Software and Systems Modeling manuscript No.
(will be inserted by the editor)

A Security Requirements Modelling Language for Cloud Computing Environments

Haralambos Mouratidis[1], **Shaun Shei**[1], **Aidan Delaney**[1]

Centre for Secure, Intelligent and Usable Systems, University of Brighton, School of Computing, Engineering and Mathematics, Brighton, UK

Received: date / Revised version: date

Abstract This paper presents a novel security modelling language and a set of original analysis techniques, for capturing and analysing security requirements for cloud computing environments. The novelty of the language lies in the integration of concepts from cloud computing, with concepts from security and goal-oriented requirements engineering to elicit, model and analyse security requirements for cloud infrastructures. We then propose three analysis techniques, which support an automated process where given a model of a cloud computing systems, developed with the proposed language, will enhance the model with new security knowledge, for example threats and vulnerabilities, mitigation strategies and assets and actor responsibilities. This is, to the best of our knowledge, the first attempt in the literature to develop a language for cloud computing security modelling and analysis, based on such integration and support it with a set of automated techniques that enhanced the stakeholder created models with security knowledge. The proposed modelling language and techniques are illustrated through walking examples and a case study based on our work in the VisiOn European project.

1 Introduction

The premise of the cloud computing paradigm is that computing resources are offered by third party providers as a form of commodity accessed through network connections [7]. In comparison to traditional IT solutions, this lowers capital costs and abstracts away implementation and infrastructure details by allowing cloud users to select from pre-configured computing services. However one of the prerequisites for cloud computing; outsourcing data and processes to third parties, raises several security [8,27] and legal questions [9]. To the cloud user, cloud computing is a black box where the user has little to no control over how or where their data is processed. In order to understand the security issues in cloud computing environments, developers need to capture and describe components interconnected through multiple conceptual layers, such as data, cloud services and cloud infrastructure. In order for cloud service providers to service multiple cloud users, multi-tenancy is an important architecture in cloud computing. Multi-tenancy refers to multiple cloud users running independent logical processes but sharing the same physical components, such as CPU, RAM and storage. Multi-tenancy in cloud computing systems is enabled through virtualisation. Virtualisation is the enabling technology for virtual machines, which emulates a physical server and is managed through software known as hypervisors. Therefore a single physical server can host a hypervisor managing one or more virtual machines. Each virtual machine is then allocated to a cloud user and runs cloud services. However from a cloud computing context, the mutual distrust in multi-tenancy environments brings up questions about the security of user data when sharing physical infrastructure [10]. For example consider the scenario where two companies are using virtual machines hosted on the same physical server. A virtual machine escape vulnerability [12] can be exploited, enabling one company to access the sensitive data of another company. This attack has been practically demonstrated in [13], in order to extract information from a target co-residing virtual machine.

Our work provides benefits to a number of users including those involved in the process of securing cloud computing systems, such as organisational stakeholders that wish to migrate aspects of their business system to the cloud; security engineers modelling and analysing cloud environments and cloud users that wish to have a better understanding of the security challenges in cloud systems. Therefore in the context of this paper, we refer to the term developers as the users of our work, for example organisational stakeholders and cloud security engineers under their employment.

Send offprint requests to:

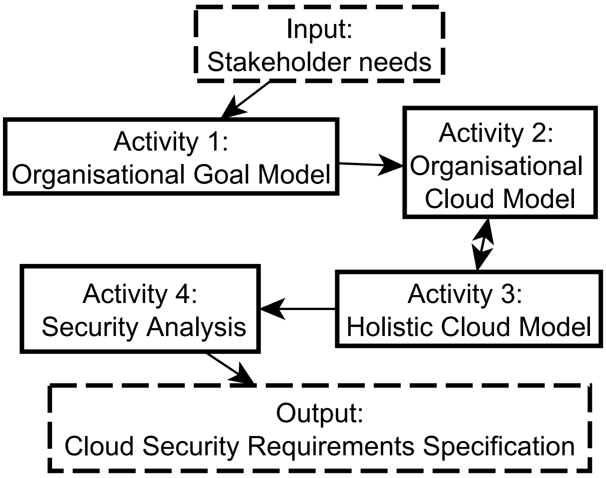


Fig. 1 Outline of the secure cloud requirements elicitation process.

The work presented in this paper is part of a doctoral thesis and has been refined through several iterations of published work, the latest appearing in [14]. The work in [14] focuses on describing the components and properties of the proposed cloud modelling language, which has been refined over the course of research, in order to capture cloud domain-specific concepts. This paper builds upon this work and presents a novel modelling language that uniquely combines concepts from cloud computing, security engineering and goal-oriented requirements engineering [11], a machine-readable syntax in order to facilitate semi-automated reasoning through tool-support, and a concrete syntax, which maps graphical notation to each of the concepts in the language. This is proposed in order to address the need for semi-automated tool-support, which would assist requirement engineers to model and discover the security issues and solutions in cloud computing environments. The paper also presents a set of original analysis techniques, which support an automated process where given a model of a cloud computing systems, developed with the proposed language, will enhance the model with new security knowledge, for example threats and vulnerabilities, mitigation strategies and assets and actor responsibilities.

This is, to the best of our knowledge, the first attempt in the literature to develop a language for cloud computing security modelling and analysis, based on such integration and support it with a set of automated techniques that enhanced the stakeholder created models with security knowledge. In particular, our contributions in this paper are as follows:

– *C1*: A cloud meta-model aligning concepts between security engineering, goal-based requirements engineering and cloud computing.
– *C2*: A modelling language describing cloud computing concepts, relationships and properties.
– *C3*: A machine-readable syntax notation to unambiguously describe the modelling language.
– *C4*: Three security analysis techniques; cloud security analysis, security mitigation analysis, and transparency analysis.

The overall aim of the work is to provide a decision support framework enabling developers to elicit cloud security requirements from cloud computing environments. The purpose of the framework is to provide an integrated set of domain-specific concepts and functionality, which enables the systematic application of a rigorously defined process. In the doctoral thesis we have described a framework consisting of a modelling language, the secure cloud process to systematically apply the concepts to the system-under-design and techniques to facilitate semi-automated security analysis. In the scope of this paper, we briefly introduce the process required to understand how our proposed modelling language and analysis techniques are applied in order to capture and analyse security requirements for cloud computing environments.

An outline of the process is illustrated in Fig. 1, showing the four activities involved from the initial requirements elicitation and the input and output of the process. Activity 1 in the process captures the information of traditional software systems and applies our domain-specific knowledge in cloud security using the propose modelling language. Activity 2 centres around identifying and describing cloud services, stakeholders and assets in the cloud environment with their security needs. Activity 3 guides the developer through the process of capturing the physical, virtual and social concepts and components involved in the cloud computing environment. Activity 4 describes how our proposed analysis techniques help guide the developer in performing security analysis on the cloud models produced throughout the previous activities, to ensure models are well-formed and enrich the system-under-design with additional knowledge through analysis techniques.

The rest of the paper is structured as follows. The cloud meta-model, cloud modelling language and syntax are defined in Sect. 2. In Sect. 3 we present our cloud security analysis techniques, building upon the cloud modelling language to provide support for semi-automated reasoning. We have used a university cloud computing environment as an example for the development of the models and the demonstration of the analysis techniques. In Section 4 we present brief information about a tool that supports the described language and techniques. A motivating case study based on our work in the VisiOn European project is described in Sect. 5. In Sect. 6 we discuss the related work. Finally we conclude the paper in Sect. 7, noting the on-going work and contributions.

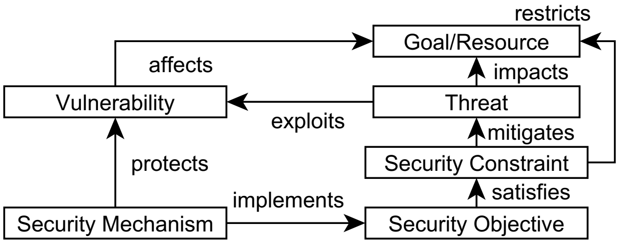


Fig. 2 The relationships between the security concepts in our research.

2 The Secure Cloud Modelling Language

In this section we present our secure cloud modelling language in order to define the concepts and relationships required to model cloud computing systems from a security requirements perspective. A simplified fragment of the cloud meta-model is shown in Fig. 3, illustrating concepts as boxes and relationships as shaded boxes, where boxes with thick outlines indicate the key cloud computing concepts and relationships proposed in our work. A more detailed description of the concepts and relationships of the language and a full version of the cloud meta-model please refer to [15]. Concepts extended from the Secure Tropos methodology [16] is shown as boxes with thin outlines.

In order to analyse the security of cloud systems, we focus on the concepts of resources and goals which represent abstractions of the services offered by the cloud system and the physical and virtual resources required to achieve these goals. This centres around the relationship of cloud services within the scope of the system, encapsulating information about the input and output components. Thus in the event of a security incident targeting assets in the system, we are able to model and trace the security consequences. There security analysis techniques can be performed on the cloud model generated through our cloud modelling language in order to explain the security consequences in detail, for example determining the impact of a threat in terms of the value metric associated with targeted assets, outlining the cost metric associated with implementing security measures to mitigate threats or validating the satisfaction of security properties against user-defined rules. The security analysis chain shown in Fig. 2 illustrates the relationship between the security concepts that are defined in our research. This security analysis chain governs whether the security concepts presented in a cloud model is well-formed, that is, the security associations and values are valid according to the cloud modelling language presented in this paper. We now present our novel cloud computing concepts.

2.1 Proposed Cloud Computing Concepts

In this subsection we introduce our concepts for modelling cloud computing systems, in the context of security requirements engineering. To keep the paper to reasonable length, we present a simplified fragment of the metamodel, shown in Fig. 3, focusing on the main cloud computing concepts. For a detailed representation of the meta-model, including "focused" metamodels for each concept, please refer to [15]. Later on in Sect. 5 we demonstrate how our concepts are applied from a practitioners perspective in the context of a case study.

Cloud Service Goal: *A cloud service goal provides a specific computing capability, is managed and owned by actors and requires virtual and physical resources in order to deliver its capability.* The cloud service goal concept is a specialisation of the goal concept found in the Secure Tropos methodology [16], which represents a way to achieve a specific need. We define a cloud service goal to embody a way of achieving a specific stakeholder need, through cloud computing capabilities. The Cloud Service Goal concept has the properties Capability, Security Property, Deployment Model and Service Model, indicating the specific computing capability, security property, cloud service deployment model and the service model. within the context of security. The *Deployment Model* determines specific threats and vulnerabilities affecting public, private, hybrid or community models and the *Service Model* determines cloud-specific threats and vulnerabilities impacting different levels such as Infrastructure as a Service(IaaS), Platform as a Service(PaaS) and Software as a Service(SaaS).

Virtual Resource: *A virtual resource represents intangible assets in a cloud computing system.* In order to differentiate between tangible and intangible resources, we create a specialisation of the resource concept to represent intangible resources as virtual resources which can be of *Type Data* or *Software* to determine jurisdictional properties. An example of an intangible resource is student grades of type *data*, which has the *group* visibility limiting access to actors within the defined group.

The *Virtual Resource* concept has the properties *Type* and *Visibility*. The *Type* property denotes the type of resource using the enumeration *ResourceType*, with values; *Data* and *Software*. The Visibility property denotes the level of visibility of a resource, using the enumeration *Visibility* with values; *Public*, *Private* and *Group*.

Physical Infrastructure: *A physical infrastructure represents a tangible system which, given a geographical location, hosts a group of physical assets within its local proximity.* We define this concept as a specialisation of the resource concept, given that cloud computing resources are hosted in physical infrastructure such as a data-centre. This is essential as properties belonging to the physical infrastructure contain fields such as geographical location, ownership and responsible parties; which is required for performing security analysis within

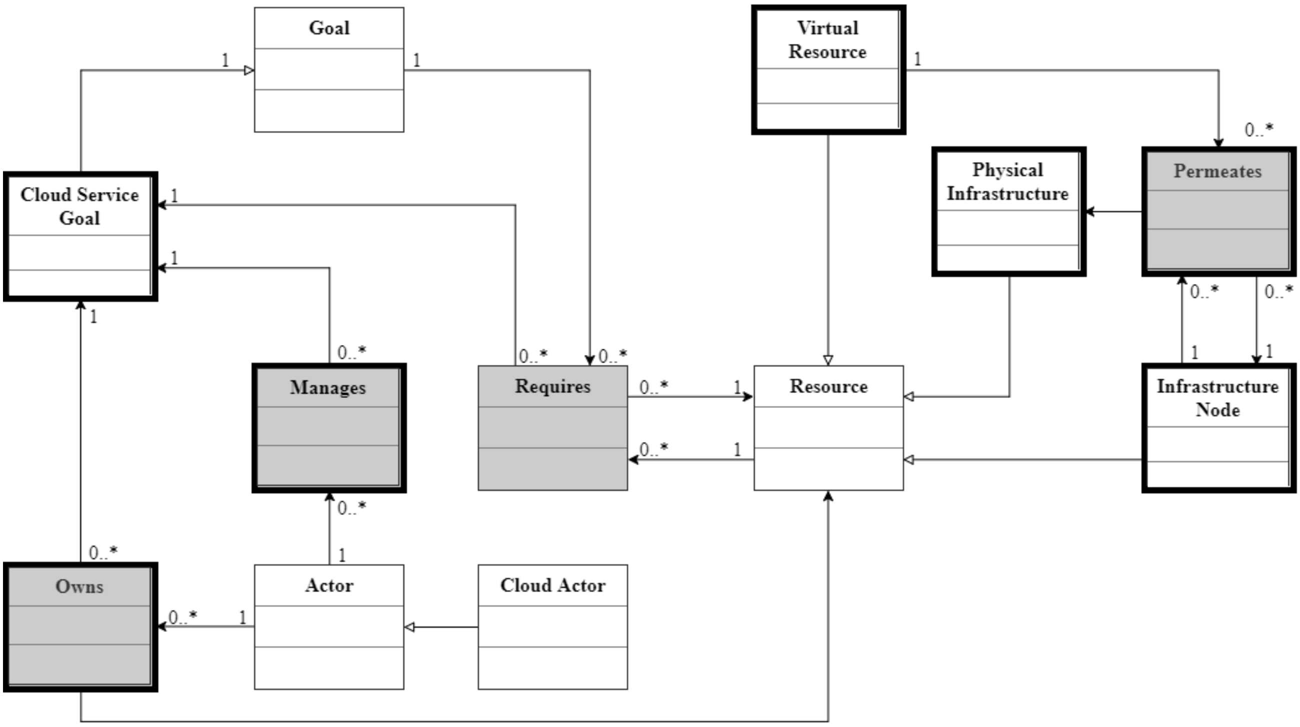


Fig. 3 A simplified fragment of the proposed meta-model showing cloud computing concepts.

a jurisdiction such as the EU *General Data Protection Regulation.* A physical infrastructure concept has zero or more infrastructure nodes, which conceptually represents the grouping of physical computational resources within an infrastructure.

The *Physical Infrastructure* concept has the properties *Jurisdiction.* The *Location* property denotes one or more jurisdictional constraints on the physical infrastructure using values defined in the enumeration *JurisdictionType*, for example the value *General Data Protection Regulation.*

Infrastructure Node: *An infrastructure node represents a single instance of a computing component such as a server, data storage or network connection.* A tangible resource is defined as a specialisation of the resource concept. The *NodeID* provides an unique identifier for each instance, which can be a *type* of compute, network or storage resource. We define the tenancy as single or multi-tenant, in order to determine cloud-specific threats such as hypervisor weakness leading to side-channel attacks between virtual machine instances [13].

The *Infrastructure Node* concept has the properties *Type* and *Tenancy.* The *Type* property denotes the type of infrastructure node, which is defined through the enumeration *NodeType* using the values *Compute*, *Network* and *Storage.* The *Tenancy* property denotes the tenancy of the infrastructure node, which is enumerated through *Tenancy* with the values *Single* and *Multiple.* The tenancy indicates whether an infrastructure node only involves a single unique user or if multiple users are involved.

The NodeID provides an unique identifier for each instance of an infrastructure node in a cloud model. The enumeration *JurisdictionType* consists of the following items: US, UK, EU and Asia. This represents which jurisdiction the asset falls under, and therefore has to adhere to. The enumeration *Tenancy* indicates whether a process is limited to a single tenant or if one or more users are involved.

Permeates: *This indicates the relationship which interrelates data-in-transit and data-at-rest from the virtual resource concept to the infrastructure node and physical infrastructure, and from the infrastructure node to another instance of the infrastructure node or a physical infrastructure.* A virtual resource is said to be traceable to an infrastructure node if the physical component hosts the virtual resource. For example when a cloud service processes user data stored on a physical hard drive, the data is traceable to the hard drive from a computation node.

A virtual resource is also traceable to a physical infrastructure given the traceable link to an infrastructure node that is part of the physical infrastructure. An infrastructure node can also be traceable to another infrastructure node or physical infrastructure given an exchange of information between the instances, for example a computation node requesting and processing data on a storage node.

Owns: *Owns indicates an actors level of responsibility as a relationship where the initiating actor possess ownership over a physical asset, is the creator of a virtual asset or has data ownership over a virtual asset.* For example in cases where a cloud users data is physically stored on assets owned by third party providers, who are responsible in meeting the security needs of the data.

This relationship is used to depict the level of responsibility an actor posses in relation to a cloud service goal goal or resource. It is important to define the difference between the data creator, data ownership and physical ownership. The data creator refers to the case where an actor produces data, and thus is the creator of the data in the legal sense. Data ownership refers to the case where data is physically stored on assets owned by third party providers, therefore the third party providers are responsible for the handling of the data. Physical ownership refers to the case where an actor is the owner of a physical asset, such as a server or data-centre.

The *Owns* relationship has the property *Responsibility*, which indicates the type of ownership an actor possess in relation to a cloud service goal or a resource and its specialisations. The enumeration *Ownership* contains the following values; data creator, data ownership and physical ownership.

An actor can initiate zero or more *Owns* relationships to a cloud service goal, resource or their specialisation. A cloud service goal, resource or their specialisation can be the target of zero or *Owns* relationships initiated from an actor. The data creator represents an actor that generates virtual resources, for example personal information created by a patient. The data ownership represents an actor in possession of virtual resources, for example a hospital is responsible for their patients medical records. The physical ownership represents actors who are responsible for processing resources through their own physical infrastructure, for example a cloud service provider has physical ownership over customer data stored on the cloud service providers physical infrastructure.

Manages: *Manages indicates an actors level of responsibility as a relationship, in the configuration and delivery of a cloud service goal.* Actors managing cloud services inherit the responsibility for meeting the security needs of resources required from the cloud service and its dependencies. An actor can have zero or more *Manages* relationships, indicating that some actors are not involved in the management of cloud services. A cloud service can be the target of one or more *Manages* relationships from a range of actors, indicating that a cloud service goal is managed by one or more actors.

The *Manages* concept has the properties *cloud service goal*, *Deployment Model*, *Service Model* and *Manager*, indicating the instance of the cloud service goal managed by the actor, the deployment model and service model the actor is responsible for and the instance of the actor initiating the relationship. The *cloud service goal* property holds an instance of the *cloud service goal* concept, representing the target of the manage relationship. The *Deployment Model* property specifies which type of cloud deployment model is managed, where the enumeration *DeploymentModel* includes the values *Public*, *Private*, *Hybrid* and *Community*. The *Service Model* property specifies which level of the service model an actor manages, where the enumeration *ServiceModel* includes the values *SaaS*, *PaaS*, *IaaS* and *XaaS*. The *Manager* property holds an instance of the *Actor* concept, representing the actor managing the cloud service goal.

2.2 Extended Concepts and relationships

In this subsection we outline the extensions to existing concepts in the Secure Tropos methodology, bridging concepts from cloud computing to the security requirements engineering domain. These extensions include definition properties to existing concepts, allowing the language to express richer levels of information.

Several concepts have an associated **Security property**, which for the security constraint, security mechanism and security objective concepts, describes specific security needs such as confidentiality, integrity or availability on the concept. In the case of threats and vulnerabilities, this represents the impact of a breach in security, for example a threat with the integrity security property indicates an impact on the integrity of targeted concepts.

Cloud Actor: *The cloud actor concept has two properties; DeploymentModel representing deployment model and CloudActorType representing the cloud actor role.* NIST defines five types of cloud actors; Cloud Service Provider, Cloud Consumer, Cloud Broker, Cloud Carrier and Cloud Auditor [7]. The type of cloud actor determines the set of responsibilities and also constrains the validity of relationships with other concepts. For example a cloud consumer who isn't also a CSP cannot provide physical infrastructure to another CSP actor.

Requires: *A goal, cloud service goal or resource requires a cloud service goal or resource, in order to satisfy a stakeholder need, fulfil a capability or collaborate with other resources or cloud service goals.* This relationship indicates the resource or cloud service goal instances required by a goal, cloud service goal or resource. The *Filter Security Property* property can whitelist or blacklist specific security needs as security properties on associated concepts.

Impacts: *A goal, cloud service goal or resource is impacted by a threat, which threaten their security properties.* This relationship indicates a resource or cloud service is impacted by a specific instance of a threat, where a *Impact Metric* denotes the security consequences of a breach as a qualitative or quantitative value and the *Probability* indicates the likelihood of a threat impacting the concept.

Exploits: *A threat is able to exploit a vulnerability.* This relationship indicates that the specific instance of a threat exploits one or more vulnerabilities.

The *Preventable* property denotes if the relationship of an instance of a threat exploiting a vulnerability is preventable. This allows the analysis to determine if mitigation strategies exist.

2.3 Syntax

In this subsection we present the instance and concrete syntax of the cloud modelling language. We define the machine-readable syntax as machine-readable encoding of instantiated concepts, relationships and properties from our cloud meta-model. We define the concrete syntax as graphical representations of concepts, properties and relationships in our meta-model, which provides a unique one-to-one mapping of a concept from the metamodel to an graphical representation in a cloud model.

The purpose of the machine-readable syntax is to provide a formal and condensed representation of concept instances, which is machine readable in order to perform analysis on cloud models. Thus the machine-readable syntax allows the unambiguous encoding of concepts in a textual format, which describes the instantiated concepts from a cloud model to facilitate security analysis. The general structure of the machine-readable syntax is as follows: an instance of a concept is defined with a lower case abbreviation with parenthesis surrounding the properties and relationships of the instance, the properties and relationships of the instance is defined with upper case abbreviations inside the parenthesis.

The concrete syntax is visualised using graphical notation, where each concept in the modelling language is mapped to an unique graphical notation. The shapes of the graphical notation was chosen arbitrarily in order to distinguish between Secure Tropos notation and our novel cloud computing concepts.

We now describe the syntax of the modelling language using the following format: concept, machine-readable syntax, description and concrete syntax.

Cloud Service Goal: The machine-readable syntax of a cloud service goal is *cs(D,CAP,DM,SM)*. The machine-readable syntax for a cloud service goal describes the following: *cs()* describes an instance of a cloud service goal with associated concepts encapsulated inside the parenthesis, *D* provides a description of the cloud service goal, *CAP* describes the capability of the cloud service goal, *DM* is the deployment model and *SM* is the service model.

Cloud Actor: The machine-readable syntax of a cloud actor is *ca(D,[T])*. The machine-readable syntax for a cloud service goal provider describes the following: *D* is a description of the cloud service provider, *[T]* is a list denoting one or more roles played by a cloud actor. The type of role played by a cloud actor is selected from the enumeration *CloudActorType* with the following list of roles; Cloud Service provider(*csp*), Cloud Consumer(*cu*), Cloud Broker(*cb*), Cloud Carrier(*cc*), and Cloud Auditor(*ca*). For example a cloud service provider with the description *hospital* is encoded as *ca(hospital,[csp])*. In case of cloud actors playing multiple roles, for example *dropbox* who is a cloud service provider and also a cloud consumer is encoded as *ca(dropbox,[csp,cu])*.

Virtual Resource: The machine-readable syntax of a virtual resource is *vr(D,RT,V)*. The machine-readable syntax for a virtual resource describes the following: *D* is the description of the virtual resource instance, *RT* is the type of resource such as data or software, *V* is the type of visibility such as public, private or group. For example the virtual resource *Patient data* with the resource type *data* and visibility type *private* is encoded as *vr(data,private)*.

Physical Infrastructure: The machine-readable syntax of a physical infrastructure is pi(D,L).

The machine-readable syntax for a physical infrastructure describes the following: *D* is the description of the physical infrastructure, *L* is an enumerated list of jurisdictions which the physical infrastructure falls under. For example the physical infrastructure *hospital information system* with the jurisdiction *General Data Protection Regulation* (GDPR) is encoded as *pi(hospital_is,gdpr)*.

Infrastructure Node: The machine-readable syntax of a cloud service goal is in(D,NT,TE).

The machine-readable syntax for an infrastructure node describes the following: *D* is the description of the infrastructure node, *NT* determines the type of the infrastructure node such as network or storage, *TE* determines the type of tenancy as single or multi-tenant. For example the infrastructure node *amazon server 1* with the *compute* node type and *single* tenancy is encoded as in(amazon_server1,compute,single).

3 Cloud Analysis Techniques

In this section we define security analysis techniques to support the developer through the process of identifying threats and vulnerabilities in a cloud environment, proposing alternative mitigation measures and validating cloud security needs. For example the threat analysis identifies threats on cloud assets, where the developer can determine the impact of a threat in terms of the value metric associated with targeted assets. Following the security mitigation analysis, the developer is able to determine the cost metric associated with implementing security measures such as security mechanisms to mitigate threats or validating the satisfaction of security properties against user-defined rules. In order to support developers with expert knowledge of security issues and vulnerabilities, we consult the vulnerability feeds from the National Vulnerabilities Database(NVD), an U.S. government repository of standards-based vulnerability management data [21]. It is also worth mentioning that the analysis techniques are implemented through the framework's tool, which is described briefly in Section 4.

Figure 4 illustrates the three supported cloud analysis; 1.) cloud security analysis, 2.) security mitigation analysis, and 3.) transparency analysis, where the analysis technique of each analysis are labelled 1a., 1b. respectively i.e. vulnerability analysis and threat analysis are both cloud security analysis techniques. The input and output of each analysis technique indicates which subset of the cloud environment model(*CEM*) is used as input and updated respectively. The *CEM* is used as input for the analysis techniques, where the *CEM* consists of the cloud system mapped using the machine-readable syntax during the modelling process. The order of the analysis is interchangeable depending on the developer needs and richness of the cloud environment model. For example the developer can perform the security mitigation analysis on a cloud environment model which has already identified pre-existing threats and vulnerabilities. On the other hand a cloud environment model with an empty resource knowledge base i.e. no resources in the system, will not produce any vulnerabilities or threats because there are no resources to impact or affect.

We now summarise the three types of cloud analysis, where the analysis techniques of each analysis is discussed in reference to Fig. 4:

- **Cloud Security Analysis 1a. Vulnerability analysis:** We identify vulnerabilities affecting resources in the system-under-design, represented through the cloud environment model(*CEM*), by consulting the resource knowledge base(*RKB*). We refine the *CEM* by enumerating specific vulnerabilities which affects resources in the system, updating the cloud threat model(*CVM*). Thus the *CVM* contains information on which resources are exploitable and the specific vulnerabilities affecting the system-under-design.
- **Cloud Security Analysis 1b. Threat analysis:** We identify threats which exploit vulnerabilities in the system. According to the vulnerabilities exploited, we refine the *CEM* to enumerate in the cloud threat model(*CTM*), which goals or resources are impacted by specific threats. Thus the *CTM* provides information on which resources or goals are at risk as a result of vulnerabilities identified in the *CVM*, and the specific threat exploiting the associated vulnerabilities which impacts the security properties of resources of goals in the system. The actor model(*AM*) identifies any malicious actors posing a threat on the system.
- **Security Mitigation Analysis 2a Security constraint resolution:** We identify security constraints by consulting the organisation relationships(*OR*), given an *AM*, in order to mitigate threats in the system. The purpose of a security constraint is to address the specific security property compromised by threats identified in the *CVM*. We consult the *CEM* and update the *AM* to show security constraints placed on the system as a result of identified threats. Thus the *AM* provides information on what security constraints or security needs need to be satisfied, in order to mitigate threats on the system-under-design.
- **Security Mitigation Analysis 2b. Security objective resolution:** We identify security objectives by consulting the *CTM*, in order to determine if all the security constraints placed on the system are satisfied. We refine the *CEM* to enumerate in the *CTM* security objectives and which security properties they satisfy in relation to security constraints found in the *AM*. Thus the *CTM* advises the developer of the specific security objectives which should be implemented, in order to address the security constraints placed on the system-under-design.
- **Security Mitigation Analysis 2c. Security mechanism resolution:** We identify security mechanisms by consulting the *CVM* in order to protect against the exploitation of vulnerabilities in the system-under-design. Security mechanisms are implemented through security objectives, thus the *CEM* is refined through the *CVM* and *CTM* to enumerate how vulnerabilities are addressed by security mechanisms and which security objectives implements the defined countermeasures respectively.
- **Transparency Analysis 3a. Cloud Security Management** This analysis identifies the roles of cloud actors responsible for the management or ownership of physical and virtual assets, by consulting the *AM* and cross-referencing the organisational relationships (*OR*) with resources and goals found in the *RKB*. The identified relationships enumerating these roles are reflected in the *OR* and updates the *CEM* with any modifications made. The cloud actors responsible for managing and implementing security measures are also identified by consulting the *CVM*, *CTM* and *AM*. This analysis allows developers to summarise the delegation of responsibilities in order to ensure the cloud security requirements of the system is satisfied, updating the *CTM* and *CSM*.

The following subsections describes the steps of each stage in further detail, providing a systematic process to carry out cloud security analysis. Note that the notation of the process is described using the machine-readable syntax defined earlier in Section 2.3, where capital letters in italic indicates variables holding values i.e. X, Y, and the equals sign is the assignment operator i.e. $X=Y$ assigns the value in the variable Y to X. To support readers we have used a running example based on a university cloud computing environment focusing on the career office services that a university provides, building on our own understanding of such services.

3.1 Cloud Security Analysis

The first analysis takes as input a cloud model with the minimal concepts consisting of goals, actors and resources as shown in Figure 5. Goals are required in order

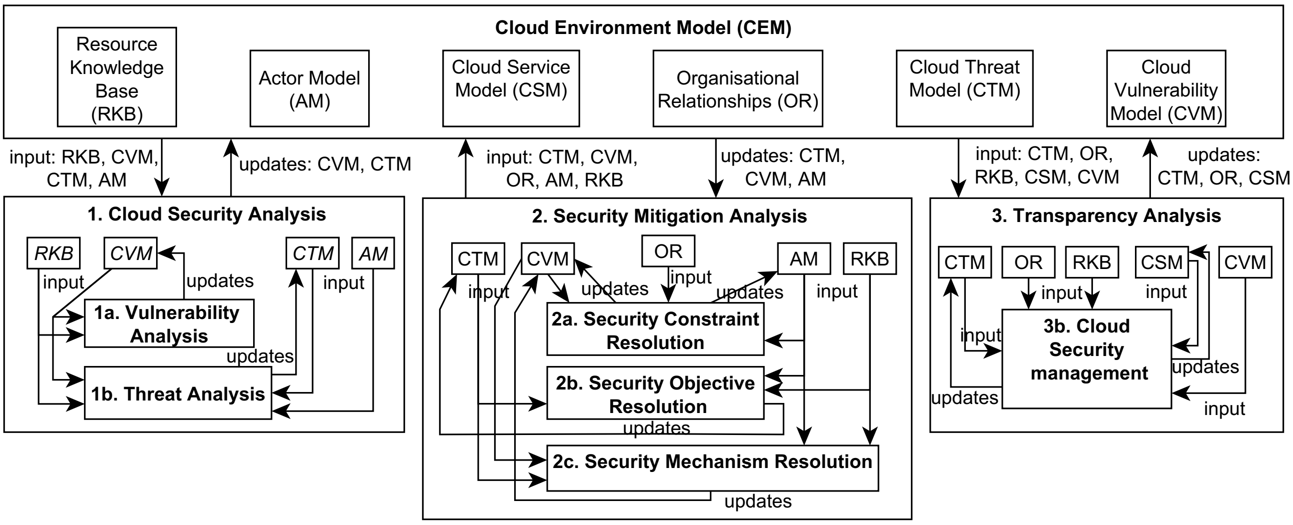
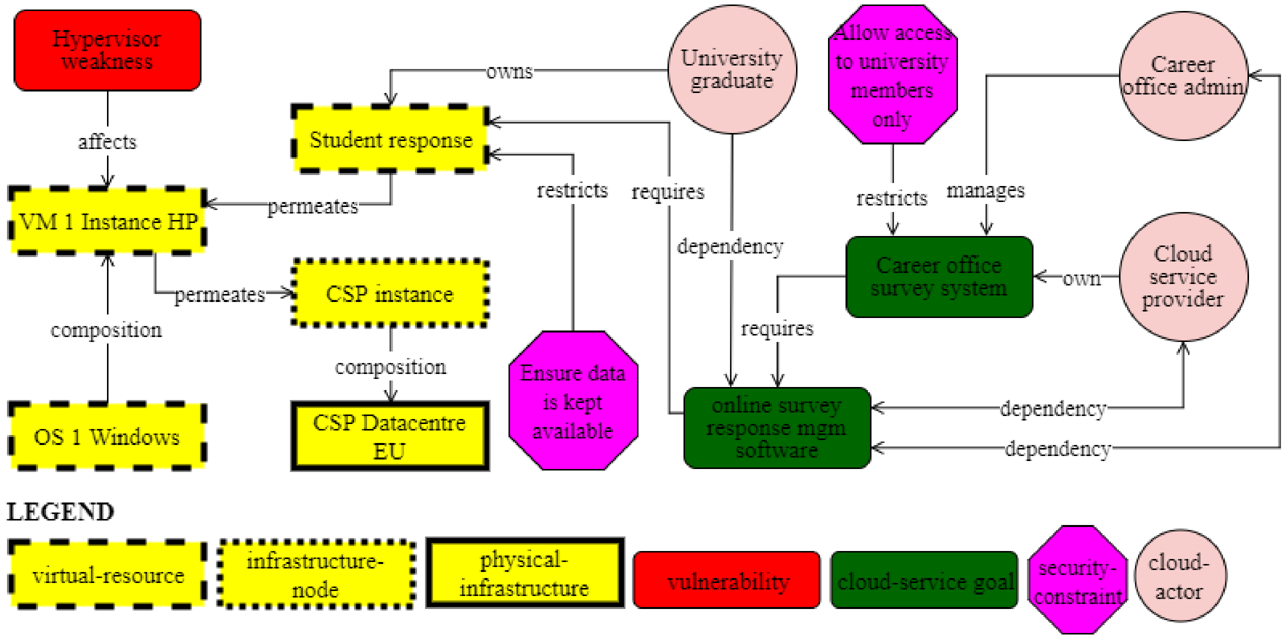

Cloud Environment Model (CEM)
Resource Knowledge Base (RKB)
Actor Model (AM)
Cloud Service Model (CSM)
Organisational Relationships (OR)
Cloud Threat Model (CTM)
Cloud Vulnerability Model (CVM)
input: RKB, CVM, CTM, AM
updates: CVM, CTM
input: CTM, CVM, OR, AM, RKB
updates: CTM, CVM, AM
input: CTM, OR, RKB, CSM, CVM
updates: CTM, OR, CSM
1. Cloud Security Analysis
RKB
CVM
CTM
AM
input
updates
input
1a. Vulnerability Analysis
updates
1b. Threat Analysis
2. Security Mitigation Analysis
CTM
CVM
OR
AM
RKB
input
updates
input
updates
input
2a. Security Constraint Resolution
2b. Security Objective Resolution
updates
2c. Security Mechanism Resolution
updates
3. Transparency Analysis
CTM
OR
RKB
CSM
CVM
input
input
input
3b. Cloud Security management
updates
updates
input

Fig. 4 Process illustrating three types of cloud analysis with inputs from and updates to the cloud environment model.



5.png

Fig. 5 A cloud model generated through the running example, which is provided as input for the analysis.

to capture the needs of the system. In addition cloud service goals are a specialisation of the goal concept, in order to describe the cloud computing needs of the system and stakeholders. Resources describe the components the system and processes require in order to fulfil their needs, which includes the cloud computing specialisations; virtual resource representing intangible resources, infrastructure node representing tangible resources and physical infrastructure representing boundary-specific components in the system. This analysis consists of two techniques; the vulnerability analysis and threat analysis.

In order to map security knowledge to our cloud models, vulnerability, threat and mitigation information is extracted from the NVD CVE XML 2.0 schema. We define NVD as a set and say that a quadruple *(cvename, product, vendor, version)* is in NVD if a NVD entry describes *cvename* in the *entry* element with *name* as the attribute, *product* in the *prod* element with *product* as the *name* attribute, *vendor* as the *vendor* attribute and *version* in the *vers* element with *version* as the *num* attribute.

3.1.1 Vulnerability Analysis Given a cloud environment model *CEM*, the vulnerability analysis examines the *RKB* of the *CEM* and searches the properties of assets within *RKB*. Specifically this analysis examines the properties of the set virtual resources *VR*, the set physical infrastructure *PI*, the set infrastructure node *IN* and their

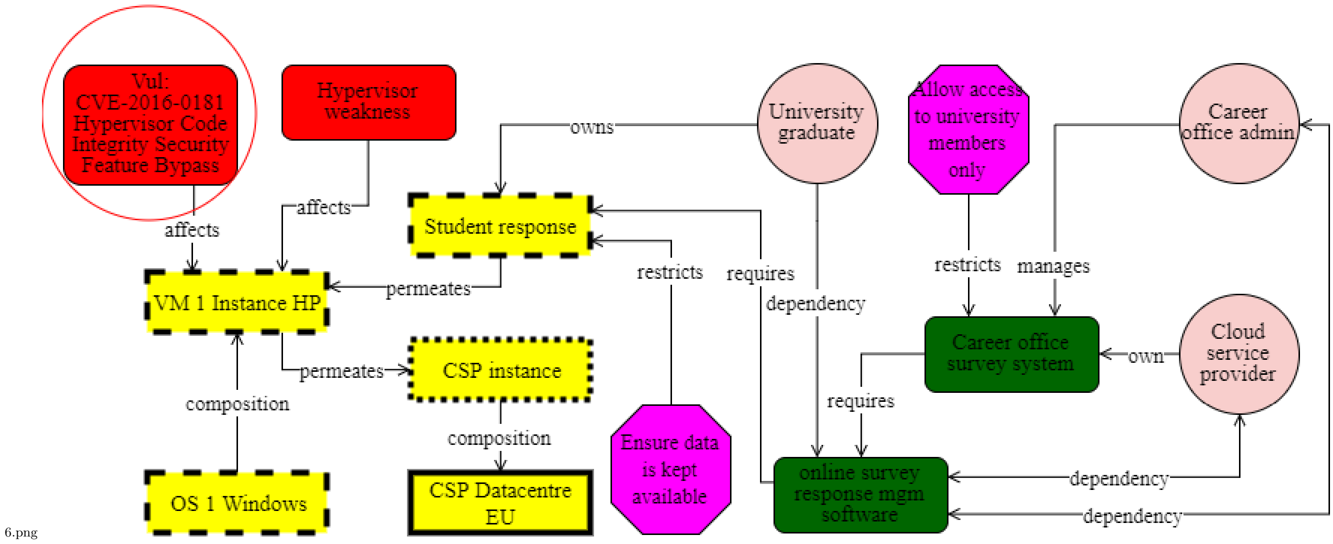


Fig. 6 A new vulnerability is identified and generated from the vulnerability analysis.

relationships *VPIR* in the *RKB*. Vulnerabilities are identified based on security knowledge and the *CVM* is updated. The following definition describes the procedure in order to perform the vulnerability analysis, where the *NVD* is used to identify and generate vulnerabilities affecting assets in a cloud model:

Definition: Vulnerability analysis For each (product, version, vendor) in a virtual resource, physical infrastructure or infrastructure node of *RKB* if (product, version, vendor) is in *NVD* then update the *CVM* with a new instance of the vulnerability concept with the *affects* relationship to the virtual resource, physical infrastructure or infrastructure node. The new vulnerability instance has the description *cvename* where *cvename* is the matching NVD entry given the quadruple (cvename, product, vendor, version).

For example given the cloud environment model shown in Figure 5, there are five entities in the *RKB* of the *CEM*; *VR1 Student response, VM 1 Instance HP, CSP instance, CSP Datacentre and OS 1 Windows*. The *CVM* has one existing vulnerability entry, enumerated as: *CVM(VM_1_Instance_HP,Hypervisor_weakness, affects(VM_1_Instance_HP,Hypervisor_weakness))*. When performing the vulnerability analysis, the properties in *VR1 Student response, VM 1 Instance HP, CSP instance and CSP Datacentre* do not match any entries in *NVD*. However the virtual resource *OS 1 Windows*, which is enumerated as *OS_1_ Windows(Windows,Microsoft, Windows_10)*, matches an entry in the *NVD* represented as a quadruple *(CVE-2016-0181,Windows,Microsoft, Windows_10)*. Therefore a new instance of the vulnerability concept was added to the *CVM* with the description *CVE-2016-0181* and the relationship *affects(CVE-2016-0181,OS_1_ Windows)*. Figure 6 illustrates the new vulnerability in the cloud model, after performing the vulnerability analysis. The *CVM* is now enumerated as: *CVM((VM_1_Instance_HP,OS_1_Windows),(Hypervisor_weakness,CVE-2016-0181),(affects(VM_1_Instance_HP, Hypervisor_weakness),affects(CVE-2016-0181,OS_1_ Windows)))*

3.1.2 Threat Analysis The threat analysis is the second analysis technique in the cloud security analysis, taking as input a *CEM*. Given a set of vulnerabilities in the *CVM*, these vulnerabilities can be exploited through threats posed by malicious actors in the *AM* to compromise the confidentiality, integrity or availability of resources and processes in the system. Therefore this analysis examines the *CVM* for vulnerabilities and existing threats in the *CTM*, generating new threats in the *CTM* to indicate the type of security property each threat impacts.

Definition: Threat analysis For each instance of a vulnerability *V* in the *CVM* of the *CEM* with the relationship *affects(V,R)* or *affects(V,G)*, if the description of *V* is in *NVD* then create a new instance of a threat *T* in the *CTM* with the following two properties; (i) the description *"Impact on (C,I,A)"* where (C,I,A) is a triple containing values from the corresponding *NVD* entry with *CVSS_vector* in the *entry* element and *C,I,A* as the attributes, (ii) the security property (C,I,A) where (C,I,A) is a triple containing values from the *NVD* entry with *CVSS_vector* in the *entry* element and *C,I,A* as the attributes.

The new instance of the threat *T* has the following relationships; (i) the *exploits* relationship to the corresponding vulnerability as *exploits(T,V)*, (ii) the *impacts* relationship to the resource *R* or goal *G* as *impacts(T,R)* or *impacts(T,G)*.

The Common Vulnerability Scoring System(CVSS) is a free and open industry standard for quantifying the severity of security vulnerabilities. The CVSS defines the impact of an exploit on a system using three security properties: "The confidentiality (C) metric describes the impact on the confidentiality of data processed by the system.", "The Integrity (I) metric describes the impact on the integrity of the exploited system." and "The availability (A) metric describes the impact on the availability of the target system. Attacks that consume network bandwidth, processor cycles, memory or any other resources affect the availability of a system.". The current version of CVSS (CVSS v3.0) was released in June 2015.

Therefore we align the impact of exploiting a vulnerability in CVSS as the impact of a threat on a resource and the exploitation of a vulnerability in our cloud modelling language. Specifically we take the three security properties defined in CVSS; confidentiality, integrity and availability to correspond to values defining security needs in the security property of our threat concept. That is we map on an one-to-one basis given the exploitation of a vulnerability in CVSS, the impact on one or more of the three security properties corresponds to a threat enumerating these security properties.

For example Figure 7 illustrates the *exploits* relationship of a threat on a vulnerability, where the threat also *impacts* a resource. In this example the vulnerability entry with the CVE identifier *CVE-2016-0181* is found in the NVD vulnerability data feed, where the following CVSS information is extracted for that entry; $< entry\ CVSS\_vector = "(C : N/I : P/A : N) >$. This extract from the CVSS indicates that there is no impact on the confidentiality property of resources affected by the vulnerability if exploited, partial impact on the integrity property of associated resources and no impact on the availability property. Therefore the threat with the name *"Impact on integrity"* with the set of security property $[I : P]$ is generated in the model, indicating the threat of partially impacting integrity if the associated vulnerability is exploited.

3.2 Security Mitigation Analysis

The second type of analysis supported is the security mitigation analysis, where the purpose is to examine threats and vulnerabilities from the *CVM* and *CTM* in order to propose potential approaches for mitigation. This analysis takes a *CEM* as input, given that there exists vulnerabilities in the *CVM* and threats in the *CTM.*

3.2.1 Security Constraint Resolution Security constraints represent the security needs of the system on assets or processes, where a security constraint needs to be satisfied in order to mitigate a threat. A security constraint mitigates a threat, indicated by the *mitigates(SC,T)* relationship in the *CTM.* A security constraint *SC* restricts a resource *R* and its specialisations or a goal *G* and the cloud service goal specialisation, indicated through the relationship *restrictsresource(SC,R,A)* and *restrictsggoal(SC,G,A)* in the *OR*. We define this technique to identify security constraints, create the *mitigates* relationship from a security constraint to a threat *T* and create the *restricts* relationship from a security constraint to a resource *R* or goal *G*. In the scope of this technique, *R* and *G* refers to the entry in the *impacts(T,R)* or *impacts(T,G)* relationships of the *CTM.* Therefore this analysis identifies the security needs of the system through security constraints, in order to address threats impacting resources and goals.

Definition: Security constraint resolution For each threat *T* in the *CTM* of the *CEM* where *mitigates(SC,T)* does not exist, then create an entry *mitigates(SC,T)* in the *CTM* where *SC* has the following properties; (i) the description *"Protect (C,I,A) of (R—G)"* where (C,I,A) is a triple in the security property of *T* and *R—G* is a single in the description of *R* or *G*, (ii) the security property *(C,I,A)* where (C,I,A) is a triple in the security property of *T*.

For each *impacts(T,R)* or *impacts(T,G)* in the *CTM* where there does not already exist a *restricts(SC,R)* or *restricts(SC,G)*, create the following relationship with the security constraint *SC*; (i) *restricts(SC,R)* or *restricts(SC,G)* where *R* or *G* corresponds to the entry found in the *impacts(T,R)* or *impacts(T,G)* relationships of the *CTM.*

For example Figure 8 shows a security constraint created in order to mitigate a threat, while restricting a resource. Following the threat analysis, we identify the threat named *Impact on integrity* in the cloud model, with the security property containing the set of values *I.* Taking the name of the threat as *Impact on integrity*, the target of the impacts relationship from the threat is identified as *OS 1 Windows*, which is assigned to *A*. The security constraint with the name *Protect integrity* is created, with the security property *I.* A mitigates relationship from the security constraint *Protect integrity* to the threat *Impact on integrity* is then created. Finally a restricts relationship from the security constraint *Protect integrity* to the resource *OS 1 Windows* is created.

After performing this analysis we have created a security constraint with the security property *I*, which has the mitigates relationship to the threat *Impact on integrity* and the restricts relationship to the resource *OS 1 Windows.* This security constraint therefore represents the security need which has to be satisfied to mitigate the indicated threat, where the security need is the security property integrity. The restricts relationship from the security constraint to the resource tells us which component in the system has security needs, specifically it tells us which security property needs to be satisfied.

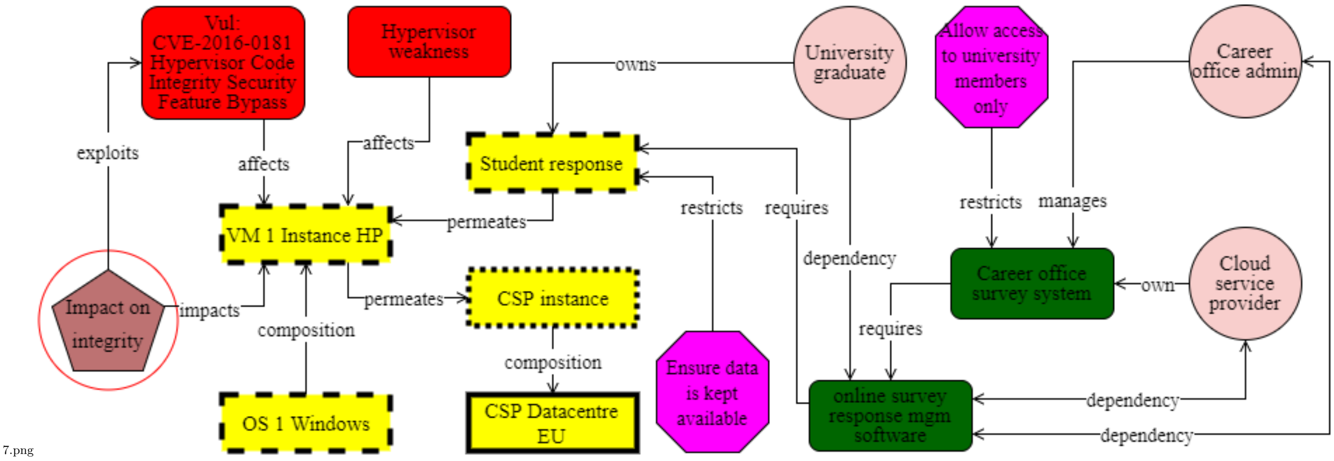


Fig. 7 Identifying a threat exploiting a vulnerability and impacting a resource.

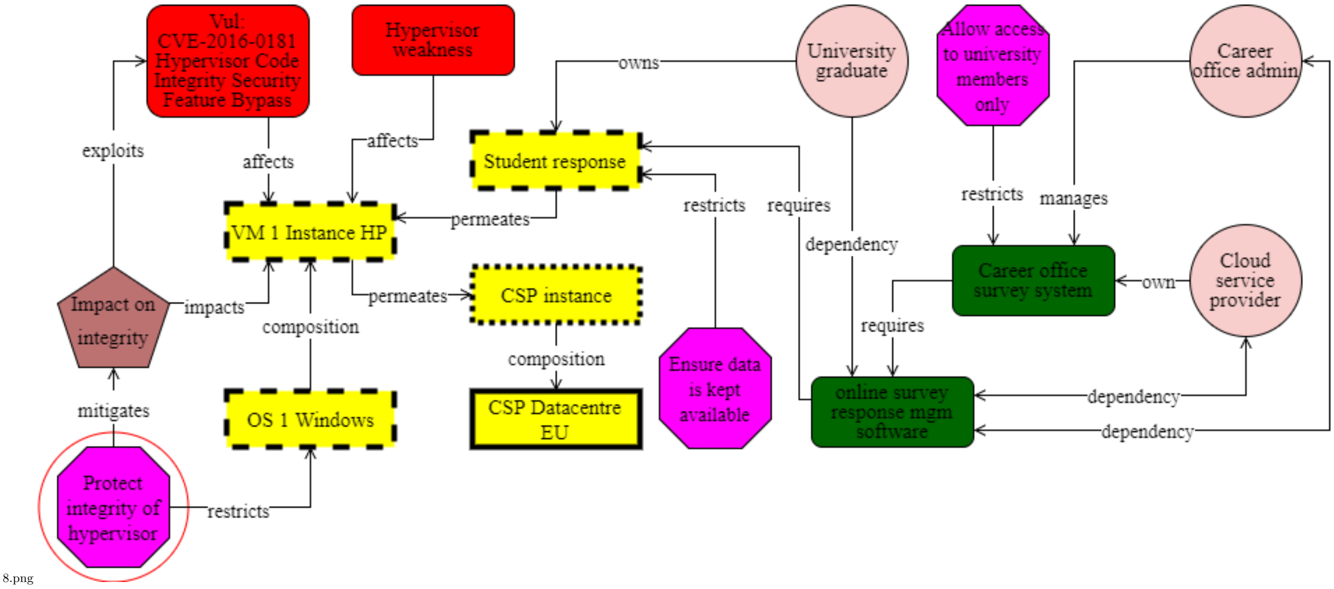


Fig. 8 Creating a security constraint which mitigates a threat and restricts a resource.

3.3 Security Objective Resolution

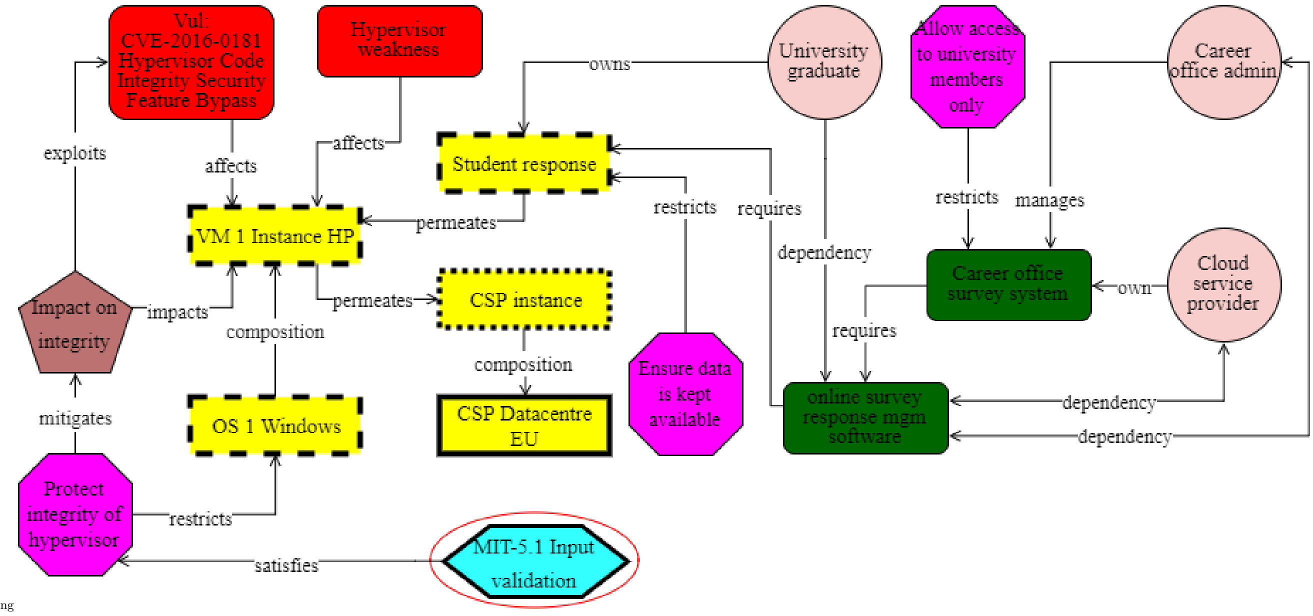
The second part of the security mitigation analysis takes as input the *CTM*, *AM* and *RKB*. The purpose of this analysis is to cross-reference CVE and CWE entries between the NVD vulnerability data feed in the NVD and the CWE database, in order to identify and create security objective entries in the *CTM*. Therefore in this technique security objectives are identified to satisfy security constraints through the *satisfies* relationship, where properties of the security objectives are extracted from entries in the NVD and CWE.

Definition: Security objective resolution For each security constraint *SC* in the *CTM* of the *CEM* where *satisfies(SO,SC)* does not exist, then create an entry *satisfies(SO,SC)* in the *CTM* where the security objective *SO* has the following properties; (i) the description *(M)* where (M) is a single that exists in the CWE core content database with the *Mitigation_Strategy* element and *M* attribute corresponding to the CWE entry, (ii) the security property *(C,I,A)* where (C,I,A) is a triple that exists in the CWE core content database which describes (C,I,A) in the *Consequence_Scope* attribute of the *Common_Consequence* element given a corresponding CWE entry.

Figure 9 shows the identification of a security objective as a result of consulting the CVE during the security objective resolution. In this case the security objective has the *securityProperty* of value *I* and the relationship *satisfies(MIT-5.1_Input_validation,Protect_integrity_of_hypervsior)*, which corresponds to the *securityProperty I* in *Protect_integrity_of_hypervsior*.

3.4 Security Mechanism Resolution

The final stage of the security mitigation analysis as input the *CVM*, *CTM*, *AM* and *RKB* from the *CEM*. The purpose of this technique is to identify the security mech-



9.png

Fig. 9 Creating a security objective which satisfies a security constraint.

anisms required to address vulnerabilities in the system-under-design. Specifically the CVE entry of vulnerabilities are cross-referenced through the NVD and CWE, where matching entries specify the mitigation method through the *protects(SM, V)* relationship.

Definition: Security mechanism resolution For each vulnerability *V* in the *CVM* of the *CEM* where *protects(SM, V)* does not exist, then create an entry *protects(SM, V)* in the *CVM* where the security mechanism *SM* has the following properties; (i) the description *(M)* where (M) is a single that exists in the CWE core content database with the *Mitigation_Description* element and *M* attribute corresponding to the CWE entry, (ii) the security property *(C,I,A)* where (C,I,A) is a triple that exists in the CWE core content database which describes (C,I,A) in the *Consequence_Scope* attribute of the *Common_Consequence* element given a corresponding CWE entry.

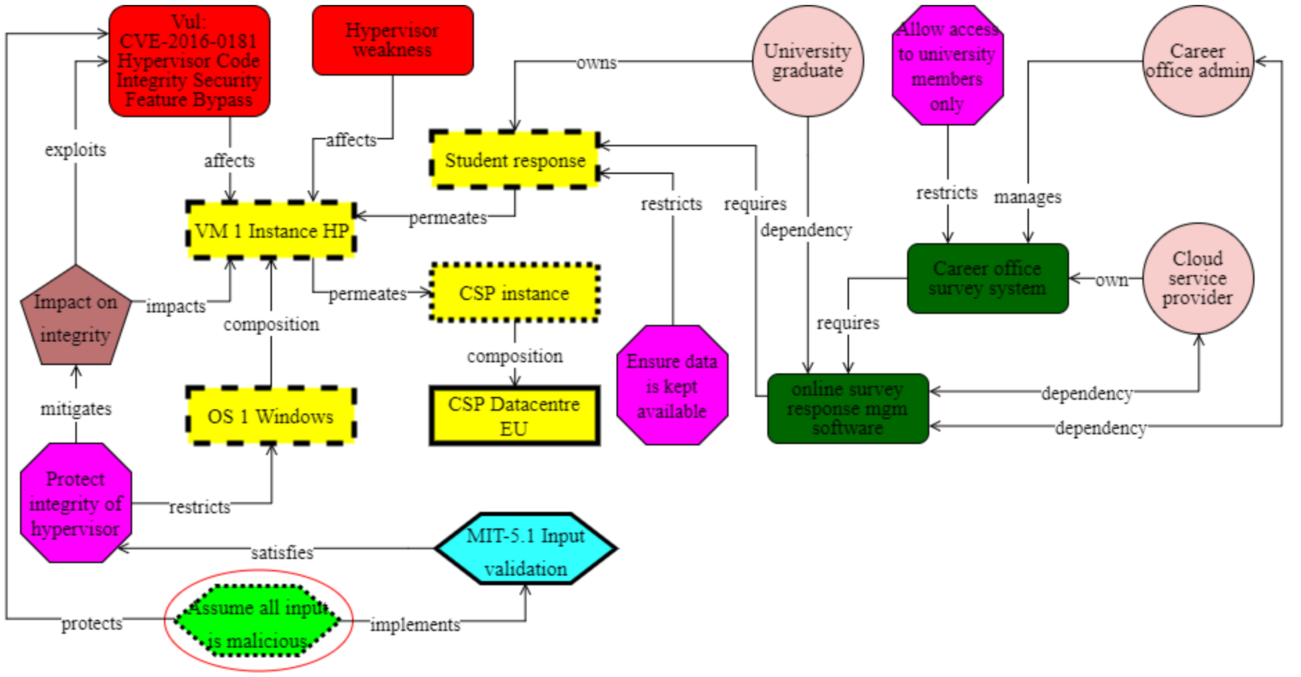
For each *protects(SM, V)* in the *CTM* where there does not exist an *implements(SO,SM)* relationship, create the following relationships with the security objective *SO*; (i) an *implements(SO,SM)* where the responsibility property of *implements* is the actor *A*.

Figure 10 shows a security mechanism which implements a security objective. In this case the security mechanism indicates how the security objective can be implemented, not the technical implementation instance or method. This limitation is due to the level of information provided in the security sources selected in our security analysis process. We argue that in the scope of security requirements engineering, providing exact technical implementation details during the early requirements stage is difficult due to the abstraction of components and concepts. That is during the design of the system-to-be, the developer does not posses enough information to be able to provide or make use of precise technical implementation level details.

3.5 Transparency Analysis

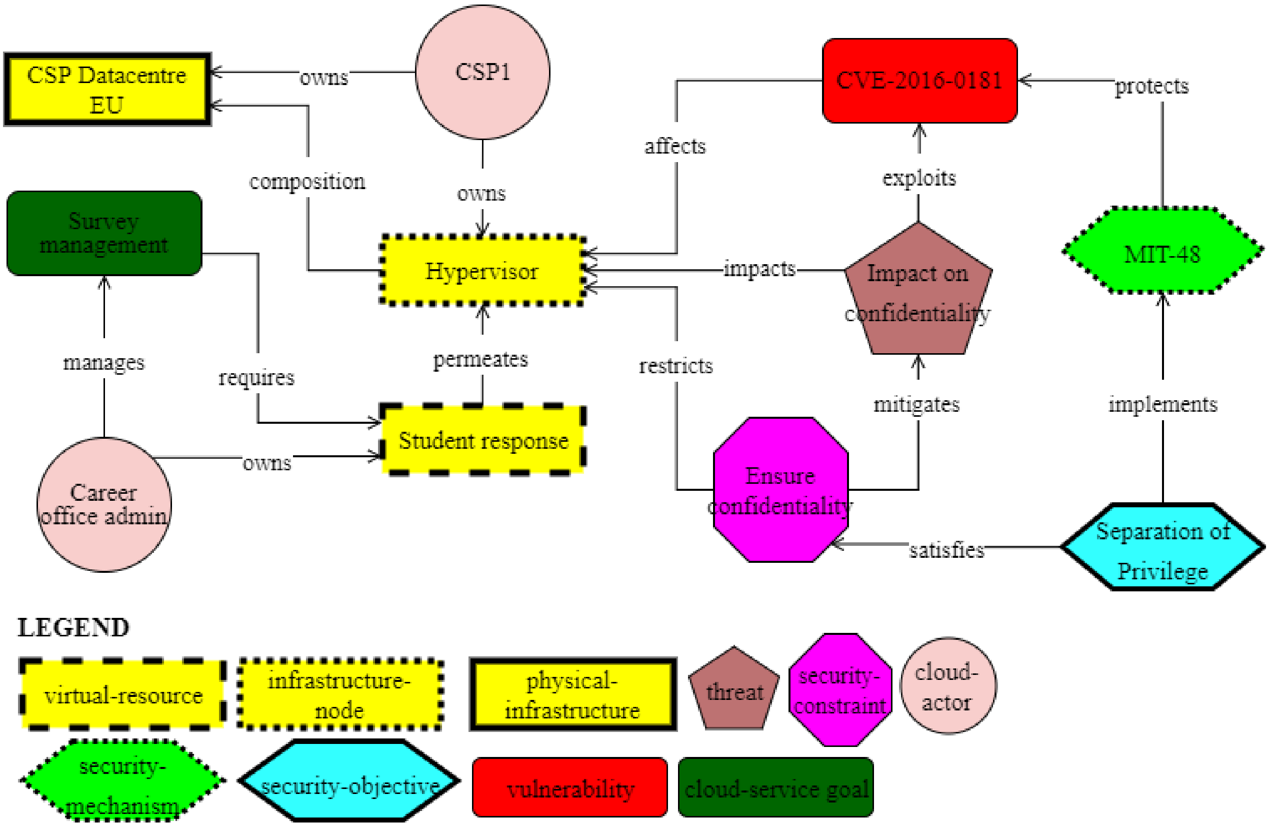
The analysis techniques defined previously have dealt with identifying security issues and solutions in a cloud computing environment, based on the properties of concepts such as cloud service goals and resources in the *CEM*. In order for developers to understand how cloud computing characteristics impacts the security needs in a cloud computing system, the responsibilities of cloud actors should be made clear. In this case there are several factors which determine a cloud actors scope of security responsibilities in a cloud computing environment. The primary concern is to determine the ownership and management of processes, data and infrastructure in a cloud computing system. Therefore by associating cloud actors with concepts in the system-under-design, the tractability of security responsibilities can be determined. As an example, given a cloud service goal with the SaaS cloud service model, the cloud service provider will be responsible for managing the infrastructure and applications required by the cloud service. Therefore the security implications in this example is that the cloud service provider will be responsible for addressing any security challenges that impact the infrastructure and applications associated with the cloud service.

3.5.1 Cloud Security Management The purpose of this analysis is to examine the security needs of a cloud computing system, in order to identify cloud actors responsible for satisfying these security needs. The analysis takes



10.png

Fig. 10 Creating a security mechanism which implements a security objective and protects a vulnerability.



11.png

Fig. 11 Cloud environment model showing the security needs of a system-under-design.

as input the *CVM*, *CTM*, *OR*, *RKB* and *CSM*. Specifically the analysis examines the *affects*, *exploits*, *impacts* and *restricts* relationships in the *CVM*, *CTM* and *OR* to determine the security challenges imposed on goals and resources. The *protects*, *satisfies*, *implements* and *mitigates* relationships in the *CVM* and *CTM* determine how to address security challenges in the system-under-design. The *owns*, *manages* and *poses* relationships in the *AM* indicates the association between cloud actors and the processes, data and infrastructure in the cloud computing environment.

Definition: Cloud security management For each security constraint *SC* of *CEM*, if *SC* is in *OR* where *restrictsresource(SC,R,A)* or *restrictsgoal(SC,G,A)* and *requiresgoal(G,R)* exists then each *SC* has the following values appended to its *SecurityRequirement* property:

- append *"The cloud actor A is responsible for SC on G R: "* where the properties of *A*, *SC*, *G* and *R* is in the *restrictsresource(SC,R,A)* or *restrictsgoal(SC,G,A)* of *OR*
- for each *affects(V,R)*, *affects(V,G)* and *protects(SM,V)* in *CVM* where *R* or *G* is the same scope of *SC* then append *"Security mechanism SM to protect against vulnerability V "*
- for each *exploits(T,V)* and *mitigates(SC,T)* in *CTM* where *T* and *SC* is in the same scope of *SC* then append *"Security constraint SC mitigates the threat T "*
- for each *satisfies(SO,SC)* and *implements(SO,SM)* in the *CTM* where *SC* and *SM* is in the same scope of *SC* then append *"Security objective SO satisfies security constraint SC and implements security mechanism SM"*

Figure 11 illustrates a cloud environment model consisting of the following concepts: cloud actor *Career office admin* and *CSP1*, cloud service goal *Survey management*, virtual resource *Student response*, infrastructure node *Hypervisor*, physical infrastructure *CSP Datacentre*, vulnerability *CVE-2016-0181*, threat *Impact on confidentiality*, security constraint *Ensure confidentiality*, security objective *Separation of Privilege* and security mechanism *MIT-48*. These concepts are formally represented in

$$CEM = \langle RKB, AM, OR, CSM, CVM, CTM \rangle$$

where relationships such as *restrictsresource(Ensure confidentiality,Hypervisor,CSP1)* in the *OR*, *protects(MIT-48,CVE-2016-0181)* in the *CVM* and *satisfies(Separation of Privilege,Ensure confidentiality)* in the *CTM* are described. These relationships are examined when performing the cloud security management analysis, describing the security needs of the system. After performing this analysis, the security constraint *Ensure confidentiality* in *CEM* has the following *securityRequirement* property representing the security responsibility of a cloud actor:

> *"The cloud actor CSP1 is responsible for Ensure confidentiality on Hypervisor: Security mechanism MIT-48 to protect against vulnerability CVE-2016-0181, Security constraint Ensure confidentiality mitigates the threat Impact on confidentiality, Security objective Separation of Privilege satisfies security constraint Ensure confidentiality and implements security mechanism MIT-48".*

In summary, after performing the analysis techniques in the security analysis process, a security-enhanced cloud model is produced by semi-automatically enriching the model with security concepts such as vulnerabilities, threats, security constraints, security objectives and security mechanisms. Specially the security constraints tells us which resources and goals in the system is impacted by threats or affected by vulnerabilities, and how they are restricted in terms of the security properties that need to be protected. The security objectives provide high level descriptions of what needs to be done in order to satisfy the security properties outlined in security constraints. The security mechanisms describe at a lower level, the method or technique which need to be carried out in order to implement security objectives.

While the granularity of the proposed concepts and mitigation strategy do not provide a direct way for developers to proceed to the implementation stage through coding guidelines, we argue that this is a limitation given the level of information provided from the security sources. Rather our security analysis approach guides the developer of cloud systems in analysing the goals and resources of a system, drawing from industry standard security sources to create and illustrate the relationships and concepts to help developers understand the security needs of the system-under-design.

4 Tool Support

Apparatus Software Tool(ASTo) is an open source graphical security analysis tool for Internet of Things(IoT) networks, providing tool support for the Apparatus security framework in [19]. In order to provide developers with semi-automated tool support when applying our secure cloud environment framework, our contribution is the SectroCloud module, which extends ASTo with cloud security requirements concepts. The automated reasoning support includes a graphical interface improving quality-of-life through task automation, the means to create visual models and perform semi-automated security analysis. The contributions of the SectroCloud module is as follows:

- Visualisation of the concepts and relationships defined by the modelling language.
- Graphical representation of the conceptual cloud environment model, with three views focusing on the organisational, application and infrastructure level of concepts.

- Provides semi-automated reasoning based on the three cloud analysis techniques defined in the previous section. In particular Cloud Security Analysis, Security Miitgation Analysis and Transparency Analysis. Taking as input models and information from public databases (as explained in the previous section) the tool enhances the models created by the user with relevant threat, vulnerability information.
- Quality-of-life utility and interface usability to support developers during the modelling process, such as concept and relationship filters, baseline security analysis techniques, data overview and model categorisation.

ASTo with the SectroCloud module is available for download from the authors online repository [1]. The instructions for installing the tool are provided in the online repository. One of the main issues in modelling complex systems in a cloud environment is the scalability and visual presentation of data, when developers need to model cloud systems with hundreds of nodes and relationships. With tool support, features such as the visualisation and grouping of concepts within a specific scope helps developers to uncouple complex models and understand the workings of the system under design.

5 VisiOn Case Study: Municipality of Athens

In this section we evaluate our modelling language and analysis techniques using a scenario from the Visual Privacy Management in User Centric Open Environment (VisiOn)[2] European project case study. VisiOn is an integrated tool solution that organisations can connect to part or all of their systems within their infrastructure. The VisiOn tool is accessed through a web interface by users such as citizens, allowing them to interact with the public administration(PA) in order to search for information, updates and requirements on the management of their data. This case study was selected because of the relevancy between the VisiOn project and the cloud computing needs of their stakeholders, specifically in scenarios where the cloud security requirements of stakeholders need to be understood as they transition from traditional systems to a cloud computing environment.

The scenario from this case study was selected from one of three pilot scenarios proposed by a local Greek government development company named "City of Athens IT Company" (DAEM) for the VisiOn project. In this scenario an Athenian city resident, George, submits a request for an annual membership at a public swimming pool in Athens, with a disability discount. In order to verify his current residence and medical situation to be eligible for specific city services, the swimming pool faculty is responsible for verifying the birth and medical certificates of the applicant, in order to approve the citizenship status and medical circumstances for a discount. The swimming pool administration team will outsource their computing needs to a third party cloud computing provider, where the business data and processes of their system is stored and processed externally in a cloud computing environment. The municipality of Athens is responsible for providing copies of various certificates of citizens, where their data is accessed and stored by certified governmental cloud computing providers. The clinic in this scenario stores the medical information and certificate of citizens on certified health-care cloud computing infrastructure. Therefore in this scenario the personal data of citizens is exchanged between the public authority from the municipality of Athens, personnel at a clinic and the swimming pool administration staff. This scenario is examined from the perspective of DAEM, where they are responsible for determining the cloud security requirements of the stakeholders in this scenario, and disseminating the identified requirements to the appropriate parties.

The stakeholders in this case study are the representatives of the public authority (Municipality of Athens), swimming pool administrator and Athenian citizens (George), where their requirements are represented by DAEM.

The description and security requirements of the stakeholders in the Municipality of Athens scenario were obtained in the context of the VisiOn project [3]. The security requirements are as follows:

- **SR1** The actor *SP Information System* require *Badge* to authenticate in order to achieve goal *Badge issued.*
- **SR2** The actor *SP Information System* require unlinkability on *Badge* in order to achieve goal *Badge issued.*
- **SR3** The actor *Municipality of Athens* shall ensure the integrity of the required resource *Medical certificate* permeating to *SP information system* is preserved.
- **SR4** The actor *Municipality of Athens* shall ensure the confidentiality of the required resource *Medical certificate* permeating to *SP information system* is preserved.
- **SR5** The actor *Municipality of Athens* shall ensure the integrity of the required resource *Birth certificate* permeating to *SP administrator* is preserved.
- **SR6** The actor *Municipality of Athens* shall ensure the confidentiality of the required resource *Birth certificate* permeating to *SP administrator* is preserved.
- **SR7** The actor *SP Information System* shall ensure the confidentiality of the required resource *Bank details* is preserved.

[1] https://github.com/NOMNUDS/apparatus

[2] http://www.visionprivacyplatform.eu

[3] http://www.visioneuproject.eu/wp-content/uploads/2016/03/2015-VSN-RP-053-D2.2-Citizens-and-Public-Administrations-Privacy-Requirements.pdf

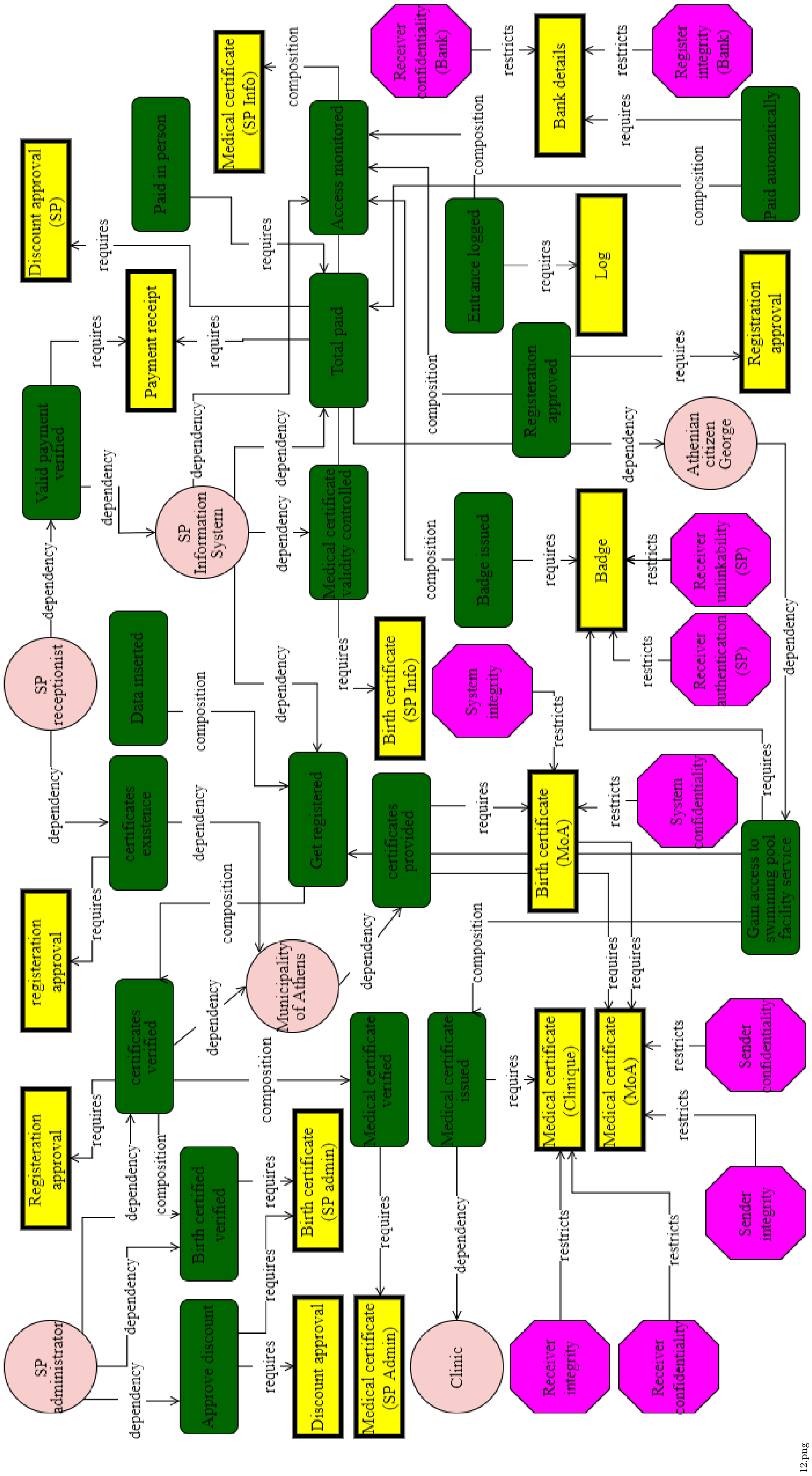


Fig. 12 Organisational goal model of the VisiOn Municipality of Athens scenario.

- **SR8** The actor *SP Information System* shall ensure the integrity of the required resource *Bank details* is preserved.
- **SR9** The actor *Clinic* shall ensure the integrity of the required resource *Medical certificate* is preserved when received by actor *SP information system.*
- **SR10** The actor *Clinic* shall ensure the confidentiality of the required resource *Medical certificate* is preserved when received by actor *SP information system.*

Taking these requirements into account, the aim of this scenario is to realise these requirements in a cloud computing environment. This is achieved by applying the secure cloud process illustrated previously in Figure 1 to refine cloud models and perform security analysis in order to produce cloud security requirements.

5.1 Outcome from the Application of the Secure Cloud Process

In this subsection we report on the outcomes of applying the secure cloud process to the Municipality of Athens scenario, in order to evaluate our proposed modelling language and analysis techniques. Here we outline the outcomes of each activity following the application of the secure cloud process, focusing on the output in terms of producing and refining cloud models throughout the modelling and security analysis activities without detailing each step.

5.1.1 Organisational Goal Model The Municipality of Athens scenario describes a situation where the primary goal of an Athenian citizens is a request for an annual membership for a local swimming pool in Athens, which is processed by the administration staff. In the process of determining the validity for a disability discount, the administration staff at the swimming pool requests access to copies of the citizens birth and medical certificates. This is carried out through communication with public administration staff, represented as the actors *Municipality of Athens* and medical personnel at a local clinic. Figure 12 shows a graphical representation of an organisational goal model in the Municipality of Athens scenario, generated using the SectroCloud tool. The organisational goal model was constructed by the author based on the information found in stakeholder requirements and specification documents available from the VisiOn European project. The actors in this scenario are; *Athenian citizen George*, *SP Information System*, *Municipality of Athens*, *SP administrator*, *Clinic* and *SP receptionist.* The primary goal of the actor *Athenian citizen George* in the scenario is *Gain access to swimming pool facility service.* This is composed of the sub-goals *Medical certificate issued* and *Get registered.* The model has six conceptual boundaries corresponding to each actor, where each bounded area includes the goals and resources of each actor. Several goals depend on the goals located in cross-boundary areas, denoting the exchange of data outside the scope of individual organisations or stakeholders. For example the *Municipality of Athens* depends on the *SP administration* to ensure the medical certificate is confidential during transmission.

5.1.2 Organisational Cloud System Following the creation of the organisational goal model, this next activity focuses on the identification and creation of cloud services, corresponding to the goals of the system and stakeholders. As this activity is performed from the perspective of DAEM, we identify the cloud service goals required in order to transmit, access and store data towards achieving the primary goal of *Gain access to swimming pool facility service* by the actor *Athenian citizen George.* Thus the following cloud service goals have been identified so far; *SP payment service*, *Access monitoring service*, *Certificate management service*, *PA banking service.* The *SP administration service* represents a cloud service goal which acts as a gateway to business processes essential for the operation of the swimming pool. The *Access monitoring service* represents a cloud service goal which is part of the *SP administration service*, specialising in capabilities within the scope of monitoring membership access. The *Certificate management service* represents a cloud service goal for managing the access, verification and issuing of medical and citizenship certificates. The *PA banking service* represents a cloud service goal used by public administration for banking services, such as accessing, withdrawing and setting up payments for bank accounts.

The cloud service filter shown in Figure 13 showcases how the graphical highlighting of a specific concept helps developers narrow the scope of a cloud model to cloud services. Thus given a large model composed of complex relationships and concepts, developers are able to spot patterns and manipulate the graphical representation of the cloud model to further their understanding. This was useful in this scenario due to the requirements for modelling and visualising the satisfaction of security properties, such as confidentiality and integrity on components, which involves understanding visually complex chains of relationships and properties. We now discuss how the relationships and properties of the cloud services identified in this activity are refined in the following activity.

5.1.3 Holistic Cloud Model In order to address the ten security requirements on the cloud system, represented through security constraints, the framework guides the developer through the process of devising the mitigation strategy. Therefore during the organisational modelling step, six security objectives were added to address the existing security constraints. Nine security mechanisms were proposed to implement all identified security objectives. Specifically the security constraints *Receiver authentication (SP)* and *Receiver unlinkability (SP)* are satisfied by the security objectives *Ensure authentication*

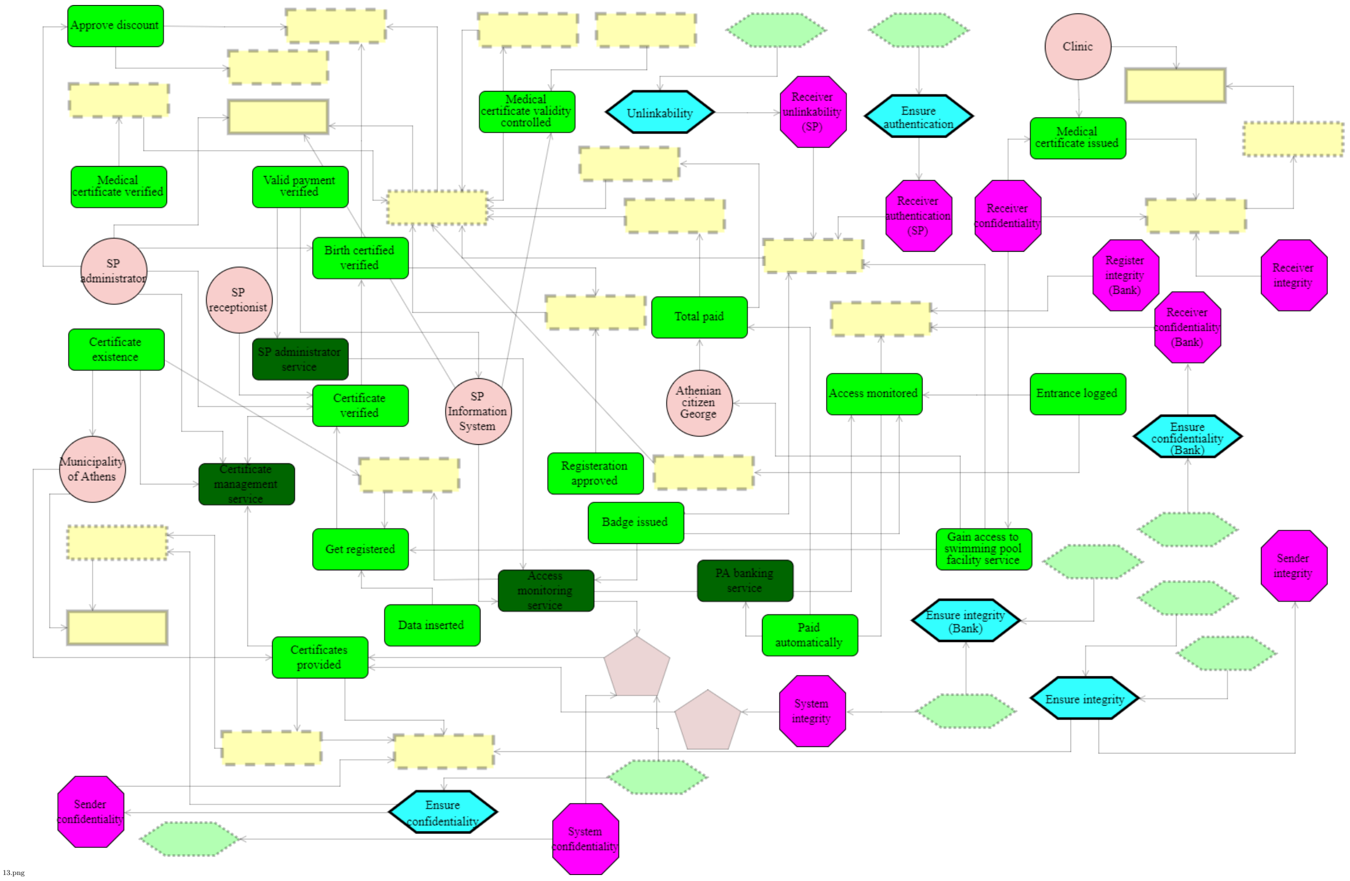


Fig. 13 View filter highlighting the cloud services in the scenario cloud model.

and *Unlinkability*, which are implemented by the security mechanisms *Username/password* and *Pseudonymizer tools* respectively. The security constraints *Receiver confidentiality*, *Receiver confidentiality (Bank)*, *Receiver integrity* and *Receiver integrity (Bank)* are satisfied by the security objectives *Ensure confidentiality* and *Ensure integrity*, which are implemented by the security mechanisms *Encryption*, *Host IDS* and *Network IDS* respectively. Finally the last set of security constraints *Sender integrity* and *Sender confidentiality* are satisfied by the security objectives *Ensure confidentiality* and *Ensure integrity*, and is implemented by the security mechanisms *Encryption* and *Mirroring* respectively. These security measures have been identified and represent at a high level, a mitigation strategy for the satisfaction of the cloud security needs. The holistic cloud model produced in the VisiOn case study is therefore shown in Figure 14.

5.1.4 Security Analysis Once the relevant model has been constructed, the next step involves the application of the security analysis process. It is worth mentioning that this section does not report on the complete analysis of the case study but focuses on a specific node to demonstrate how the automated analysis can be applied to such case studies.

The properties on the model are used to identify matching vulnerabilities in expert databases from sources such as the National Vulnerability Database(NVD). In this case the properties type "enterprise linux", vendor "redhat" and version "5" of "SP Software System" matches the vulnerability "CVE-2011-1576" from the NVD. The "Vulnerability identification" analysis generates a vulnerability in the cloud model with the description "CVE-2011-1576" and relationship "affecting" the "SP Software System" node. In addition according to the entry data in the NVD, the "CVE-2011-1576" vulnerability is associated to the vulnerability type "CWE-119: Buffer Errors", which is generated in the cloud model as "CWE-119: Buffer Errors" and is composed of the originally identified vulnerability "CVE-2011-1576". The "Malware" threat exploits the "Insecure storage" vulnerability affecting the "Hypervisor" infrastructure node.

After identifying the vulnerabilities and creating instances in the cloud model, the next step is to determine security mechanisms suitable for addressing these vulnerabilities. As an example of our analysis, the security mechanism resolution identifies two security mechanisms; "Patch: RHSA-2011:0927-1" and "Strategy: Language Selection", in order to address the "CVE-2011-1576" and "CWE-119: Buffer Errors" vulnerabilities respectively. These security mechanisms were identified and modelled based on the mitigation details described in the vulnerabilities entries, when referring to the expert knowledge in the NVD.

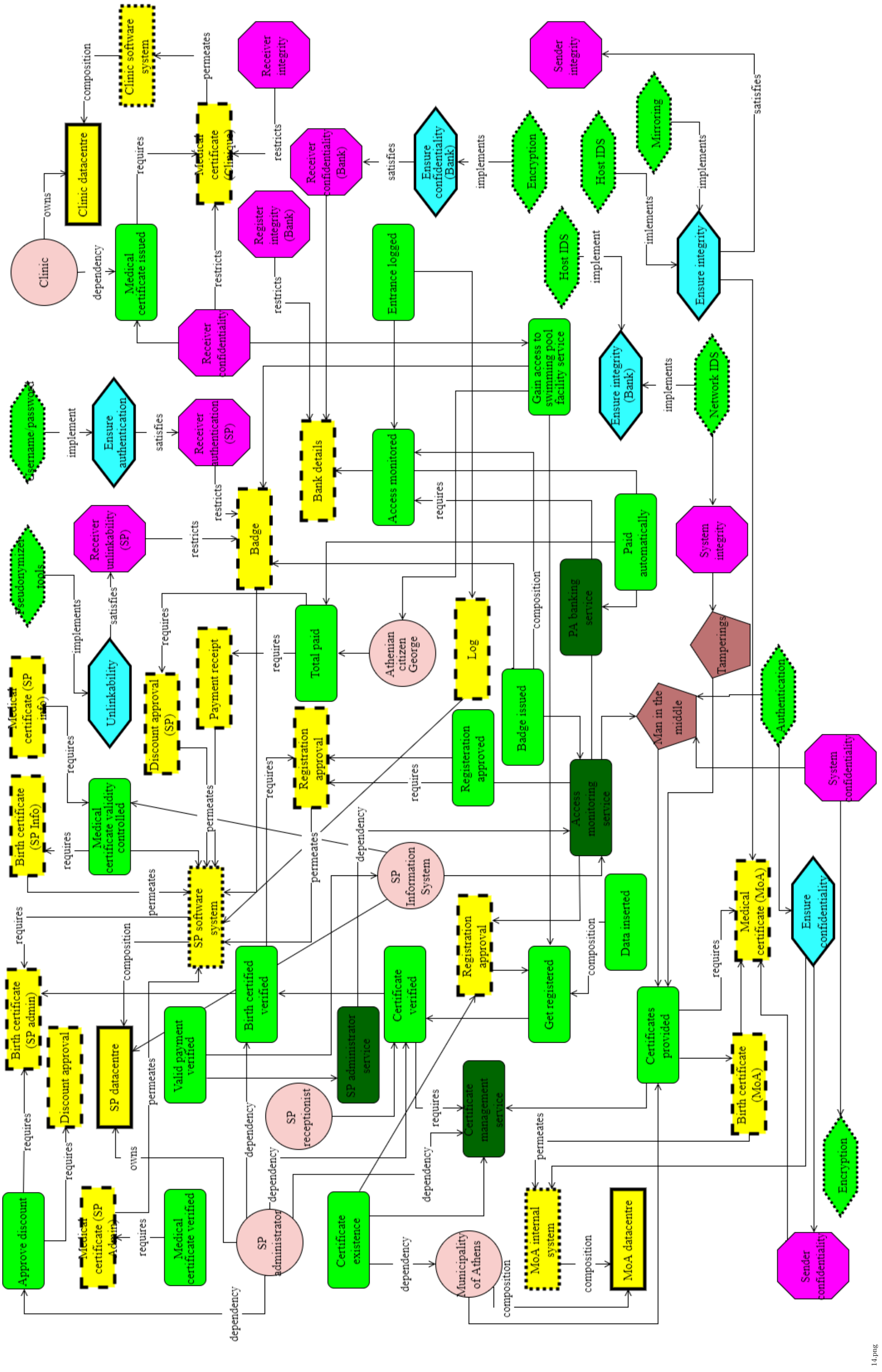


Fig. 14 Visualisation of the holistic cloud model.

5.1.5 Lessons Learned In this sub-section we discuss the lessons learned after the application of our work in the VisiOn case study. During the collaborative work with the requirement engineers in the application of the tool, we have observed issues in the scalability of models and visual presentation. Specifically the model in question was created from requirements specifying over eighty nodes, where several nodes had a high concentration of incoming and outgoing edges. Nodes with multiple edges would partially obscure or overlap with neighbouring nodes and edges, which resulted in models with sections that were visually difficult to understand. This issue with visual presentation of highly concentrated models was exacerbated in the figures produced in this thesis, primary due to the limitation of presenting readable figures which still fits within an A4 page. However in the models created by requirements engineers during the collaborative phase, the spacing was more relaxed to allow for visualising systems with higher complexity and density. This phenomenon was attributed to the fact that in a practical work space, the practitioner focuses on fragments of a system. For example Figure 15 shows a fragment of the cloud environment model generated for the VisiOn case study, where the spacing is relaxed. As such, they are not bound by the limitation of presenting a visual model which scales down to fit an A4 page.

Therefore from this observation we conclude that from a practitioners perspective, the process of creating and editing models using the SectroCloud tool-support is manageable for systems containing up to eighty nodes. In terms of scalability and the impact on the visual presentation of models, nodes with multiple incoming and outgoing edges are partially or fully obscured when located in areas with a high density of nodes. This is further exacerbated when attempting to present the entirety of the model in one view, in particular when scaling the image to an A4 page. This can be partially avoided by allocating additional work space and providing adequate spacing between each node, at the cost of creating models which are not rendered readable when scaled down to an A4 sized page. Therefore with this limitation in mind, we have proposed as future work to provide the functionality of resizable nodes, in order to reshape nodes which has a high number of incoming or outgoing edges, such that edges and labels no longer overlap. A functionality should also be added to support modifications to the anchor points of edges, where the path of edges can be manipulated by the developer.

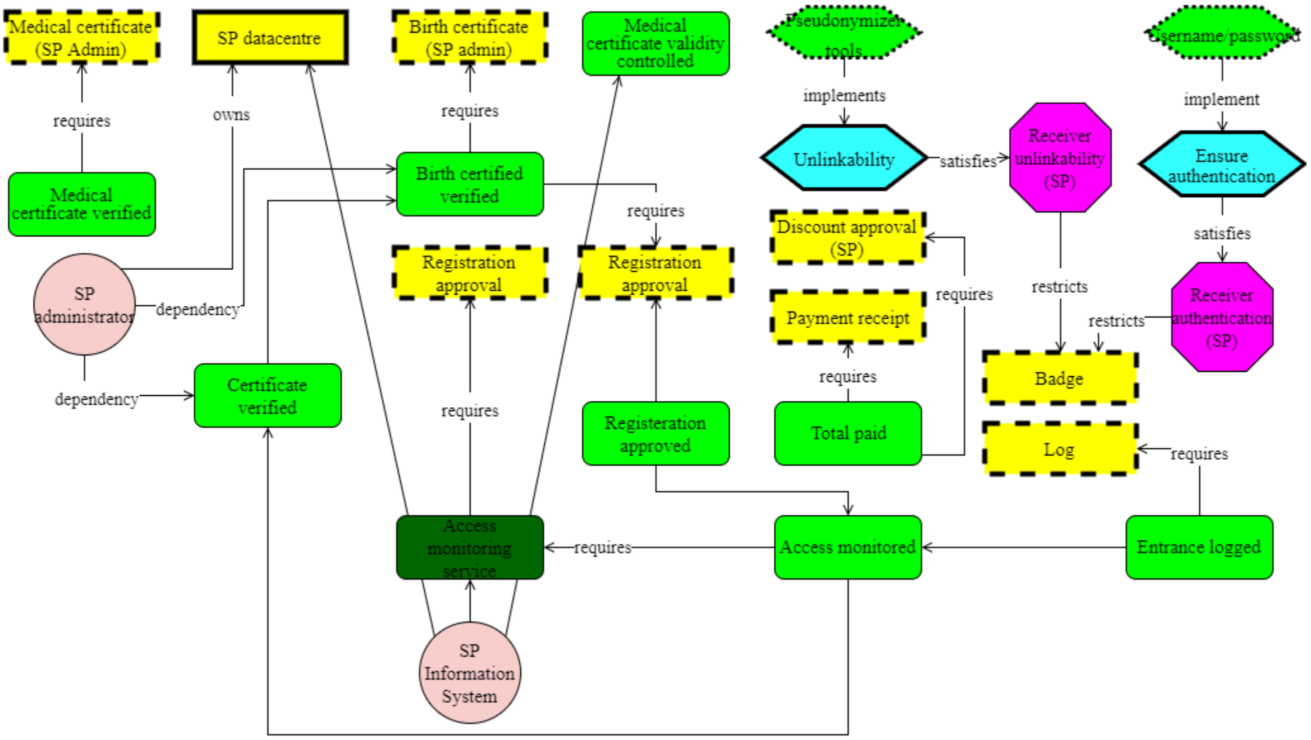
Another observation was regarding the usability of the tool in terms of the graphical notation and user interface. The participants involved in the development process have provided comments on the usability of the tool, noting in comparison to sketching models on papers and boards, the tool-assisted approach was more user friendly. Specifically in one case a participant was involved in a meeting where they were responsible for presenting system requirements to stakeholders. In this case they were able to demonstrate in real time, using the tool, the requirements of the system-under-design during the video conference.

Regarding the usefulness of the framework, our feedback on the ViSion case study, was that the modelling language helps by providing concepts that are easily associated with the cloud computing domain and its properties including virtual resource, physical infrastructure and infrastructure node. The semi-automated analysis of cloud environments was also found useful as it enabled the identification of threats and security vulnerabilities, and validation of security constraints, objectives and mechanisms. Although all these could be manually done for small and very simplistic cloud based systems, the interconnections and size of commonly available cloud based systems makes such analysis difficult to perform. Positive feedback was also received for the secure cloud process indicating that it provides practitioners with a comprehensive approach to the secure cloud environment framework, ensure that they are able to consistently understand and apply fundamental activities throughout the cloud security requirements elicitation process. They also appreciated the fact that during the activities in the process, the developer is made aware of the input and output artefacts, as well as the tasks to be performed during each activity.

The feedback also indicated that an important challenge that developers face when considering security issues in cloud computing is the need to consider multiple perspectives in order to capture the security needs of cloud actors. It was suggested that our framework addresses such challenge by by providing a modelling language with the expressive power needed to describe cloud security needs at a high level of abstraction, without omitting technical details. This is supported by the three separate views of a cloud model, representing refinements at the organisational, application and infrastructure level. Each view allows developers to focus on the concepts and relationships represented from a specific perspective, culminating in a holistic cloud model with varying levels of granularity.

6 Related Work

There are many works in the state of the art which surveys current work to determine a standard definition for cloud computing [1–4]. Vaquero et al. studies over twenty definitions in order to extract a consensus and the minimum set of essential characteristics [1]. Foster et al. discusses the overlap between the cloud paradigm and existing technologies such as grid computing and distributed systems in [2]. Both [1] and [2] make comparisons between the grid paradigm and cloud computing, noting the similarities and highlighting the differences. For example virtualisation existed before cloud computing, however it enables key cloud computing features such as on-demand resource pooling and security



15.png

Fig. 15 Fragment of the model in the VisiOn case study.

by isolation. Qian et al. provides a reference architecture of the cloud through a core stack and the management, where the core stack consists of three layers; resource, platform and application [3]. Qian et al. also provide a classification of cloud computing using two categories; service boundary and service type. They relate the context of each category to the target user group, such as enterprises, developers and external parties. Moreno-Vozmediano et al. presents the concept of services available on the Internet in the Internet of Services (IoS) model [5]. In their view some of the challenges encountered in cloud computing are based on areas that are already defined in the existing literature, such as virtualisation, grid computing and automated computing. Zhang et al. divides the cloud computing environment into four layers; the hardware/datacenter layer, the infrastructure layer, the platform layer and the application layer [6]. Zhang et al. examine the challenges in cloud data security, outlining the responsibility of infrastructure providers to maintain confidentiality when accessing and transferring data and ensuring the auditability of application security settings and policies. Due to the virtualised nature of the cloud environment, the authors highlight the need for considering security on all four layers of the cloud computing environment.

The literature in cloud security primary focuses on mitigating mechanisms and software solutions at the implementation level, which targets software systems that are already implemented and operational [20]. This approach is counter to our argument that security is a factor that is most effective when integrated as early as possible in the software development life-cycle [17]. In existing requirements engineering approaches we are able to capture high level concepts such as stakeholders, goals and resources [22], and security concepts such as vulnerabilities, threats and security constraints [16]. Secure Tropos [16, 28] is a goal-oriented software engineering methodology extending the Tropos methodology to model security concerns throughout the software system development process, based on the notion of agents and related notions such as actors and goals and focusing on the early analysis and design stages. Mouratidis et al. presents a framework to elicit the security and privacy requirements of software systems and select a cloud service provider based on satisfaction of the requirements. They define a modelling language as part of the process, the language extends from several concepts in the software engineering discipline, such as the i* framework, PriS [29] and Secure Tropos which provides specialization in requirements engineering, security engineering and privacy engineering respectively [30]. However the approaches in [22] and [16] lack the expressive power to support developers in modelling the relationship between their organisational, business and security needs in a cloud computing environment. Specifically existing approaches lack an language expressive enough to model cloud-specific concepts such as multi-tenancy, virtualisation and cloud services in the context of cloud security. Beckers et al. proposes a pattern-based method to elicit cloud security requirements aimed at guiding cloud customers during the process of modelling cloud systems [23]. However their approach is focused on establishing an Information Security Management System (ISMS) according to the ISO 27001 standard, without considering the propagation of users cloud security needs. Li et al. provides a holistic security requirements-eliciting approach towards socio-technical systems [24]. However their work lacks expressive power for capturing cloud-specific properties, which is essential for inferring cloud security issues, impact on resources and mitigation strategies. Review efforts indicates that while most work covers multiple security sub-areas, they only target cloud computing issues in isolation, for example considering security properties in software systems or human factors on a social level but failing to provide direct correlation between different conceptual layers [27,25]. Kalloniatis et al. presents a methodology towards eliciting and analysing security and privacy requirements of software systems and the selection of appropriate cloud deployment models [31]. Their framework provides a modelling language based on Secure Tropos [16], the agent-oriented modelling language i* and the PRiS [29] language, which incorporates concepts from security requirements engineering and cloud engineering through a systematic process. Bandara et al. carries out a comparative evaluation of model-based security patterns to examine the extent of support of constructs provided by security requirements engineering approaches. They cover three main categories of modelling approaches; design, goal-oriented requirements and problem-oriented. Their results suggest that "current approaches to security engineering are, to a large extent, capable of incorporating security analysis patterns" [32]. Zardari et al. argues that the Goal Oriented Requirements Engineering (GORE) approach provides a paradigm which addresses the lack of requirements engineering methodologies applicable in cloud adoption [33]. To capture user requirements at various levels of detail, they decompose goals down to business, core and operational goals. Their proposed lifecycle facilitates the negotiation and alignment of user requirements with cloud provider provisioning, taking into account mismatches, trade-offs and risk management. Their outcome is the selection of the optimal cloud service provider given a set of user requirements. Our approach focuses on security requirements of cloud systems from a developers perspective, while taking into account the security needs of cloud users through constraints. We also consider the impact of cloud-specific characteristics such as virtualisation and multi-tenancy, implicitly modelling these concepts in the system-under-design. This allows developers to assess the system from a cloud perspective, identifying threats unique to cloud systems.

Our proposed approach ensures that the system-under-design incorporates security from the early requirements stage, by providing an expressive modelling language able to capture cloud computing concepts and characteristics. We achieve this by building upon existing work in security requirements engineering that lacks the capability to capture or reason about cloud-specific security issues from a holistic point of view [26,16]. Thus our modelling language captures essential cloud characteristics and security implications in order to progress towards addressing the security problem in cloud computing [9]. Specifically we model cloud characteristics such as multi-tenancy, describe cloud service configurations, identify cloud-specific threats and vulnerabilities, in addition to modelling the impact of attacks on physical and virtual resource within the context of a cloud computing system.

7 Conclusion

Currently there is a lack of domain-specific support for developers during the process of eliciting security needs in organisational and business systems adapting the cloud computing paradigm. Our work seeks to fill this gap by providing a modelling language and a set of analysis techniques for eliciting secure cloud environment needs from a requirements engineering perspective, enabling developers to realise organisational needs from a cloud computing context with consideration to security needs. In particular in our contributions *C1* and *C2* we have defined a modelling language to capture cloud computing concepts, which enables the modelling of essential cloud properties required to describe cloud services. We demonstrate through our work in the ViSiOn case study that the language is able to represent both abstractly and through a fine-grained perspective, the organisational needs and the relationships required for modelling cloud security needs. For *C3* we have defined a concrete and machine-readable syntax, providing a method to semi-automate the process of applying our modelling language following established security requirements engineering concepts. *C4* defines three analysis techniques to identify threats and vulnerabilities in cloud systems, propose mitigation strategies and provide transparency in the responsibilities of cloud actors when managing cloud security requirements. This provides a rigorous approach for developers to model and address cloud security needs, threats and mitigation mechanisms through formal textual and visual notation.

For future work we are investigating methods to align state-of-the-art security knowledge from public vulnerability and security control repositories with our framework. This would enrich models created using our framework and provide developers the option to semi-automate the transformation of cloud security controls into security patterns, thus providing a pattern library for applying security policies and mechanisms from a security requirements perspective.

Security Patterns Currently the SectroCloud module does not support importing security patterns to a cloud environment model. While initial efforts have been taken to identify patterns from several domains in the Cloud Controls Matrix (CCM) provided by the Cloud Security Alliance (CSA), the process of replicating these security patterns using tool support is manual. This involves the creation of security controls as models using concepts from our proposed language, which can be saved as a pattern using the SectroCloud tool support. The bulk of the work is in devising a method for the automated transformation of security patterns from expert databases to our proposed language. Further work is needed to allow the importing and exporting of patterns in the tool support, for example replicating a set of security mechanisms to mitigate a specific vulnerability and generating the pattern through our concepts in an existing model. Therefore the automated transformation of cloud security controls into security patterns through tool support will provide practitioners with an extendable library of security patterns, consisting of industry standard solutions.

Scalability and Usability of SectroCloud The focus of the SectroCloud module was to provide semi-automated tool support for practitioners, allowing the visual modelling of cloud computing systems using our cloud security concepts. Further investigation of the scalability of the visual component can improve the usability of the tool, as the effect of the complexity of visual models on the decision making process has not yet been studied within the scope of this work. Two features have been proposed to address the visual clutter of complex models; *(i)* enabling the manual resizing and scaling of nodes, *(ii)* allowing the addition of anchor points on edges. Supporting the resizing of nodes would allow developers to manually adjust the scale of each node in the model, in order to reduce the amount of overlap over edges and labels in complex models. The support for manual manipulation of edges in the model would allow developers to reshape edges which overlap, further reducing visual clutter. These two proposed features is aimed at improving the usability of the tool-support, while addressing the visual component of the scalability issue.

References

1. Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2008). A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), 50–55.
2. Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008, November). Cloud computing and grid computing 360–degree compared. In 2008 Grid Computing Environments Workshop (pp. 1–10). Ieee.
3. L. Qian, Z. Luo, Y. Du and L. Guo, "Cloud Computing: An Overview", Lecture Notes in Computer Science, pp. 626–631, 2009.
4. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50–58.
5. Moreno-Vozmediano, R., Montero, R. S., & Llorente, I. M. (2013). Key challenges in cloud computing: Enabling the future internet of services. Internet Computing, IEEE, 17(4), 18–25.
6. Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state–of–the–art and research challenges. Journal of internet services and applications, 1(1), 7–18.
7. Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.
8. Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. Journal of network and computer applications, 34(1), 1–11.
9. Almorsy, M., Grundy, J., & Müller, I. (2010, November). An analysis of the cloud computing security problem. In Proceedings of APSEC 2010 Cloud Workshop.
10. Lombardi, Flavio, and Roberto Di Pietro. "Secure virtualization for cloud computing." Journal of Network and Computer Applications 34.4 (2011): 1113–1122.
11. Dubois E., Mouratidis H., Guest editorial: security requirements engineering: past, present and future. Requir. Eng. 15(1): 1-5 (2010).
12. Luo, Shengmei, et al. "Virtualization security for cloud computing service." Cloud and Service Computing (CSC), 2011 International Conference on. IEEE, 2011.
13. Ristenpart, T., Tromer, E., Shacham, H., & Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. Proc. 16th ACM conference on Computer and communications security (pp. 199–212). ACM.
14. Shei, S., Mouratidis, H., & Delaney, A. (2017). A Security Requirements Modelling Language to Secure Cloud Computing Environments. In Enterprise, Business-Process and Information Systems Modeling (pp. 337–345). Springer, Cham.
15. Shei, S. (2018). A model-driven approach towards designing and analysing secure systems for multi-clouds, PhD Thesis, University of Brighton, UK.
16. Mouratidis, H., & Giorgini, P. (2007). Secure tropos: a security-oriented extension of the tropos methodology. International Journal of Software Engineering and Knowledge Engineering, 17(02), 285–309.
17. Kissel, R., Stine, K., Scholl, M., Rossman, H., Fahlsing, J., & Gulick, J. (2008). Security considerations in the system development lifecycle (NIST 800–64 rev. 2)
18. Argyropoulos, N., Shei, S., Kalloniatis, C., Mouratidis, H., et al. (2017). A Semi-Automatic Approach for Eliciting Cloud Security and Privacy Requirements. Proc. 50th Hawaii International Conference on System Sciences.
19. Mavropoulos O., Mouratidis H., Fish A., Panaousis E. (2019). Apparatus A framework for security analysis in internet of things systems. Ad Hoc Networks 92.
20. Modi, C., Patel, D., Borisaniya, B., Patel, A., & Rajarajan, M. (2013). A survey on security issues and solutions at different layers of Cloud computing. The Journal of Supercomputing, 63(2), 561–592.
21. National Vulnerability Database. http://nvd.nist.gov/.
22. Eric, S. Yu. "Social Modeling and i*." Conceptual Modeling: Foundations and Applications. Springer Berlin Heidelberg, 2009. 99-121.

23. Beckers, K., Côté, I., Faßbender, S., Heisel, M., & Hofbauer, S. (2013). A pattern-based method for establishing a cloud-specific information security management system. Requirements Engineering, 18(4), 343–395.
24. Li, T., Horkoff, J., Beckers, K., Paja, E., & Mylopoulos, J.(2015). A holistic approach to security attack modeling and analysis. Proc. of the Eighth International i* Workshop(2015).
25. Iankoulova, I., & Daneva, M. (2012, May). Cloud computing security requirements: A systematic review. In Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on (pp. 1–7). IEEE.
26. Fabian, B., Gürses, S., Heisel, M., Santen, T., & Schmidt, H.(2010). A comparison of security requirements engineering methods. RE, 15(1), 7–40.
27. Sengupta, S., Kaulgud, V., & Sharma, V. S. (2011, July). Cloud computing security–trends and research directions. In Services (SERVICES), 2011 IEEE World Congress on (pp. 524–531). IEEE.
28. Mouratidis, Haralambos. "Secure Tropos: An agent oriented software engineering methodology for the development of health and social care information systems." International Journal of Computer Science and Security 3.3 (2009): 241–271.
29. C. Kalloniatis, E. Kavakli and S. Gritzalis, "Addressing privacy requirements in system design: the PriS method", Requirements Engineering, vol. 13, no. 3, pp. 241–255, 2008.
30. Mouratidis, H., Islam S., Kalloniatis C., Gritzalis S.: A framework to support selection of cloud providers based on security and privacy requirements. Journal of Systems and Software 86(9), 2276–2293 (2013)
31. Kalloniatis, C., Mouratidis, H., & Islam, S. (2013). Evaluating cloud deployment scenarios based on security and privacy requirements. Requirements Engineering, 18(4), 299–319.
32. Bandara, Arosha, Shinpei, H., Jurjens, J., Kaiya, H., Kubo, A., Laney, R., Mouratidis, H., et al.: Security patterns: comparing modeling approaches. In: Software Engineering for Secure Systems: Industrial and Research Perspectives: Industrial and Research Perspectives (2010): 75.
33. Zardari, S., & Bahsoon, R. (2011, May). Cloud adoption: a goal-oriented requirements engineering approach. In Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing (pp. 29-35). ACM.