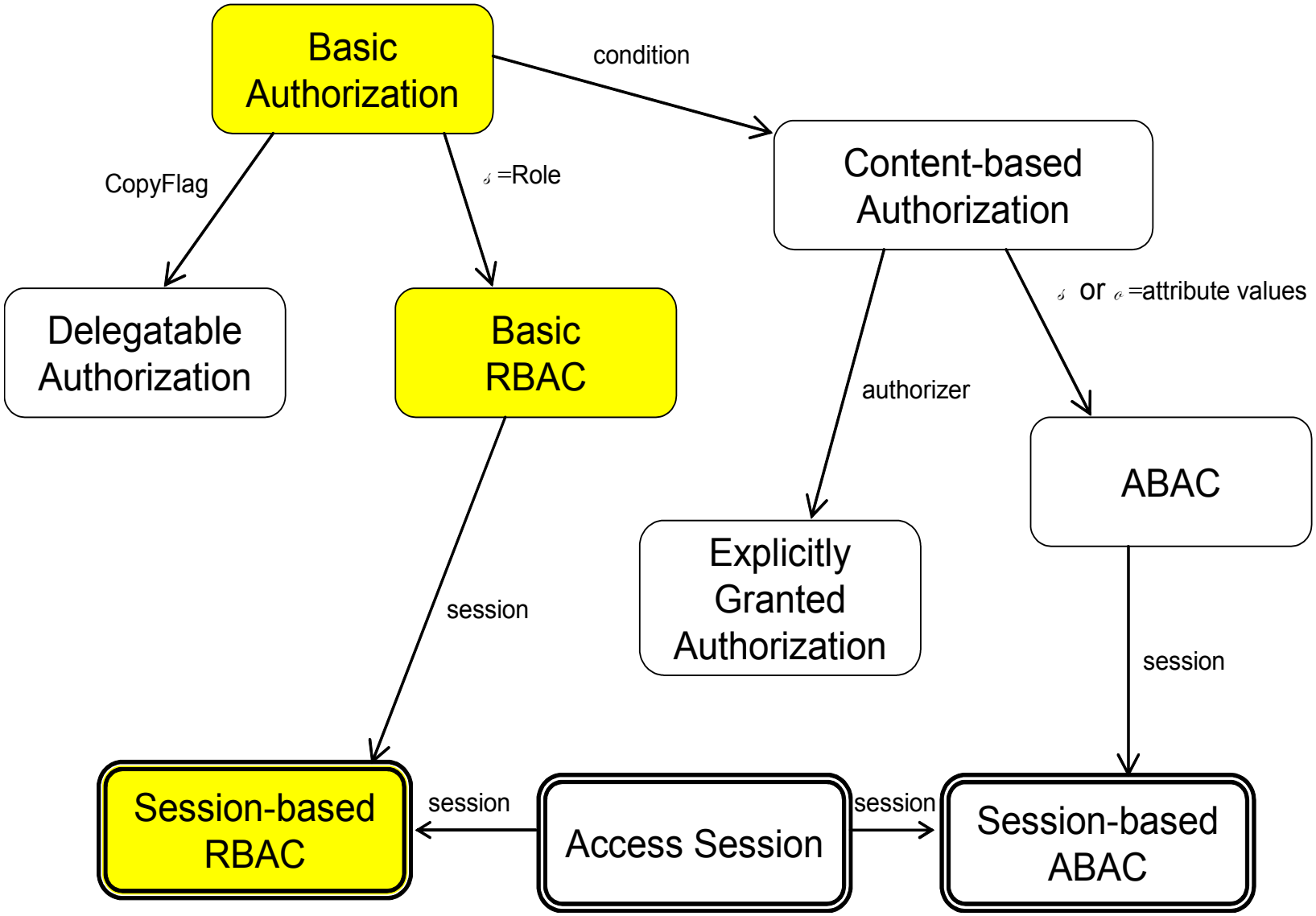
- ***Authorization.*** ***How do we describe who is authorized to access specific resources in a system? A list of authorization rules describes who has access to what and how.***
- ***Role-Based Access Control (RBAC).*** ***How do we assign rights to people based on their functions or tasks? Assign people to roles and give rights to these roles so they can perform their tasks.***
- ***Multilevel Security*****. How to decide access in an environment with security classifications.**

Role-Based Access Control

- ***Users are assigned roles according to their functions and given the needed rights (access types for specific objects)***
- ***When users are assigned by administrators, this is a mandatory model***
- ***Can implement least privilege and separation of duty policies***

Basic RBAC pattern

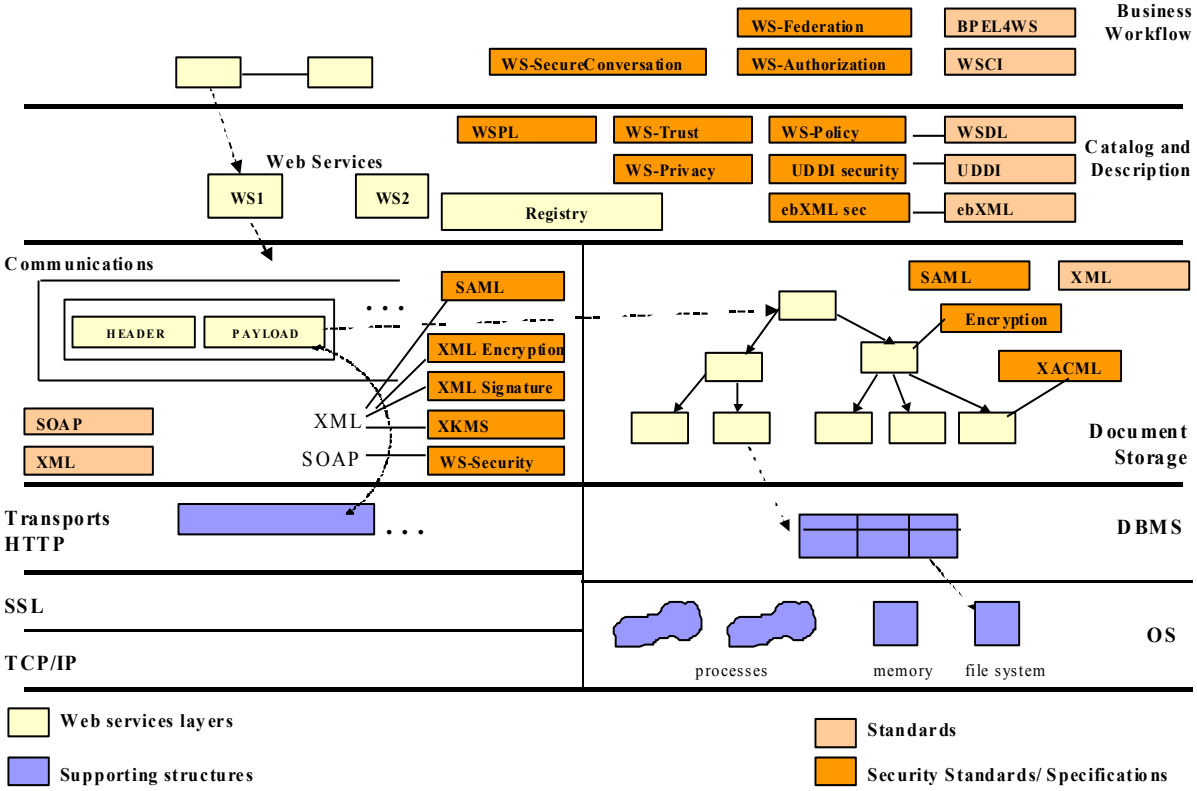



Basic Authorization
condition
CopyFlag
=Role
Content-based Authorization
Delegatable Authorization
Basic RBAC
or =attribute values
authorizer
ABAC
Explicitly Granted Authorization
session
session
Session-based RBAC
session
Access Session
session
Session-based ABAC

Web services security

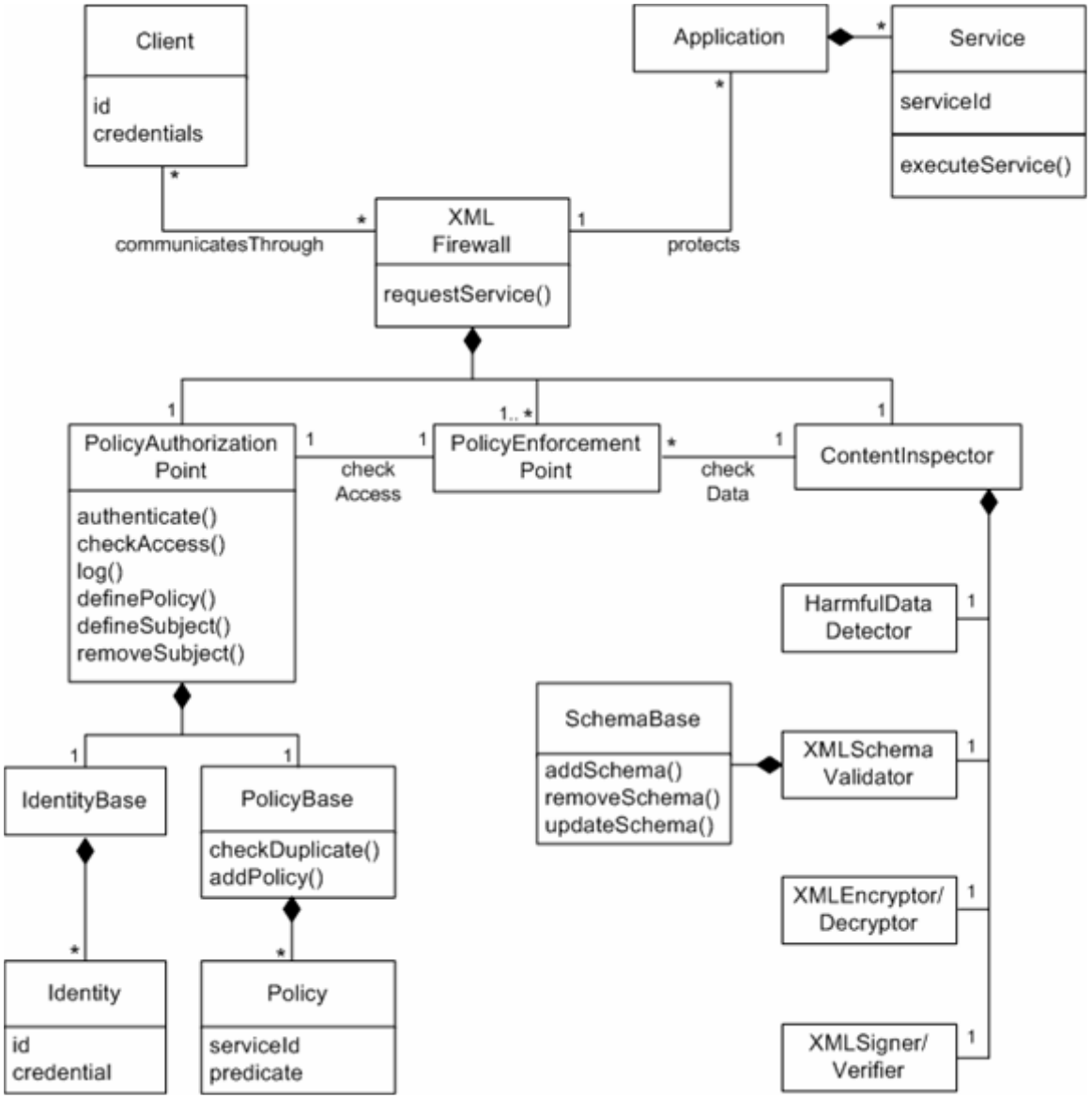
- *Application Firewall [Del04]. The application firewall filters calls and responses to/from enterprise applications, based on an institution access control policies.*
- *XML Firewall [Del04]. Filter XML messages to/from enterprise applications, based on business access control policies and the content of the message.*
- *XACML Authorization [Del05]. Enable an organization to represent authorization rules in a standard manner.*
- *XACML Access Control Evaluation [Del05]. This pattern decides if a request is authorized to access a resource according to policies defined by the XACML Authorization pattern. .*
- *WSPL [Del05]. Enable an organization to represent access control policies for its web services in a standard manner. It also enables a web services consumer to express its requirements in a standard manner.*

Standards for web services security



XML firewall

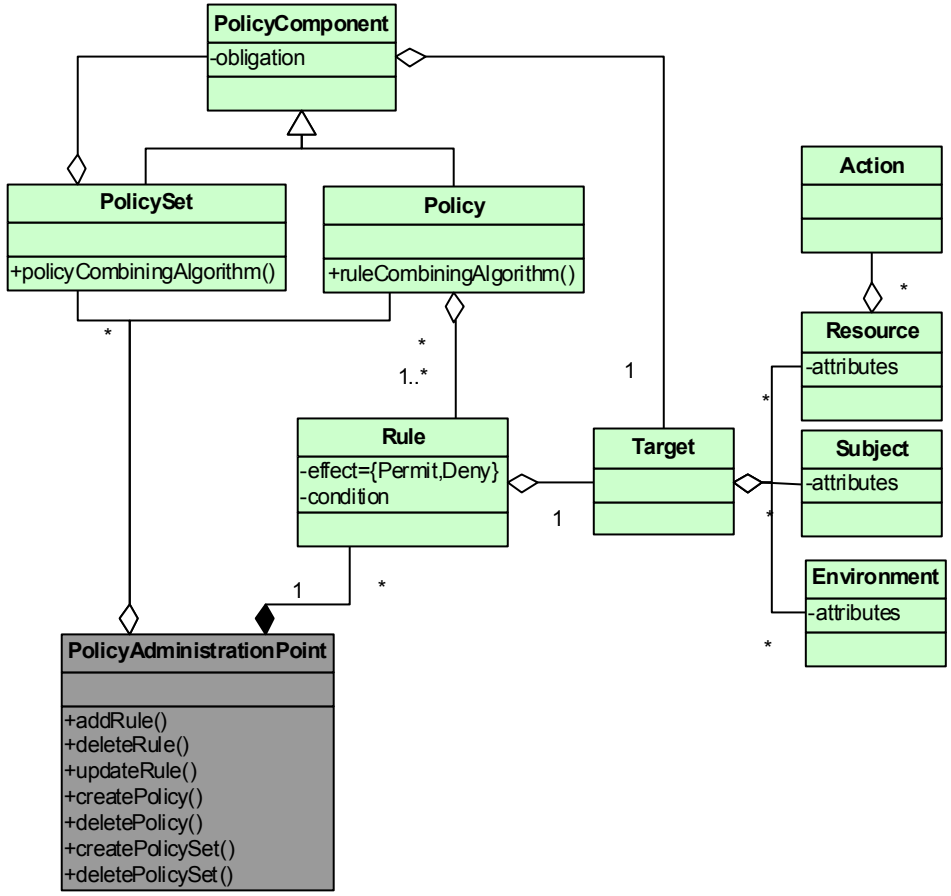
- Controls input/output of XML applications
- Well-formed documents (schema as reference)
- Harmful data (wrong type or length)
- Encryption/decryption
- Signed documents


Client
id
credentials
Application
Service
serviceId
executeService()
communicatesThrough
XML Firewall
requestService()
protects
PolicyAuthorization Point
authenticate()
checkAccess()
log()
definePolicy()
defineSubject()
removeSubject()
check Access
PolicyEnforcement Point
check Data
ContentInspector
HarmfulData Detector
SchemaBase
addSchema()
removeSchema()
updateSchema()
XMLSchema Validator
XMLEncryptor/ Decryptor
XMLSigner/ Verifier
IdentityBase
PolicyBase
checkDuplicate()
addPolicy()
Identity
id
credential
Policy
serviceId
predicate

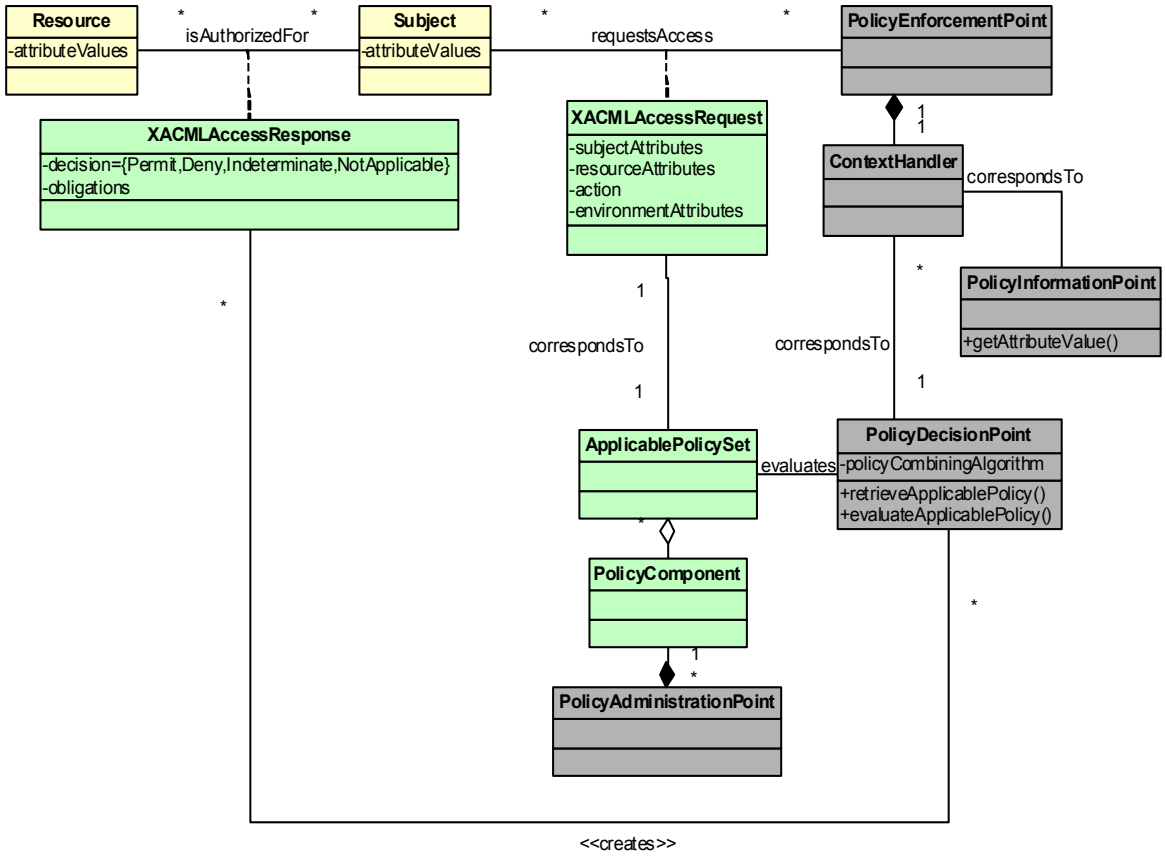
XACML

- ***Special technical committee of OASIS***
- ***Specification of policies for information access over the Internet and their enforcement***
- ***Combines work of IBM Tokyo and University of Milano, Italy.***
- ***Implemented by Sun in early 2003***

XACML Authorization

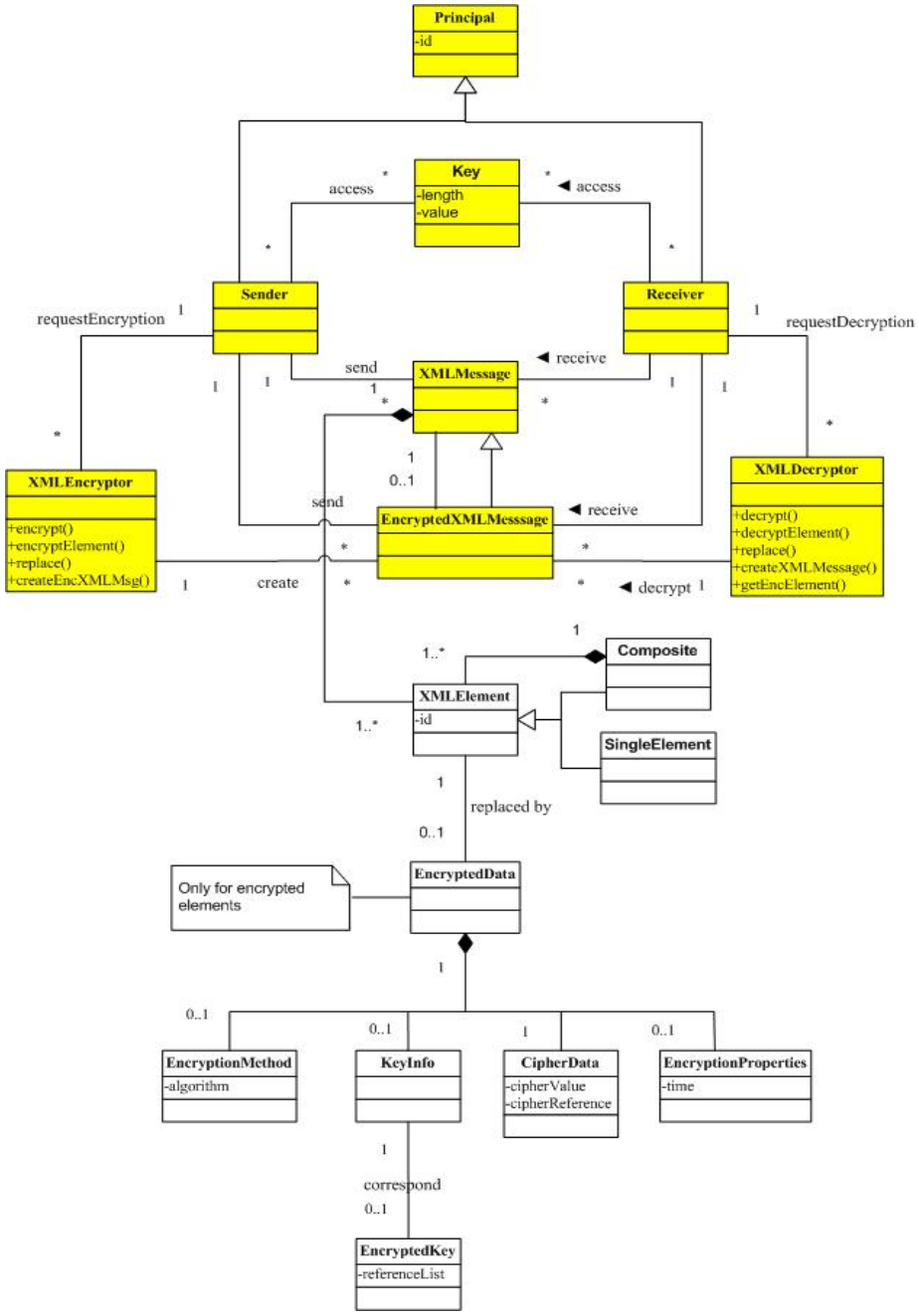


Access control evaluation

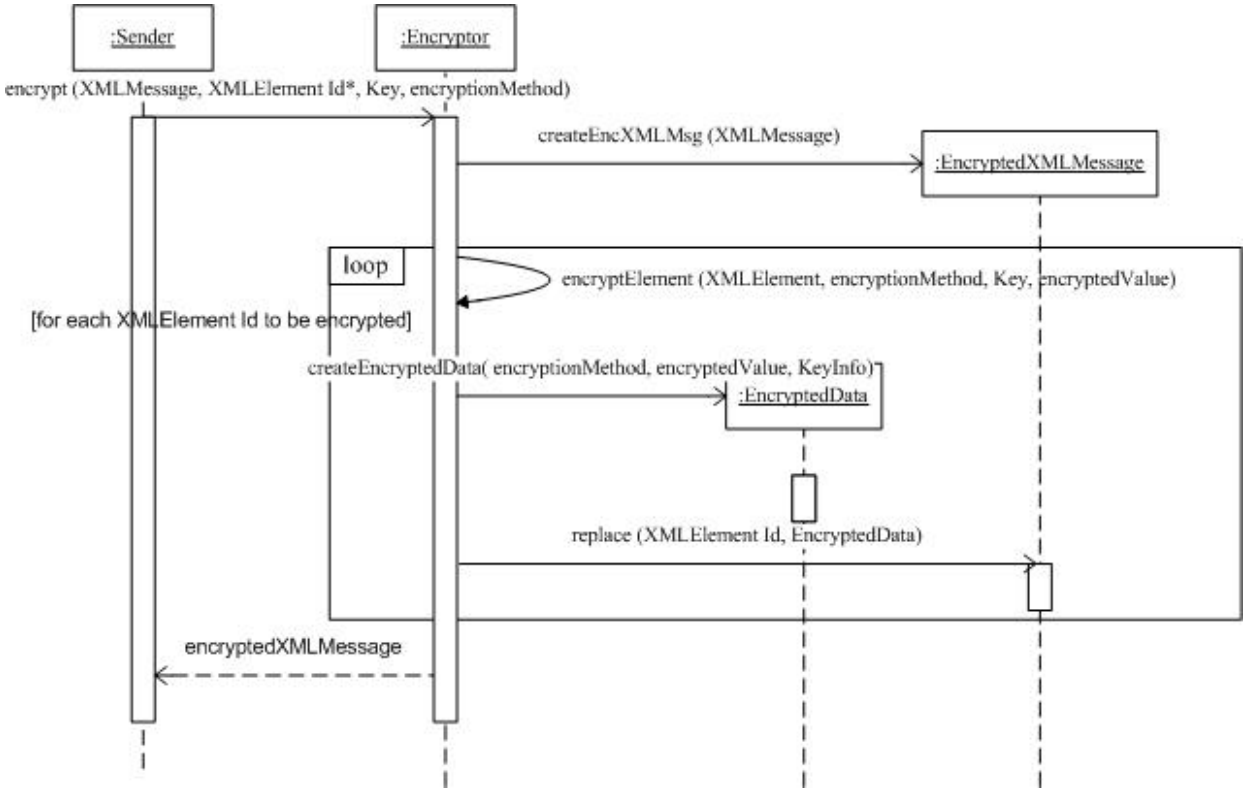


XML Encryption

- ***The XML Encryption standard describes the syntax to represent XML encrypted data and the process of encryption and decryption. XML Encryption provides confidentiality by hiding selected sensitive information in a message using cryptography.***


Principal
-id
Key
-length
-value
access
access
Sender
Receiver
requestEncryption
requestDecryption
send
XMLMessage
receive
XMLEncryptor
+encrypt()
+encryptElement()
+replace()
+createEncXMLMsg()
send
EncryptedXMLMesssage
receive
XMLDecryptor
+decrypt()
+decryptElement()
+replace()
+createXMLMessage()
+getEncElement()
create
decrypt
Composite
XMLElement
-id
SingleElement
replaced by
Only for encrypted elements
EncryptedData
EncryptionMethod
-algorithm
KeyInfo
CipherData
-cipherValue
-cipherReference
EncryptionProperties
-time
correspond
EncryptedKey
-referenceList

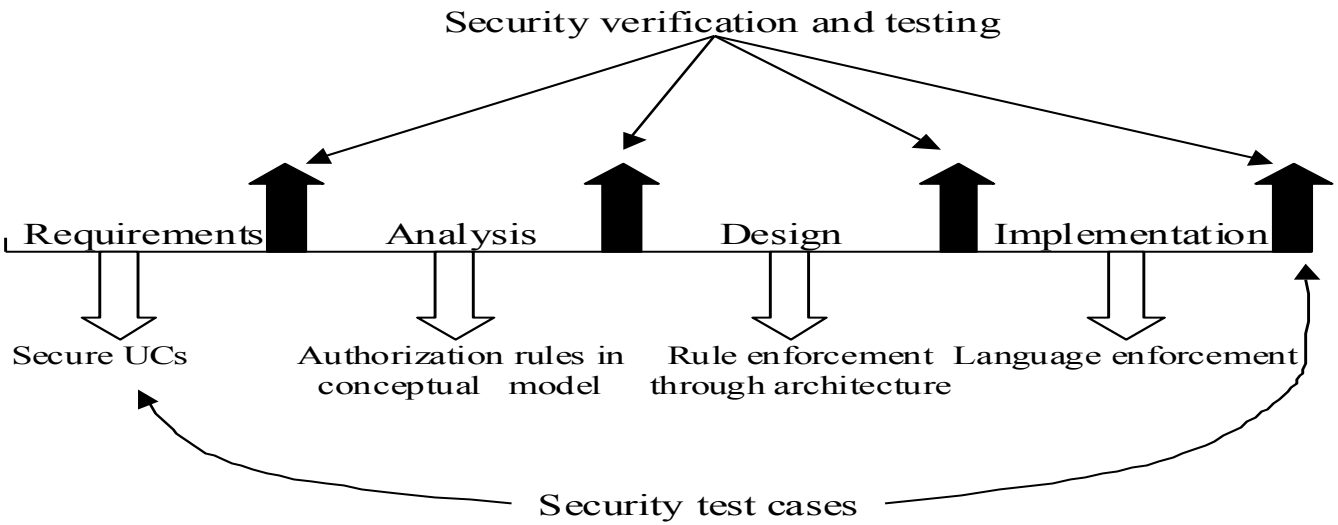
Encrypting elements



How to apply the patterns?

- ***A good catalog and classifications of patterns help a designer select among alternatives.***
- ***However, there is still the problem of when to apply a pattern during system development***
- ***We need some systematic approach to decide when we need to use a pattern, a secure systems methodology***

Security along the life cycle



A methodology for secure systems design I

- ***Domain analysis stage**: A business model is defined. Legacy systems are identified and their security implications analyzed. Domain and regulatory constraints are identified. Policies must be defined up front, in this phase.*
- ***Requirements stage**: Use cases define the required interactions with the system. Applying the principle that security must start from the highest levels, it makes sense to relate attacks to use cases. We study each action within a use case and see which threats are possible. We then determine which policies would stop these attacks. From the use cases we can also determine the needed rights for each actor and thus apply a need-to-know policy.*

Requirements stage

- *Use cases are determined*
- *Activity diagrams for use cases or sequences of use cases*
- *Define level of security needed*
- *Identify attacks*
- *Select attacks based on risk analysis*

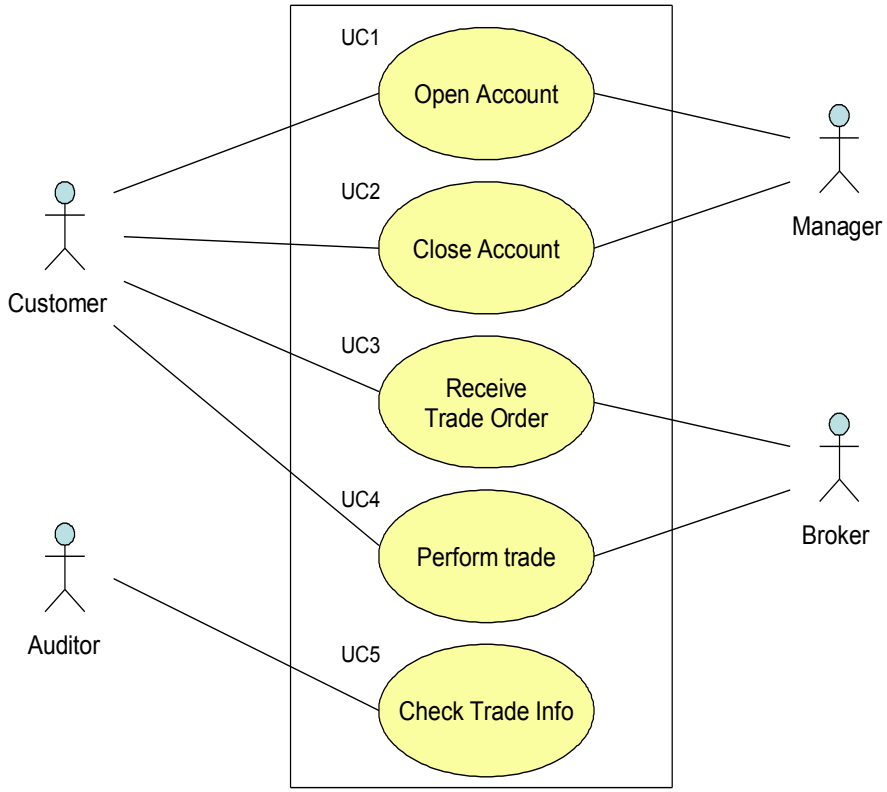
Identifying attacks

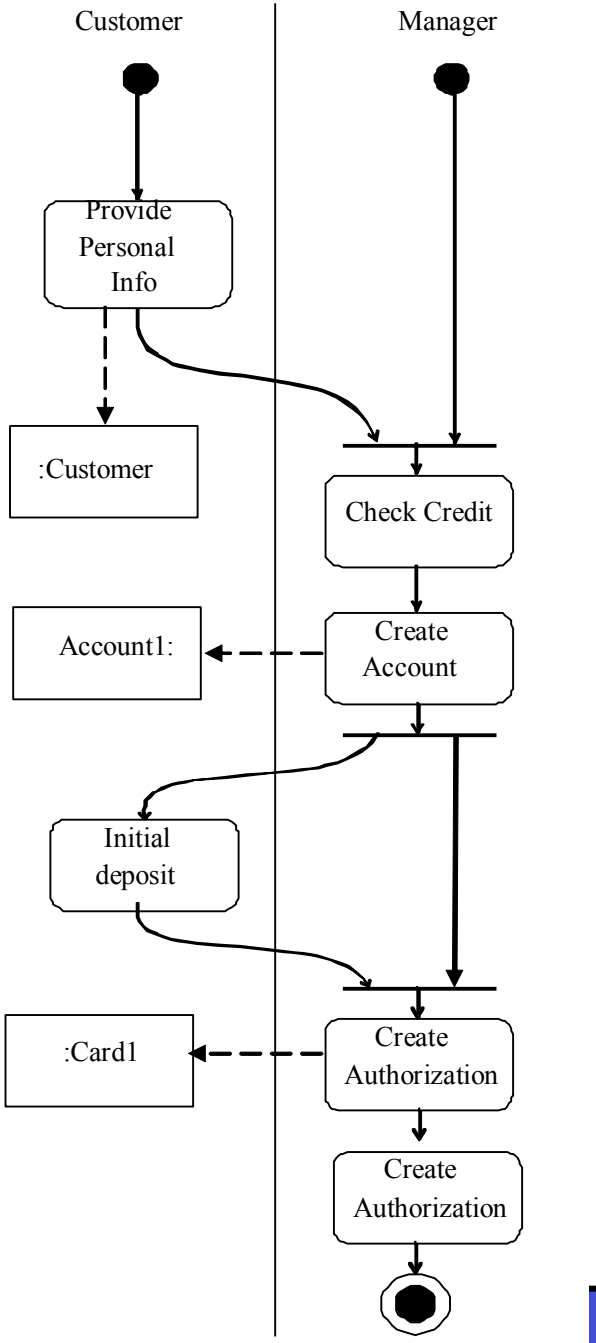
- ***We need to know what kind of attacks to expect.***
- ***We relate attacks to attacker goals***
- ***We study systematically all the possible attacks to each activity in a use case***
- ***Use cases define all functional interactions***

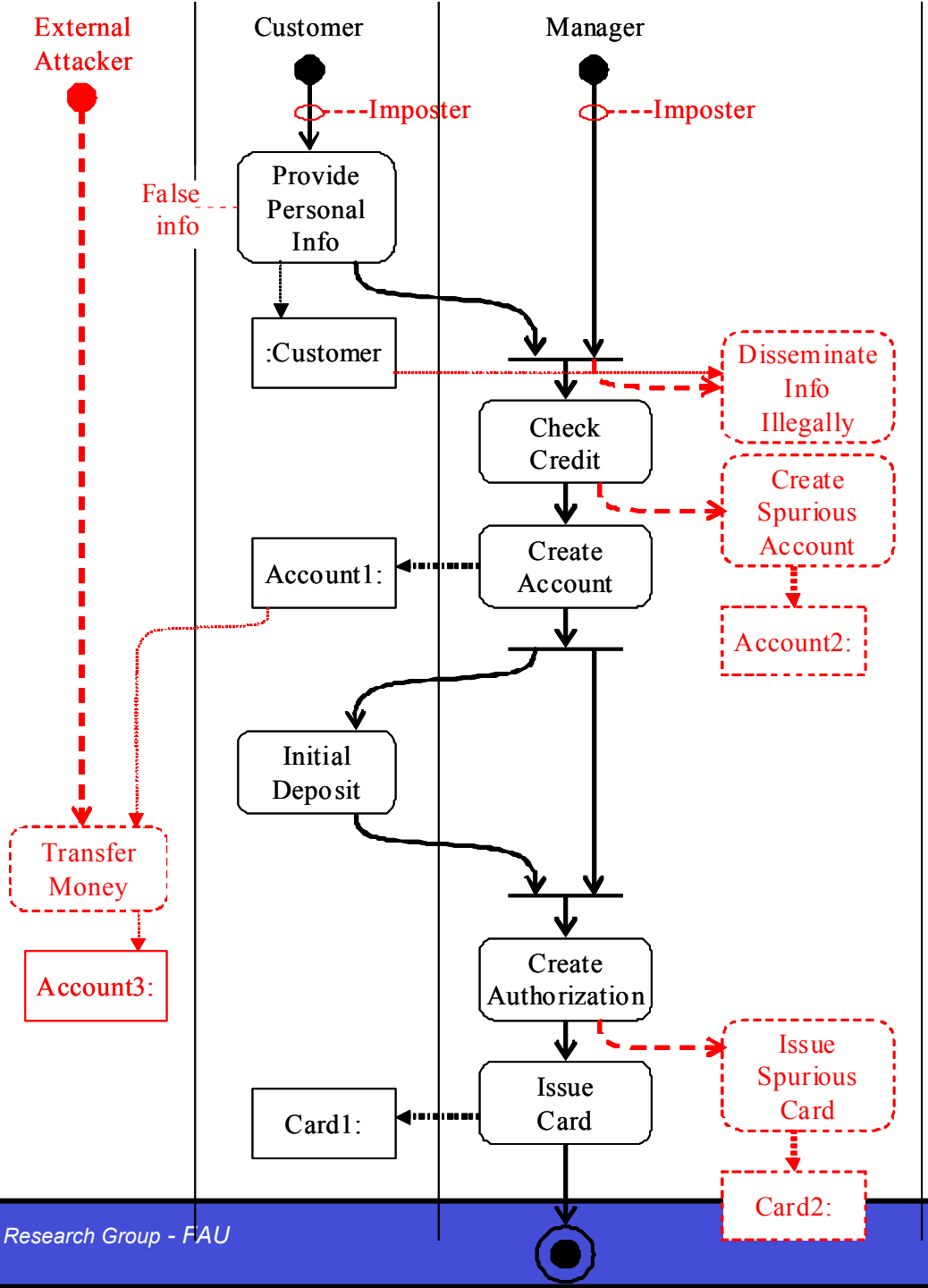
Attacker goals

- ***Attacker is not interested in changing a few bits or destroying a message***
- ***Attacker wants to accomplish some objective, e.g., steal money, steal identity***
- ***This is applying the principle of defining security at the semantic levels***
- ***We also need to comply with standards***

A financial institution




Customer
Manager
Provide Personal Info
:Customer
Check Credit
Account1:
Create Account
Initial deposit
:Card1
Create Authorization
Create Authorization


External Attacker
Customer
Manager
Imposter
Imposter
False info
Provide Personal Info
:Customer
Disseminate Info Illegally
Check Credit
Create Spurious Account
Account1:
Create Account
Account2:
Initial Deposit
Transfer Money
Account3:
Create Authorization
Issue Spurious Card
Card1:
Issue Card
Card2:

Threats

- ***T1.The customer is an impostor and opens an account in the name of another person***
- ***T2.The customer provides false information and opens an spurious account***
- ***T3.The manager is an impostor and collects data illegally***
- ***T4.The manager collects customer information to use illegally***
- ***T5.The manager creates a spurious account with the customer's information***
- ***T6.The manager creates a spurious authorization card to access the account***
- ***T7.An attacker tries to prevent the customers to access their accounts***
- ***T8.An attacker tries to move money from an account to her own account***

Analysis stage

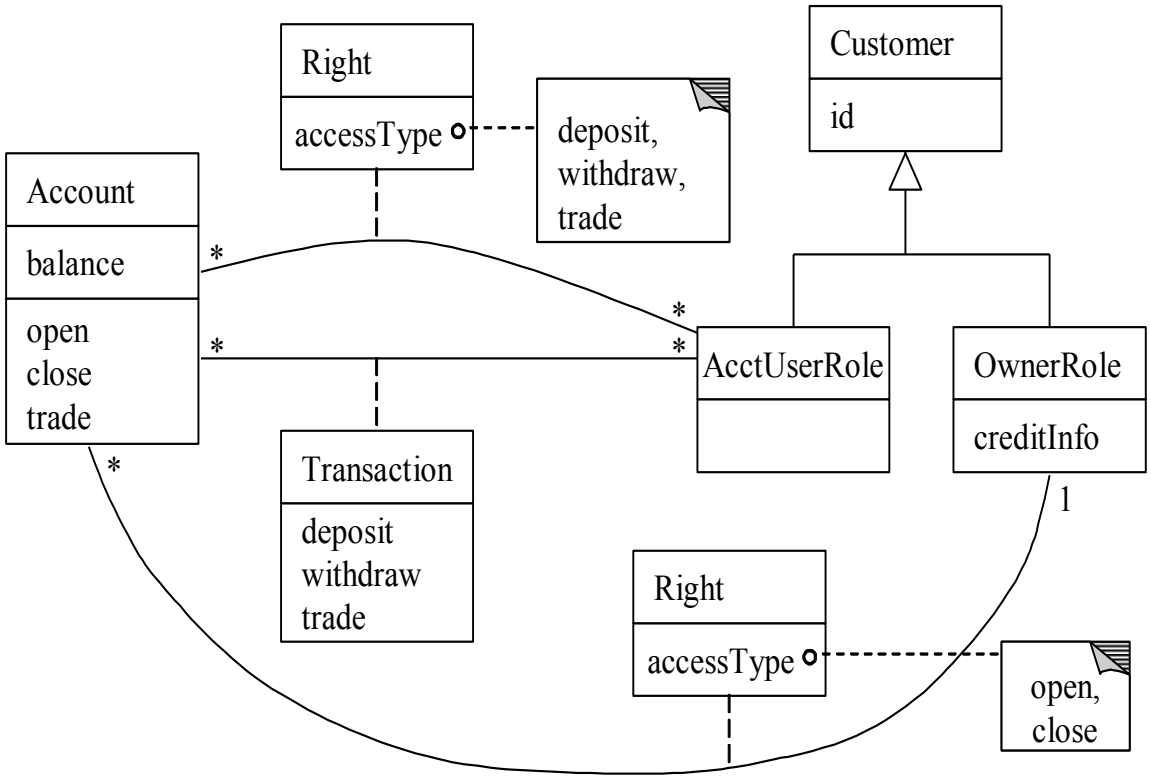
- ***Analysis patterns can be used to build the conceptual model. Security patterns describe security models or mechanisms. We can build a conceptual model where repeated applications of a security model pattern realize the rights determined from use cases.***

Use case analysis leads to policies

- ***T1. T3. Mutual authentication. Every interaction across system nodes is authenticated.***
- ***T2. Verify source of information.***
- ***T4. Logging. Since the manager is using his legitimate rights we can only log his actions for auditing at a later time.***
- ***T5. T6. Separation of administration from use of data. For example, a manager can create accounts but should have no rights to withdraw or deposit in the account.***
- ***T7. Protection against denial of service. We need some redundancy in the system to increase its availability.***
- ***T8. Authorization. If the user is not explicitly authorized he should not be able to move money from any account.***

Policies can be realized with patterns

Rights for financial application



Methodology summary

- ***Use case activities define attacks***
- ***Attacks lead to policies to stop them***
- ***Use cases define needed actor rights***
- ***Access matrix or RBAC models formalize these rights***

Security methodology III

- ***Implementation stage:*** ***This stage requires reflecting in the code the security rules defined in the design stage. Because these rules are expressed as classes, associations, and constraints, they can be implemented as classes in object-oriented languages. In this stage we can also select specific security packages or COTS, e.g., a firewall product, a cryptographic package. Some of the patterns identified earlier in the cycle can be replaced by COTS (these can be tested to see if they include a similar pattern).***

Misuse/Attack patterns

- ***It is not clear to an inexperienced designer what security pattern should be applied to stop a specific attack***
- ***Security patterns are not useful either for forensics because they do not emphasize the modus operandi of attacks.***
- ***Attack patterns describe, from the point of view of the attacker, how a type of attack is performed (what system units it uses and how), proposes ways of stopping the attack by enumerating possible security patterns that can be applied for this purpose, and helps analyzing the attack once it has happened by indicating where can we find forensic data as well as what type of data.***

Current work

- ***Cloud Computing definition***
  - According to NIST, Cloud Computing is “a model for enabling convenient, on-demand network access to a shared of pool configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”.

Latest results

- ***Performed a systematic survey of security issues for cloud environments where we enumerated the main cloud threats, vulnerabilities , and possible defenses found in the literature***
- ***Developed a reference architecture to have a precise view of cloud systems to be used as a framework for security***
- ***Described three specific cloud threats in the form of misuse patterns:***
  - Resource Usage Monitoring Inference
  - Malicious Virtual Machine Creation
  - Malicious Virtual Machine Migration.
- ***Showed how to secure a reference architecture by applying security patterns to add security defenses and misuse patterns to evaluate its security level.***
  - Developed a pattern for a secure virtual machine repository system

Analysis of Security Issues

- ***A categorization of security issues for Cloud Computing focused in its service models SaaS, PaaS and IaaS.***
- ***We identify the main vulnerabilities and the most important threats.***
- ***We also present some countermeasures related to these threats.***

Vulnerabilities in Cloud Computing

ID	Vulnerabilities	Description	Layer
V01	Insecure interfaces and APIs	Cloud providers offer services that can be accessed through APIs (SOAP, REST, or HTTP with XML/JSON) [65]. The security of the cloud depends upon the security of these interfaces [40]. Some problems are: a) Weak credential b) Insufficient authorization checks c) Insufficient input-data validation Also, cloud APIs are still immature which means that are frequently updated. A fixed bug can introduce another security hole in the application [76].	SPI
V02	Unlimited allocation of resources	Inaccurate modeling of resource usage can lead to overbooking or over-provisioning [41].	SPI
V03	Data-related vulnerabilities	a) Data can be colocated with the data of unknown owners (competitors, or intruders) with a weak separation [59] b) Data may be located in different jurisdictions which have different laws [43][76][77] c) Incomplete data deletion – data cannot be completely removed [43][44][49][78] d) Data backup done by untrusted third-party providers [78][79] e) Information about the location of the data usually is unavailable or not disclosed to users [49] f) Data is often stored, processed, and transferred in clear plain text	SPI

Vulnerabilities in Cloud Computing

ID	Vulnerabilities	Description	Layer
V04	Vulnerabilities in Virtual Machines	a) Possible covert channels in the colocation of VMs [70][80][81] b) Unrestricted allocation and deallocation of resources with VMs [79] c) Uncontrolled Migration - VMs can be migrated from one server to another server due to fault tolerance, load balance, or hardware maintenance [65][67] d) Uncontrolled snapshots – VMs can be copied in order to provide flexibility [53], which may lead to data leakage e) Uncontrolled rollback could lead to reset vulnerabilities - VMs can be backed up to a previous state for restoration [67], but patches applied after the previous state disappear f) VMs have IP addresses that are visible to anyone within the cloud - attackers can map where the target VM is located within the cloud (Cloud cartography [80])	I
V05	Vulnerabilities in Virtual Machine Images	a) Uncontrolled placement of VM images in public repositories [48] b) VM images are not able to be patched since they are dormant artifacts [67]	I
V06	Vulnerabilities in Hypervisors	a) Complex hypervisor code [82] b) Flexible configuration of VMs or hypervisors to meet organization needs can be exploited	I
V07	Vulnerabilities in Virtual Networks	Sharing of virtual bridges by several virtual machines [73]	I

Threats in Cloud Computing

ID	Threats	Description	Layer
T01	Account or service hijacking	An account theft can be performed by different ways such as social engineering and weak credentials. If an attacker gains access to a user's credential, he can perform malicious activities such as access sensitive data, manipulate data, and redirect any transaction [40].	SPI
T02	Data scavenging	Since data cannot be completely removed from unless the device is destroyed, attackers may be able to recover this data [41][49][39].	SPI
T03	Data leakage	Data leakage happens when the data gets into the wrong hands while it is being transferred, stored, audited or processed [40][41][44][80].	SPI
T04	Denial of Service	It is possible that a malicious user will take all the possible resources. Thus, the system cannot satisfy any request from other legitimate users due to resources being unavailable.	SPI
T05	Customer-data manipulation	Users attack web applications by manipulating data sent from their application component to the server's application [44][55]. For example, SQL injection, command injection, insecure direct object references, and cross-site scripting.	S
T06	VM escape	It is designed to exploit the hypervisor in order to take control of the underlying infrastructure [48][83].	I
T07	VM hopping	It happens when a VM is able to gain access to another VM (i.e by exploting some hypervisor vulnerability) [41][66]	I
T08	Malicious VM creation	An attacker who creates a valid account can create a VM image containing malicious code such as a Trojan horse and store it in the provider repository [44].	I
T09	Insecure VM migration	Live migration of virtual machines exposes the contents of the VM state files to the network. An attacker can do the following actions: a) Access data illegally during migration [65] b) Transfer a VM to an untrusted host [67] c) Create and migrate several VM causing disruptions or DoS	I
T10	Sniffing/Spoofing virtual networks	A malicious VM can listen to the virtual network or even use ARP spoofing to redirect packets from/to other VMs [68][73].	I

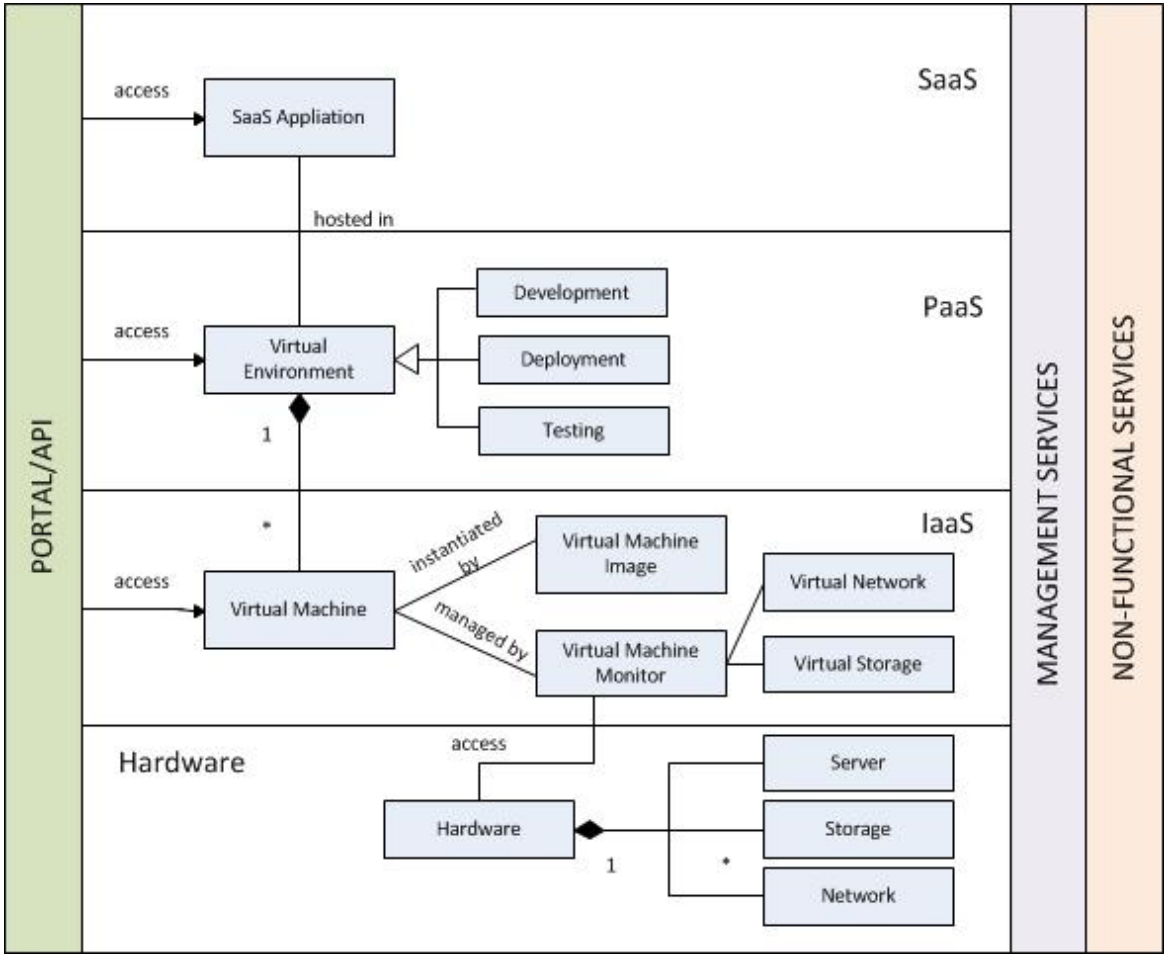
Relationships between Threats, Vulnerabilities, and Countermeasures

Threat	Vulnerabilities	Incidents	Countermeasures
T01	V01	An attacker can use the victim's account to get access to the target's resources.	Identity and Access Management Guidance [86] Dynamic credential [87]
T02	V03a, V03c	Data from hard drives that are shared by several customers cannot be completely removed.	Specify destruction strategies on Service-level Agreements (SLAs)
T03	V03a, V03c, V03d, V03f, V04a-f, V05a, V07	Authors in [80] illustrated the steps necessary to gain confidential information from other VMs co-located in the same server as the attacker. Side channel [88]	FRS techniques [89] Digital Signatures [90] Encryption [88] Homomorphic encryption [91]
T04	V01, V02	An attacker can request more computational resources, so other legal users are not able to get additional capacity.	Cloud providers can force policies to offer limited computational resources
T05	V01	Some examples are described in [55] such as SQL, command injection, and cross-site scripting	Web application scanners [92]
T06	V06a, V06b	A zero-day exploit in the HyperVM virtualization application that destroyed about 100,000 websites [93]	HyperSafe [82] TCCP (Trusted Cloud Computing Platform) [84] TVDc (Trusted Virtual Datacenter) [94][95]
T07	V04b, V06b	[96] presents a study that demonstrates security flaws in most virtual machines monitors	
T08	V05a, V05b	An attacker can create a VM image containing malware and publish it in a public repository.	Mirage [71]
T09	V04d	[97] has empirically showed attacks against the migration functionality of the latest version of the Xen and VMware virtualization products.	PALM [85] TCCP [84] VNSS [74]
T10	V07	Sniffing and spoofing virtual networks [73]	Virtual network framework based on Xen network modes: "bridged" and "routed" [73]

Reference Architecture

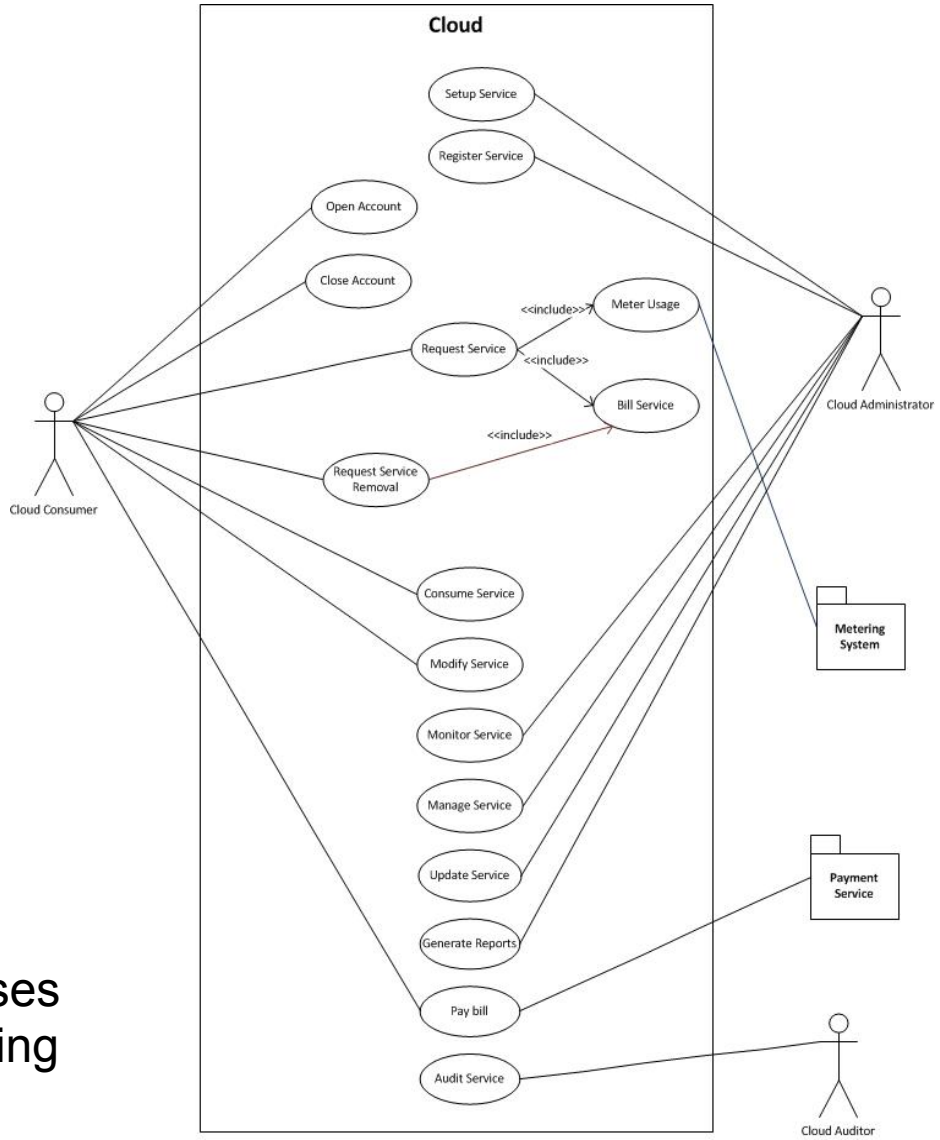
- ***Our proposed reference architecture includes:***
  - Use case models
    - Common use cases
    - Use cases for IaaS
    - Use cases for PaaS
    - Use cases for SaaS
  - Patterns for IaaS, PaaS, and SaaS

Cloud Architecture Overview



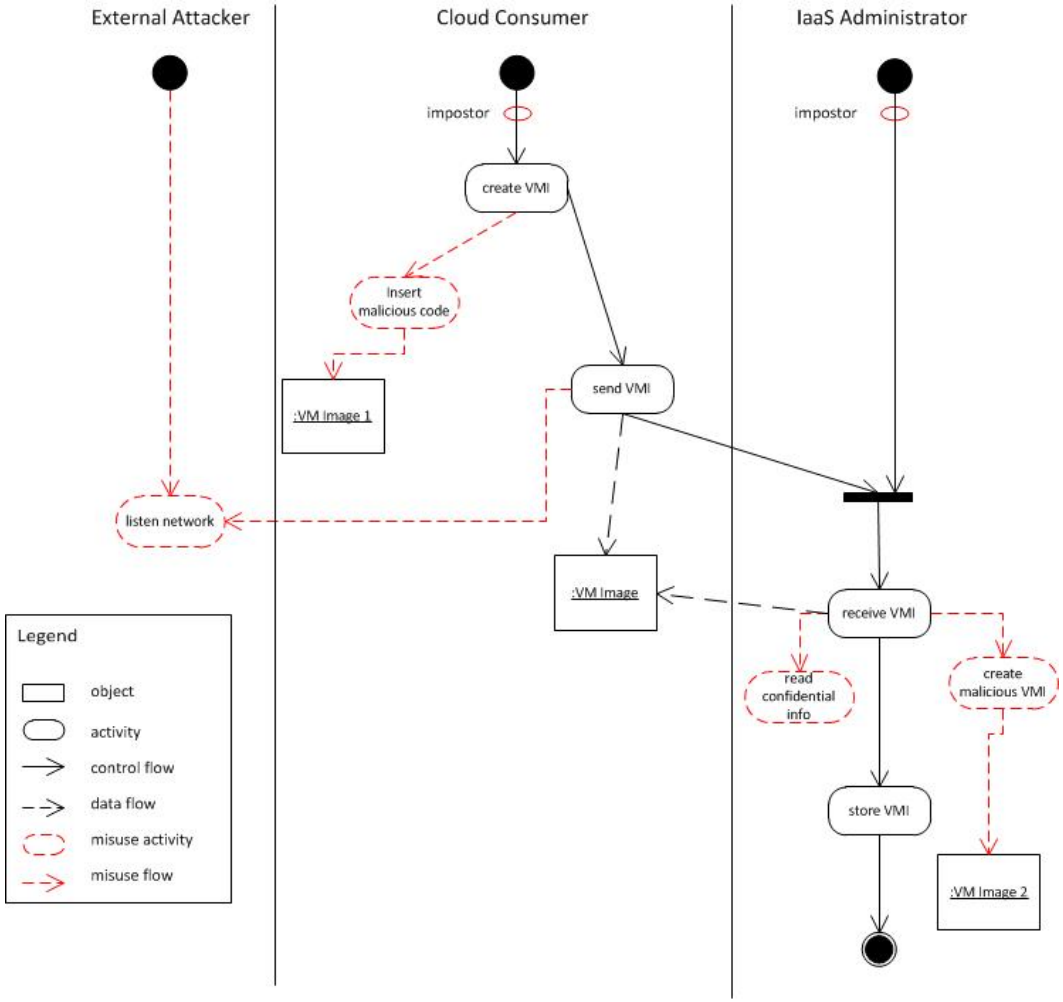
Cloud Architecture Overview

Use Cases



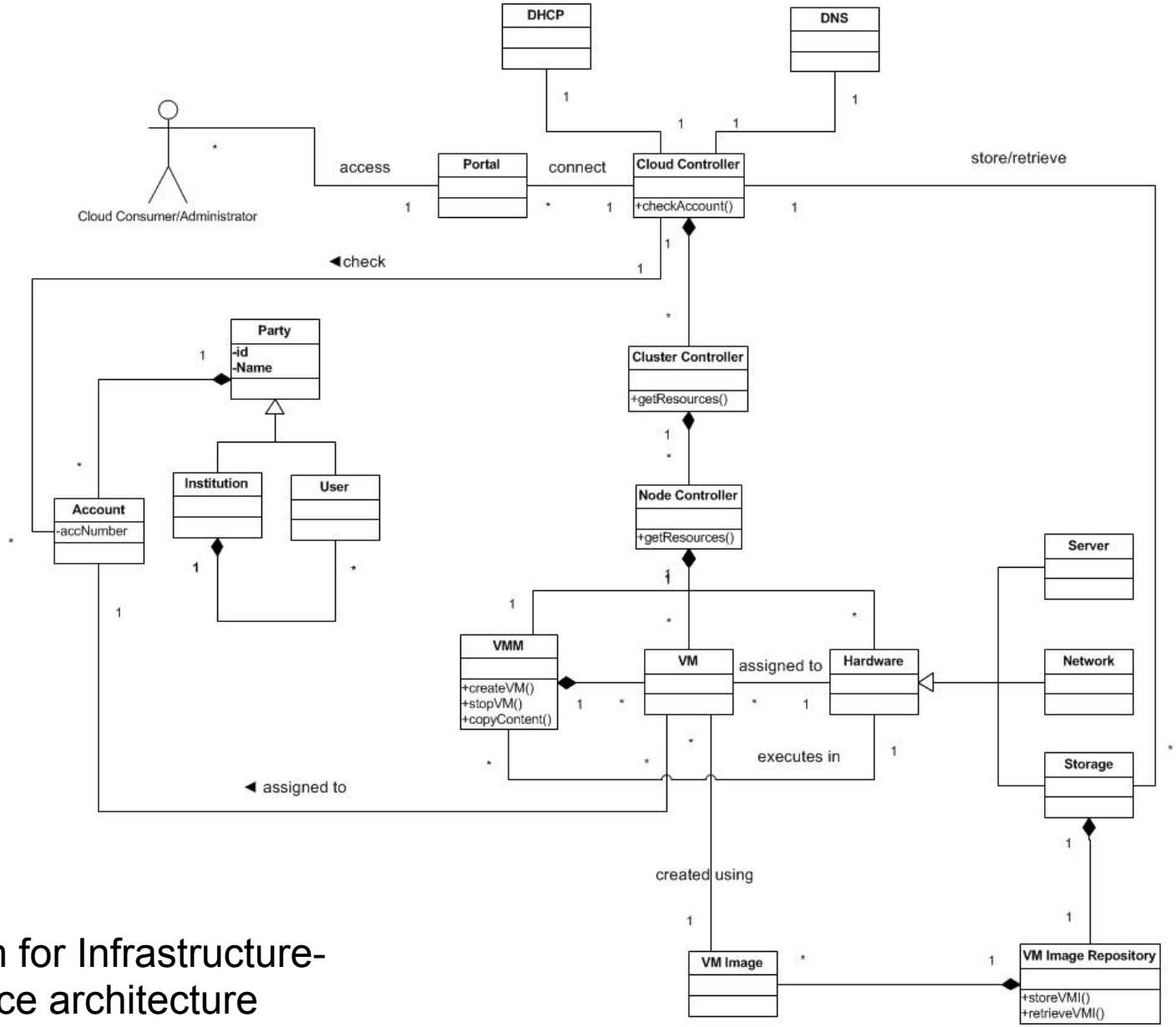
Common Use Cases
for Cloud Computing

UCs: Create VMI and Publish VMI



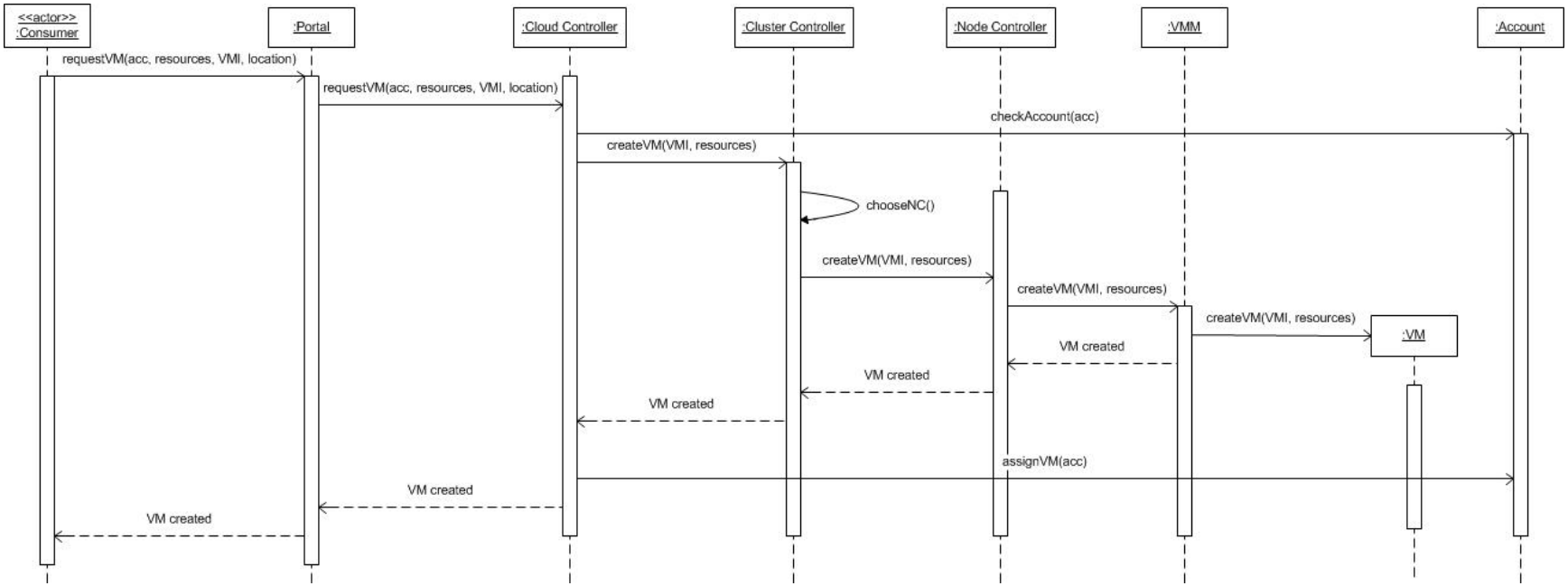
		Misuse Activity				
Actor	Action	#	Sec. Att. CO/IN/AV/AC	Source AIn/UIn/Out	Description	Asset
Cloud Consumer	Create VMI	T1	IN	Out	Insert malicious code in the image	VMI
Cloud Consumer	Send VMI	T2	CO	Out	VMI may be read while being transmitted	VMI
		T3	IN	Out	VMI may be modified while in transit	VMI
		T4	AC	Out	Disavows sending a VMI	VMI
IaaS Administrator	Receive VMI	T5	CO	AIn/UIn	Collects sensitive information from VMI	VMI
		T6	AV	AIn	Disavows receiving a VMI	VMI
		T7	IN	UIn/AIn	Insert malicious code in the image	VMI

ID	Threats	Defense
T1	The cloud customer is an impostor and publishes a VMI	Authenticator - Authorization
T2	The cloud consumer inserts malicious code within a VMI	Filter module
T3	An external attacker listens to the network to obtain information about the VMI	Secure network
T4	The IaaS administrator is an impostor and collects information within the VMI	Authenticator - Authorization
T5	The IaaS administrator creates a malicious VMI	Filter module
T6		
T7		

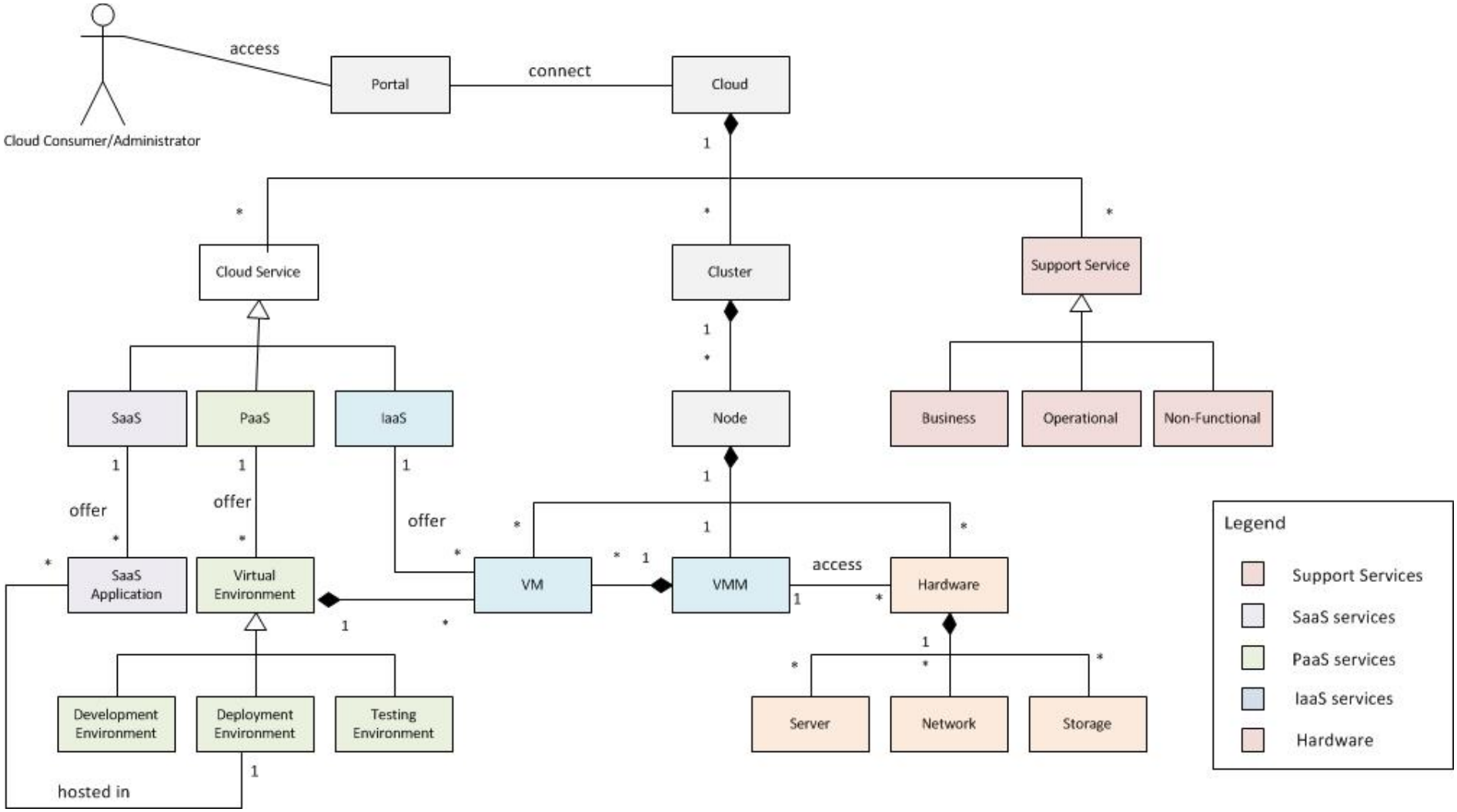


Class Diagram for Infrastructure-as-a-Service architecture

Infrastructure-as-a-Service Pattern



Sequence Diagram for Use Case Create a Virtual Machine

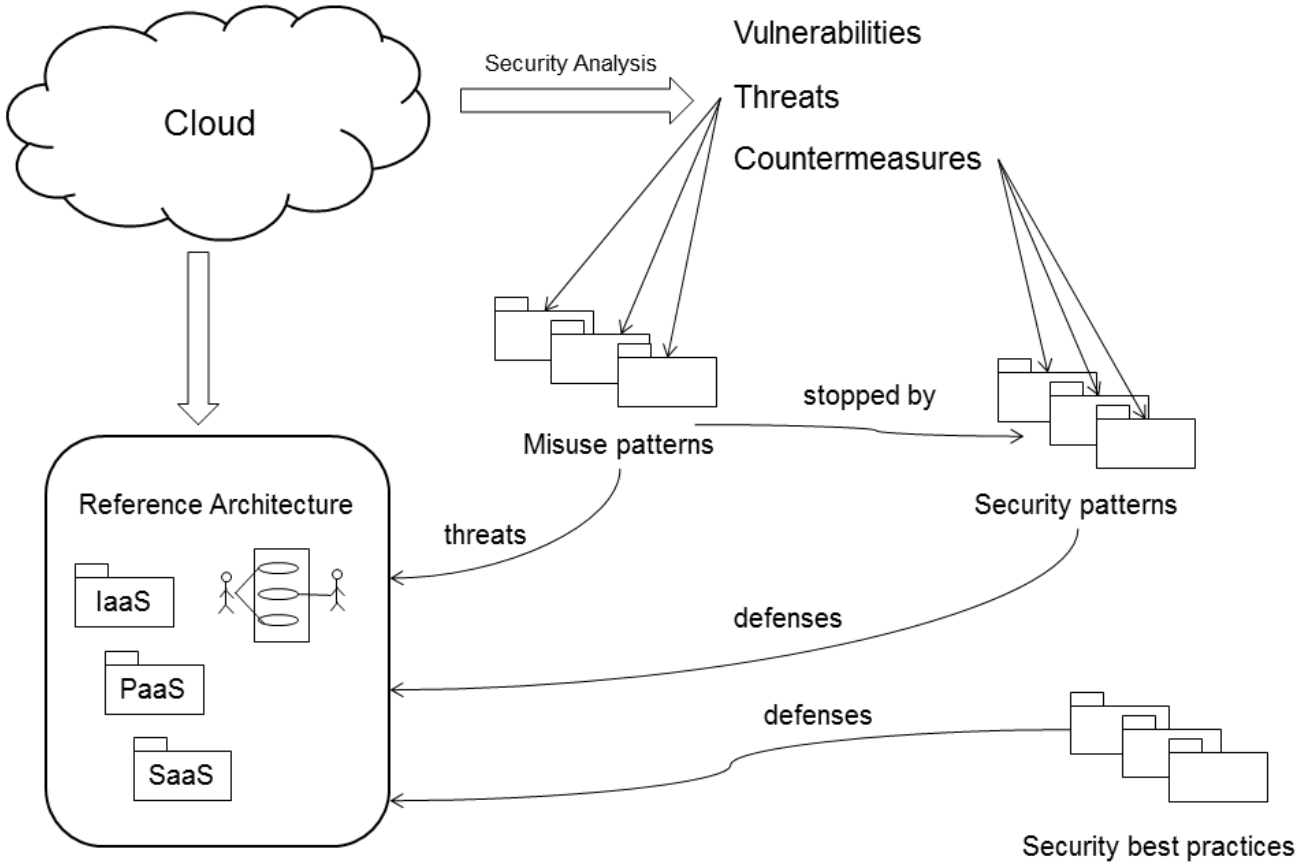


Class Diagram for a Cloud Computing Environment

Malicious VM Creation

- ***Intent***
  - An attacker may create a VM image that contains malicious code.
  - The attacker may read also confidential data from images.
- ***Context***
  - Some IaaS providers offer a VM image repository where users can retrieve images in order to initialize their VM.

Secure Reference Architecture



Securing a cloud reference architecture

Conclusions I

- ***We considered the use of security patterns and looked in detail at some of them.***
- ***We classified security patterns using architectural levels and surveyed some patterns previously developed by us and others***
- ***We considered a methodology to apply security patterns to build secure systems***
- ***Patterns are also valuable for evaluating existing systems and for teaching security concepts***

Conclusions II

- ***Patterns cannot prevent attacks that happen through code flaws but can make their effect much less harmful***
- ***Fixing design errors after coding is very expensive***
- ***Patterns can be made more formal: OCL***
- ***Security patterns are now accepted by many companies, Microsoft, Sun, and IBM have books, papers, and web pages on this subject. A general page for security patterns: www.securitypatterns.org***

Future Work

- ***Some security defenses can be represented as a form of security patterns including:***
  - Secure migration process
  - Secure hypervisor
  - Secure virtual networks
  - Virtualized trusted platform module
  - Cloud data protection
  - Secure VMI repository

Future Work

- ***Possible uses of a reference architecture:***
  - It can be used as a reference for security certification of services.
  - It can be used to support standards.
  - It can be used to provide general information before migrating any existing process or system to a cloud.
  - To define SLAs

Future Work

- ***Completing the catalog of misuse patterns***
  - Covert channels in clouds
  - Virtual machine escape
  - Virtual machine hopping
  - Sniffing virtual networks
  - Spoofing virtual networks

Publicaciones

- ***Anton Uzunov, E.B.Fernandez, and Katrina Falkner, "Securing Distributed Systems using Patterns: A Survey", Computers & Security, 2012 http://dx.doi.org/10.1016/j.cose.2012.04.005***
- ***Fernandez, E.B.; Ajaj, O.; Buckley, I.; Delessy-Gassant, N.; Hashizume, K.; Larrondo-Petrie, M.M. A Survey of Patterns for Web Services Security and Reliability Standards. Future Internet 2012, 4, 430-450. http://www.mdpi.com/1999-5903/4/2/430/***
- ***M. VanHilst, E.B.Fernandez, and F. Braz, "A multidimensional classification for users of security patterns", Journal of Research and Practice in Information Technology, vol. 41, No 2, May 2009, 87-97***

Publications

- *K. Hashizume, D. G. Rosado, E. Fernandez-Medina, and E. B. Fernandez, "An Analysis of Security issues for Cloud Computing," accepted for the Journal of Internet Computing.*
- *K. Hashizume, E. B. Fernandez, and M. M. Larrondo-Petrie, "Cloud Service Model Patterns," in 19th Conference on Pattern Languages of Programs, 2012.*
- *K. Hashizume, E. B. Fernandez, and M. M. Larrondo-Petrie, "A pattern for Software-as-a-Service in Clouds," in Workshop on Redefining and Integrating Security Engineering (RISE'12), Washington, DC, USA, 2012.*
- *K. Hashizume, E. B. Fernandez, and M. M. Larrondo-Petrie, "Cloud Infrastructure Pattern," in First International Symposium on Software Architecture and Patterns, in conjunction with the 10th Latin American and Caribbean Conference for Engineering and Technology, Panama, 2012.*
- *K. Hashizume, N. Yoshioka, and E. B. Fernandez, "Three Misuse Patterns for Cloud Computing," in Security Engineering for Cloud Computing: Approaches and Tools, D. G. Rosado, D. Mellado, E. Fernandez-Medina, and M. Piattini, Eds. IGI Global, 2013, pp. 36–53.*
- *K. Hashizume, N. Yoshioka, and E.B.Fernandez, "Misuse Patterns for Cloud Computing", Procs. of Asian PLoP 2011.*
- *Keiko Hashizume, Eduardo B. Fernandez, and Nobukazu Yoshioka, "Misuse patterns for cloud computing: Malicious virtual machine creation"", Procs. of the Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011), Miami Beach, USA, July 7-9, 2011, (Acceptance rate: 31%)*