A Multi-Layer and Multi-Tenant Cloud Assurance Evaluation Methodology

Aleksandar Hudic,
Markus Tauber,
Thomas Lorünser
AIT
{aleksandar.hudic,
markus.tauber}@ait.ac.at

Maria Krotsiani,
George Spanoudakis
City University London
{g.e.spanoudakis,
maria.krotsiani.1} @city.ac.uk

Mauthe, Andreas
Lancaster University
a.mauthe@lancaster.ac.uk

Edgar R. Weippl
SBA Research
EWeippl@sba-research.org

***Abstract*—Data with high security requirements is being processed and stored with increasing frequency in the Cloud. To guarantee that the data is being dealt in a secure manner we investigate the applicability of Assurance methodologies. In a typical Cloud environment the setup of multiple layers and different stakeholders determines security properties of individual components that are used to compose Cloud applications. We present a methodology adapted from Common Criteria for aggregating information reflecting the security properties of individual constituent components of Cloud applications. This aggregated information is used to categorise overall application security in terms of Assurance Levels and to provide a continuous assurance level evaluation. It gives the service owner an overview of the security of his service, without requiring detailed manual analyses of log files.**

Keywords — critical infrastructures, assurance, cloud

I. INTRODUCTION

An important transformation process in IT systems is currently taking place triggered by the rapid propagation of the Cloud Computing paradigm across distinct domains and organisations. Hence it is envisaged that ICT services will in future be delivered in a manner similar to utilities such as water, electricity, gas, and telephony. The main motivation for adopting Cloud technology is to increase efficiency and minimize IT costs by offering new concepts such as elasticity, scalability and on-demand resource provisioning. However, in order to automatically provision resources for elastically adaptive Cloud applications it requires both, the applications and the underlying platform to be constantly monitored to capture information at various system and operational levels and time intervals. This is particularly manifested in Critical Infrastructures, which require even more attention when these systems are hosted on top of Cloud environments.

However, the use of Cloud computing has introduced new risks that have to be sufficiently understood before an organisation should consider adopting the Cloud and using Cloud services. Moreover, due to the complexity of the application execution environment, routine tasks such as monitoring or security analysis becomes quite complex. These tasks often require close interaction and assessment between different layers of the Cloud stack. For example, certain distributed applications running within a Cloud cluster on specific virtual machine(s) (VM) require a general assurance, or even have to be certified, for maintaining specific security properties. This might also require monitoring the execution of the application on the VMs, as well as monitoring the availability of the physical resources of the VMs. Thus, this would require the use of different tool sets to collect and analyse the performance of data from each level in order to reach the point where the application can be certified.

Under these circumstances, we should gather different types of information at various levels of granularity, from low-level system metrics (e.g. CPU usage, network traffic, memory allocation, etc.) to high-level application specific metrics (e.g. throughput, latency, availability, etc.). These are collected across multiple system layers (physical, virtualization, application level) in a Cloud environment at different time intervals. Hence, the challenge in this case is to define a way to aggregate these different types of information from different levels in order to provide an overall assurance, and determine how changes in individual assurance levels of every component affects the overall assurance.

In this paper, we propose, based on existing work[10] an assurance method. We refer to assurance, motivated by common criteria, as the likelihood for a service falling victim to a cyber-attack. A high assurance level means a low probability for this to happen. Security properties, based on measurable metrics, of substituent components contribute to the overall assurance level and how they are aggregated is subject to dependency policies. This is based on a comprehensive concept for assessing security properties across multiple layers with different stakeholders for composite based systems. The dependency policies, can be flexibly adopted according to various use case requirements to derive evaluation of every individual component of a service or a system.

The rest of the paper is structured as follows. Section II outlines the related work. Section III describes our approach and introduces the Assurance Assessment Method, the way we define assurance levels, how we abstract the service as a general tree, and the assurance aggregation process. In Section IV the evaluation of the approach is provided based on a Use Case Scenario. Finally, section V provides concluding remarks and directions for future work.

II. RELATED WORK

Traditional approaches for assurance assessment in the Cloud, such as Cloud Security Alliance (CSA) [12], Information Technology Assurance Framework (ITAF) [17], or the Cloud Computing Information Assurance Framework from ENISA [18], are usually built on existing frameworks such as ISO/IEC 27000-series (e.g. current work in progress ISO/IEC 27017 and ISO/IEC 27018 which are focusing on information security and data protection in Cloud), PCI DSS Cloud Guideline [13], COBIT [14], NIST [16], or IT Baseline Protection Catalogues [15].

We have considered existing approaches, namely the Common Criteria framework [6] for assurance of IT systems (as it is the most dominant work in the field) and extends it [10] since its main focus is on assessing assurance in the development phase of the life cycle but lacks support in the subsequent production phase.

Unlike traditional approaches, the work derived from Krotsiani et.al. [11] proposes a novel approach for certifying the security of Cloud services based on incremental certification of security properties for different types of Cloud services (including IaaS, PaaS and SaaS services). This approach uses operational evidence from the services provisioning through continuous monitoring. Although the model does not directly address assurance as an explicit objective, it can be adopted to efficiently assess assurance at various levels and time intervals.

Our approach is related to autonomic monitoring systems that are based on the SECCRIT architecture model [7] and on an evidence-gathering model for assurance assessment in critical infrastructures hosted on top of Cloud environments (as introduced in [8]). Moreover, we found the concepts of Common Criteria for analysing and assessing application in preproduction phases. However, we emphasize the importance of observing systems in their production phase, as well as their dependencies with other corresponding elements inside of heterogeneous systems[1]

III. Multi Layer Assurance Assesment Model

. The popular National Institute of Standards and Technology (NIST) [3] model depicts the Cloud architecture through a dynamic tree-layered service-provisioning model (infrastructure, platform and software - as a Service layer) capable of scaling services across distinct administrative and legislative domains. However, the common practices for provisioning and delivering services (as well as the abstraction of those layers and driven technologies) differentiates based on the business objectives of a particular Cloud provider. Hence, the traditional assessment frameworks (e.g. COBIT, ISO 27000 series) are not fully applicable, especially when addressing security related concerns in Cloud environments (as discussed in [10]).

However, in order to build a comprehensive and flexible framework that is able to acquire heterogeneous information across the Cloud stack the following objectives have to be addressed:

- cross layer assessment
- technology independence
- information acquisition restrictions
- assessment, quantification and aggregation of different information sets

[1] It should be noted that our work is a part of a broader research programme, undertaken by the EU F7 project SECCRIT [4]

The assessment of such services when taking into account different Cloud layers requires a compact solution, able to embrace all requirements and produce an effective assessment tool. Especially when considering different stakeholders, various business and security objectives, a high degree of service complexity, business model, and distinct technologies. Hence, we adopt Common Criteria [6] to address assurance in Cloud related environments. Although, Common Criteria offers a comprehensive solution for assurance assessment, it lacks support for the production phase, especially when referring to those services that are hosted on top of the Cloud architectures. Taking this and the above-mentioned objectives into account, we use the Common Criteria approach in order to address assurance assessment of complex services hosted in Cloud infrastructures. Furthermore, the policies of some Cloud providers restrict information crawling across their Cloud stack (for instance software as a service Cloud provider will hesitate to reveal the information of underlying service being provided, in order to mitigate potential attack vectors on its infrastructure). Hence it is harder to analyse, indicate or predict security issues in such environments. Thus, we distinguish two main categories: a) solutions based on open-source Cloud environments (i.e. solutions where we are able to freely acquire necessary information without restrictions); and b) closed Cloud environments with restricted information access (i.e. public Cloud providers which provide any additional information via the Service Level Agreements (SLA) [21][22]). Due to the flexibility of acquiring the information and ability to modify services for provisioning the information, this paper focuses primarily on open-source Cloud solutions (e.g. OpenStack [23], CloudStack [24]). This does, however, not limit our approach to these environments.

The assessment and aggregation of different information sets (i.e. analysis of a particular entity in the Cloud with respect to a specific set of properties) is derived from the concept of *assurance levels*, supported through *aggregation policies* (i.e. decision making algorithms that cluster the security properties of each class towards the predefined assurance levels), aligned with the Common Criteria approach [6].

A. *Assurance assesment method*

Considering these objectives and building on the research presented in [10] we propose a comprehensive and flexible approach for performing assurance assessment. The approach is using a well-defined set of security properties, provided by the CUMULUS project [5]. These are additionally aligned with the SECCRIT vulnerability catalogue [20] and The Notorious Nine from Cloud Security Alliance [19].

Our assessment method emphasises three core assessment entities: *Target of Evaluation* (ToE), *Group of Evaluation* (GoE) and *Component of Evaluation* (CoE). These entities are aligned with the Common Criteria assessment framework, and are therefore designed to offer flexibility, determination of the precise impact of the individual components or group of components, scalability of assessment across different time intervals, and the possibility to highlight each individual entity of the system as an independent point of evaluation. Furthermore, we designed our method as a hierarchical tree structure defined with parent-child object relationship. Each parent can be in a direct relationship with multiple child objects. The parent object that does not have any related child objects is referred to as *leaf* object. Additionally, we also define *associations*, *dependencies*, *associated component sets* and *assurance profiles*, as supporting assessment elements of the ToE. Figure 1 illustrates the fundamental elements of our Assurance Assessment Method. More specifically, it presents how a particular service can be abstracted through a set of hierarchically organized components. We use these abstraction elements to build our method and to efficiently assess assurance according to a predefined set of security properties derived from the CUMULUS project.

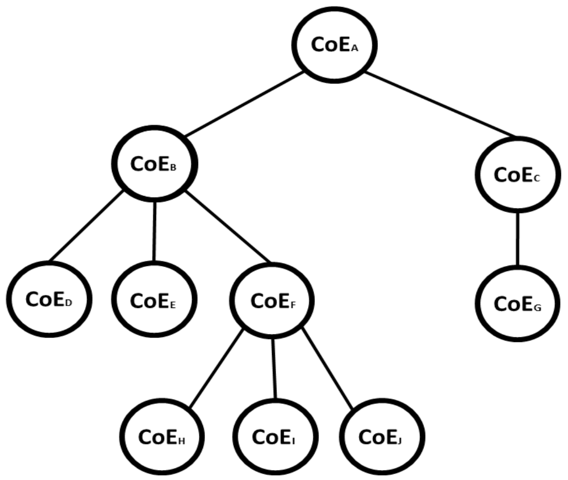

Figure 1: Hierarchical illustration of services via the general tree model structure. The service or application is defined as a Target of Evaluation (ToE) depicted with the individual Components of Evaluation (CoE), whereby each individual CoE can be associated with N distinct CoEs, referred as Associated Component Set (ACS). The correlation between two individual CoEs is referred to as a Component Dependency, which is a formal compound of Association. Moreover, CoEs are grouped in order to establish assurance of components with respect to specifc security classes, these groups are then formally defined as Groups of Evaluation (GoE).

The initial step of the assessment method defines and details the ToE. This can be either an asset of the Cloud referred to as service (e.g. a specific service operation, a set of service operations, data managed by the service) or an asset that is required or contributes to the realization of a Cloud service (e.g., a virtual machine).

Moreover, each ToE contains a set of attributes such as: (i) security objectives, which are mapped towards the related set of security claims and are formally referred to as *Security Properties* (SP); (ii) attributes that define the type of assurance (e.g. information or system assurance) according to the assurance model presented in [10]; (iii) a short description of the ToE; and (iv) the assessment interval. The security objectives are the statements of intent to counter the identified threats by IT measures. Each ToE can be formally defined as $ToE \equiv T = \{COE_i, i \in \mathbf{N}\} \mid \{GOE_i, i \in \mathbf{N}\}$. This generalized statement as presented in Figure 1 can be formulated as $ToE \equiv COE_A = \{COE_i, i \in \langle B, C, D, E, F, G, H, I, J\rangle\}$. The group of objects, formally referred as Group of Evaluation (GoE) and defined as $GoE = \{CoEi, i \in N\}$, are a compound set of individual objects that share common properties based on which the assessment is conducted. Considering Figure 1, GoE can be formulated as compound of objects, e.g. $GOE_1 = \{COE_i, i \in \langle F, H, I, J\rangle\}$. Each individual object to which we refer to as the component of evaluation (CoE) can be also handled as an independent ToE. Each GoE is composed of (i) attributes, used for describing a particular group; (ii) assurance profile, which is the essential element for evaluation; (iii) associations, an element used to describe relationships between different groups in the scope of the evaluated target; and (iv) individual components.

Component Dependency (CD) is a correlation between two individual components of the evaluated system (i.e where CD_{ij} $\{\langle COE_i , COE_j\rangle, i, j \in N\}$), that arises when a component is not self-sufficient and relies upon the presence of another component, e.g. when referring to Figure 1 $CD_{CG}=\{COE_C, COE_G\}$. *Association* is a set of two individual components that are in a direct parent-child relationship with a defined dependency, for which it is valid: $\forall\ AS_i\ i \in \mathbf{N} \equiv\ !\exists\ CD_{ij}\ \{\langle COE_i , COE_j\rangle, i, j \in N\} \implies COE_i$ parent of COE_j. An individual parent object can be associated with N distinct child objects, which we formally refer to as *Associated Component Set* (ACS), for which the following statements are valid: $ACS_K \coloneqq ACS\ (COE_K) = \{COE_i, i \in \mathbf{N}\}, \forall\ COE_i \implies\ !\exists$ Parent $= COE_K$ and $\nexists\ CD_{ij}\ \{\langle COE_i , COE_j\rangle, i, j \in N\}$.

Last but not least, the *Assurance profile* (AP), an essential element in our method used to define policy related with security properties that are mapped to the Assurance classes (AC) of a particular CoE or GoE. These security properties will at the end define the level of assurance for an individual component, group or even a whole system. We emphasize two types of Assurance profiles setup: Uniform Assurance Profile (AP_U), which is always the same regardless of class, evaluated object, group or target; and Custom Assurance Profile (AP_P), which can be customised depending on the object of appliance. In Table 1 we illustrated the AP_U for a particular assurance class. Furthermore, we can also assign a custom Assurance Profile to a particular CoE, GoE or ToE.

B. *Assurance Levels*

Assurance levels (AL) outline the scale of measurement for evaluating predefined ToE, GoE or CoE. Every individual CoE or GoE contributes directly to the assurance level of the ToE by meeting a set of SPs (i.e. a certain set of security criteria). Moreover, the SPs derive the AL per individual AC by also taking into consideration the dependencies of the evaluated object, e.g. component, group or target of evaluation if such are present. However, each AC may contain *k* of SP (*k* number of SPs) as shown in equations (5) and (6). Due to the binary decision making concept applied in our approach there can be 2^k combinations of distinct SP states where $2^k > N$, and N is the cardinality of AL, in terms of security properties (AL=

{1, 2, 3, 4 … N}). Thus, each individual combinations of SPs {SP_1, SP $_2$, SP_3, SP_4 … SP_N}, associated with a particular AC, are formally referred to as Security Property Vector (SPV) (equations (3, (4, (5, (6). Security Property Vector defines the current state of an object by identifying particular set of security properties. Each SPV, is associated with a particular assurance class, whereby each class can comprise multiple SPVs. Thus, in order to scale 2^k states over N assurance levels, we encode ranges in hexadecimal vectors that cluster a potential set of states that correspond to a particular SPV, as shown in Table 1. Hence, each individual AL is assigned with multiple SPVs, which are formally referred as Vector Set (VS), (equation (2)).

Table 1 presents an example of Assurance profile for a particular Assurance class. More specifically, it illustrates a set of relevant SPs clustered per individual AC_K represented with a hexadecimal vector. The left hand side of the table shows the SPVs, sorted by relevance, and all potential combinations for a particular security vector SPV = [SP_4, SP_3, SP_2, SP_1]. The right hand side shows a binary vector for AL_i ($i \in \{1, 2, 3 \dots 7\}$), which associates particular set of SV vectors. At the bottom of the table the Hexadecimal representation of each particular binary AL vector is illustrated.

Table 1 Assurance level association for a particular assurance class. Set of relevant SPVs clustered per individual AC_K represented with a hexadecimal vector. The left hand side of the table shows the SPVs, sorted per relevance, and all potential appearance combinations for a particular vector SPV = [SP_4, SP_3, SP_2, SP_1]. The right hand side shows a binary vector for AL_i, i ∈ {1, … 7}, which associates particular set of SV vectors. At the bottom of the table the Hexadecimal representation of each particular binary AL vector is illustrated.

Security Property Vector (SPV)				**Assurance level association**							
SP_4	**SP_3**	**SP_2**	**SP_1**	**AL**	**AL_1**	**AL_2**	**AL_3**	**AL_4**	**AL_5**	**AL_6**	**AL_7**
0	0	0	0	-	0	0	0	0	0	0	0
0	0	0	1	**AL_1**	**1**	0	0	0	0	0	0
0	0	1	0	**AL_2**	0	**1**	0	0	0	0	0
0	0	1	1	**AL_2**	0	**1**	0	0	0	0	0
0	1	0	0	**AL_3**	0	0	**1**	0	0	0	0
0	1	0	1	**AL_3**	0	0	**1**	0	0	0	0
0	1	1	0	**AL_4**	0	0	0	**1**	0	0	0
0	1	1	1	**AL_4**	0	0	0	**1**	0	0	0
1	0	0	0	**AL_5**	0	0	0	0	**1**	0	0
1	0	0	1	**AL_5**	0	0	0	0	**1**	0	0
1	0	1	0	**AL_6**	0	0	0	0	0	**1**	0
1	0	1	1	**AL_6**	0	0	0	0	0	**1**	0
1	1	0	0	**AL_7**	0	0	0	0	0	0	**1**
1	1	0	1	**AL_7**	0	0	0	0	0	0	**1**
1	1	1	0	**AL_7**	0	0	0	0	0	0	**1**
1	1	1	1	**AL_7**	0	0	0	0	0	0	1
Hexadecimal AL vector					0002	000C	0030	00C0	0300	0C00	7000

$$\forall AL_K \in AC_X: \; !\exists \; VS, \qquad \textbf{(1)}$$

$$VS = \{SPV_1, SPV_2 \dots SPV_N\}, \qquad \textbf{(2)}$$

$$SPV\textit{i} = [\, SP_1, SP_2, SP_3, SP_4], \; SP_i = \{0,1\} \qquad \textbf{(3)}$$

$$\forall VS \in \; AL_K: \; \exists \; SPV\textit{i}, \; i \in \mathbb{N} \qquad \textbf{(4)}$$

$$\forall \; SPV\textit{i} \in AC_X: |SPV\textit{i}| = k \qquad \textbf{(5)}$$

$$AC_X = \{SPV_1, SPV_2, SPV_3, \dots SPV_n\} \qquad \textbf{(6)}$$

$$\bigcap_{i=1}^{i=n} AC_i = \emptyset \qquad \textbf{(7)}$$

For each individual AC that is associated with a set of SPVs particular SP (part of SPV) may vary. Nevertheless, every individual AC, regardless of the SPs, always has to have the same cardinality k (equation (5)). In order to efficiently aggregate the assurance across the variety of architectural layers, ACs first has to fulfil the equations (5) and (6), stating that regardless of the AC, none of the SPs can be associated with more than one AC (equation (7)).

Although, we abstract ALs over N levels, for the purpose of our empirical evaluation we will conduct the assessment over 7 ALs, therefore having minimum 3 SP per AC to be able to map all assurance levels with SPVs. Depending on the property set that a particular entity (i.e. class component, group or even a whole target of evaluation) is assigned with and due to the dynamic behaviour of the Cloud the AL will also be dynamic and vary. Hence, it is crucial to efficiently assess the assurance in a continuous manner without impacting on the performance of the evaluated service or collocated services.

C. *Assurance Aggregation*

As mentioned above, we propose a concept for the assurance aggregation through a recursive process, which aggregates the individual assurance levels of the underlying associated objects (i.e. it calculates the overall assurance of the components that are associated with the root component). The overall assurance can be derived by applying the method depicted in Figure 1. Further, by conducting the proposed algorithm described in Figure 4 we can then derive the overall assurance. Therefore by referring to Figure 1, we state the CoE_A as the ToE. Since, the CoE_A is associated with two additional components, CoE_B and CoE_C, which represent the associated components set (ACS_A) of the CoE_A and are additionally connected with other components. The overall assurance in this case has to be recursively aggregated from the

$$ACS_{AL} = \bigcap AC_X (SPV_i) \, , \;\; AC_X \in CoE_M, \; i \in \{1 \dots N\} \qquad \textbf{(8)}$$

$$ACS_{AL}(i) \vdash DAL_{VS}(i) \qquad \textbf{(9)}$$

$$AL_{VS} \subseteq \; DAL_{VS} \qquad \textbf{(10)}$$

$$(DAL_{VS}(i) \wedge AL_{VS}(i)) \Rightarrow AL(AC_X) = i, \; AC_X \in CoE_M \qquad \textbf{(11)}$$

$$!\exists \; AL\textit{i} \vDash \; \forall Min(CAL\textit{j}) \;\; i \in \{1 \dots 7\}, j \in \{1 \dots N\} \qquad \textbf{(12)}$$

leafs of the tree (i.e. by aggregating all ACS (ACS_B, ACS_C and ACS_F). Therefore we will use tree traversal post order method to iteratively walk through the tree. For the first use case, we just refer to the concept of the tree traversal post order method as a tool for our concept. This method is slightly extended by integrating our Assurance Level Calculation Procedure (ALCP) from Figure 3 using recursively aggregate assurance.

The assurance level of the referenced ACS (ACS_F, ACS_B and ACS_C, respectively), by applying the ALCP aligned with the equation (8). The procedure sequentially conducts bitwise conjunction of individual SPs for each CoE across each ACS. Depending on the result of conjunction (1 or 0) it is decided if all SPVs are discarded with the bit that matches the result of the conjunction. For example, by discarding certain SPVs we are indirectly discarding those ALs that are not fulfilling the

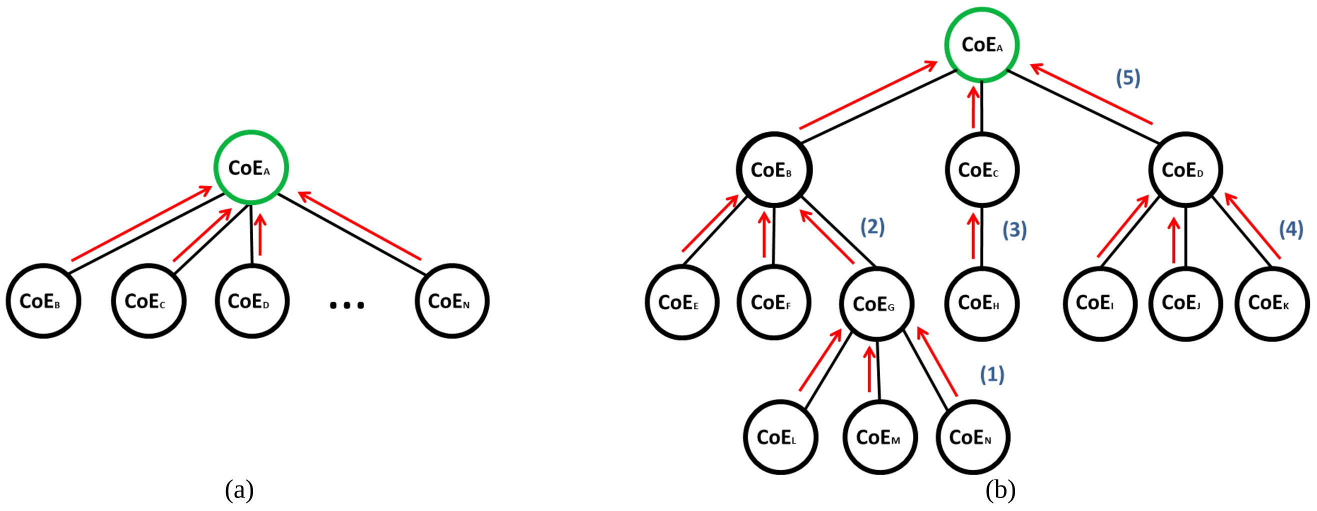

Figure 2: Evaluation use cases derived with respect to the SECCRIT [4] case studies. The subfigure (a) illustrates the basic model of a general tree where the depth of the tree is one and the degree is N. This is an initial model where the algorithm introduced in Figure 3calculates iteratively the conjunction of SPV bits to determine the overall assurance of the leaves of ACS for CoE$_i$, $i \in$ {B,C,D…N} and aggregates towards root according the policies defined in Table 2 an equations (8, (9, (10, (11, (12). Although this is straight forward, in subfigure (b), the same process is aligned with the post-order traversal method which at the end aggregates the Assurance towards the root COE$_A$.

current set of SPs for particular ACS. The next step is to map the suitable ACS$_{AL}$, according to the Table 2, towards the appropriate DAL$_{VS}$. The DAL$_{VS}$ is not only used for mapping the calculated ACS$_{AL}$, but also to customize the underlying security properties of a particular AL. Finally, we calculate the AL of the root CoE for a particular ACS, equation (9), depending on the SPs that the CoE corresponds to the AL of the ACS whereby the equations (10) and have to be fulfilled. However, in case of multiple ACs per CoE we have to consider equation (12) where we consider the AL of individual ACs to determine consolidated ALs for a CoE.

Table 2: Assurance Level per distinct Assurance classes depicted with Hexadecimal vectors. We define minimal assurance level requirements (DAL$_{VS}$) of the objects that are in direct relationship with the parent object. It also defines the assurance level requirements per level of the parent object itself, AL$_{VS}$. Additionally we define the minimum requirement for each AC in terms of AL, i.e. we define at which assurance level individual AC has to satisfy to define the overall assurance of the object. In case when we have multiple AC to consider in order to derive the overall AL we use the Consolidated Assurance Level (CAL).

ASSURANCE LEVEL		I	II	III	IV	V	VI	VII	N
AC$_1$	**AL$_{VS}$**	0002	0008	0010	0080	0C00	7000	8000	8000
	DAL$_{VS}$	0002	0004	0030	00C0	0D00	3000	C000	8000
	CAL	-	AL$_1$	AL$_2$	AL$_3$	AL$_5$	AL$_6$	AL$_7$	AL$_N$
AC$_2$	**AL$_{VS}$**	0002	0008	0020	0040	0300	1800	4000	8000
	DAL$_{VS}$	0004	0018	0020	00C0	0300	1C00	6000	8000
	CAL	-	AL$_1$	AL$_3$	AL$_4$	AL$_5$	AL$_6$	AL$_7$	AL$_N$
AC$_3$	**AL$_{VS}$**	0002	000C	0010	00C0	0200	0C00	4000	8000
	DAL$_{VS}$	0006	0004	0030	01C0	0200	1C00	6000	8000
	CAL	AL$_1$	AL$_2$	AL$_3$	AL$_4$	AL$_5$	AL$_5$	AL$_6$	AL$_N$
AC$_N$	**AL$_{VS}$**	0006	000C	0030	00C0	0D00	1C00	7000	8000
	DAL$_{VS}$	0006	000C	0030	00C0	0D00	1C00	7000	8000
	CAL	AL$_1$	AL$_2$	AL$_3$	AL$_4$	AL$_5$	AL$_5$	AL$_7$	AL$_N$

D. General Tree Model

A general tree G is a finite compound set of nodes such that there is only one designated node R, referred as root of the tree G, where each individual node has only one ancestor (Parent) node, with exception of the root, and multiple successors (Children). Each node of the tree is defined by two properties: Depth and Degree. Depth of the node is the distance of the node from the root node, and Degree of the node is the number of successors for a particular node. Moreover, each general tree can be partitioned in n > 0 disjoint subsets T_0, T_1, T_2 … T_{n-1}, where each is a tree whose roots R_0, R_1, R_2 … R_{n-1} are children of the tree G. The subset Ti ($0 \leq i \geq n$) is a subset of the trees of T.

Although we intent to depict our services through a general based tree model, they can be also depicted via the binary tree model, since the general tree model is easily transformed to A binary tree. For demonstrational purpose of our algorithm (Figure 3) we will use the general tree model. Since the model can be easily transformed, our implementation can be adopted to apply the algorithm on binary trees as well. However, we will not address the assessment of binary trees as it exceeds the scope of this work.

IV. Evaluation

The introduced approach is evaluated and explained in more details using two scenarios. As first step, the cyber-risks that exist in the use case scenario have to be understood alongside the security properties that need to be assessed and certified.

Perceptions of risk in the context of Cloud computing have to be well understood since they will inevitably influence decisions about the adoption of Clouds or the security controls that will be applied to them. Two important factors that must be taken into consideration for a better understanding of cyber-security risks are: (i) the threats and their likelihood to occur; and (ii) the vulnerabilities and an indication of their severity. A key challenge when understanding the risks associated with Cloud computing is to determine those that are specific to the use of Clouds.

Therefore, in order to comprehend the Cloud-specific risks of our scenario we use the Cloud vulnerability catalogue the SECCRIT project [4] has developed, in which we then mapped the Notorious Nine Top threats from CSA [1]. Further, with the help of the CUMULUS project's security property catalogue [2], we map these vulnerabilities to possible security properties for their assessment. The basis of this catalogue is the identification of a number of categories that enable us to focus directly on Cloud-related issues. The core of these categories is based on the NIST essential Cloud computing characteristics [3].

```
begin procedure:
  for i=k … i=1 do
    if (∀ $CoE_C$ (SPV[i]) ∃! $AL_{M,}$ M ∈ {1,2,…,7}) {
       AL = M;
       end procedure
    }
    else if ($\prod_{i=1}^{i=n} CoE_i(SPV[i])$ == 0) {
       discard ∀ SPV where SPV[i] =1;
       continue;
  }
    else ($\prod_{i=1}^{i=n} CoE_i(SPV[i])$ == 1) {
       discard ∀ SPV where SPV[i] =0;
       continue;
    }
end procedure
```

Figure 3: *Assurance level calculation procedure* (ALCP) for associated objects used in equation (8). The procedure does the bitwise conjunction of the most significant bit and based on the result decides whether to discard the SPV that have 0 or 1 assigned to a particular bit that is being analysed. Furthermore, during each iteration, the procedure checks if the remaining vectors that define a particular component are a subset of one of the vector sets associated to a particular AL_i, as shown in Table 1, for a particular AC_k

A. *Use Caseses*

The aim of the evaluation is to illustrate a real world scenario via the abstraction of a general tree model. This is used to assure the public safety of critical infrastructure services and assesses the assurance according to a set of security classes/properties. We refer in particular to the case studies from the SECCRIT project [4] in order to abstract our approach and make a proof of concept assessment algorithm.

To demonstrate our algorithm we abstract a service via the use case scenarios explained below. Moreover, we implemented our assurance algorithm in Java so we can randomly define properties of evaluation such as depth of a tree, degree of a node, security property vector bit length. Furthermore, our implementation method is founded on post-order tree traversal model in order to efficiently evaluate the assurance of service by traversing the tree to aggregate security in respect to assurance policies.

For the first use case scenario, the Depth (D_1) of the tree T_1 is 1, meaning that we have only a root with a set of children, Degree (D_2) will be N, generated randomly, as shown in (Figure 2 (a)). In the second use case both degree and depth properties are predefined, e.g. D_1=3 and D_2=3 (Figure 2 (b)). Within the second use case we want to demonstrate the application of our algorithm in a more complex general tree, which would illustrate the service more realistically.

B. *Security Properties, Vulnerabilities and Threats*

The SECCRIT case studies consider mainly risks related to the authorisation of users, data storage and data leakage. In Figure 4 we present the architecture of the system with components in different levels, as well as their dependencies. Moreover, some relevant security properties are mapped to each component that needs to be certified in order to assure the whole service.

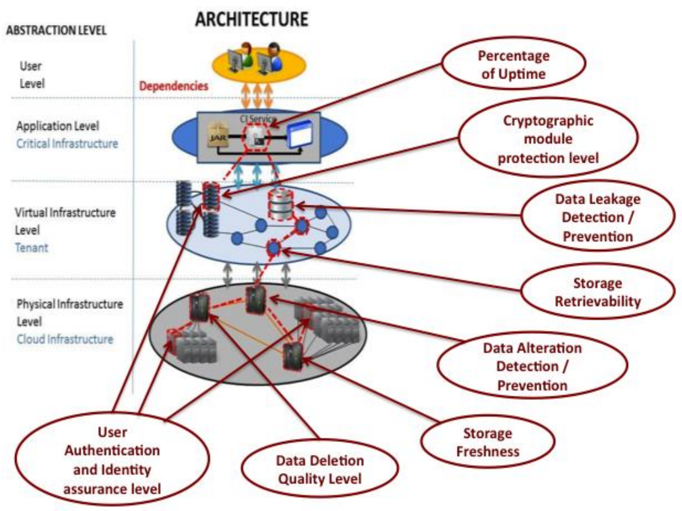
ARCHITECTURE
ABSTRACTION LEVEL
User Level
Dependencies
Application Level
Critical Infrastructure
CI Service
JAR
Virtual Infrastructure Level
Tenant
Physical Infrastructure Level
Cloud Infrastructure
Percentage of Uptime
Cryptographic module protection level
Data Leakage Detection / Prevention
Storage Retrievability
Data Alteration Detection / Prevention
Storage Freshness
User Authentication and Identity assurance level
Data Deletion Quality Level

Figure 4: Identified set of Security Properties across various architecture layers of the Cloud environment, mapped towards the SECCRIT vulnerability catalogue and CUMULUS property catalogue. Due to the fact that both catalogues enumerate large number of properties we only illustrated most representative ones for time being and will provide more detailed catalogue in our further work.

Table 3 presents a number of security properties that are relevant for the case study, their security property category, as well as the vulnerabilities and threats that are related to each of them. Moreover, the dependencies between these properties are also provided according to Figure 4. From this list we have selected four properties, e.g. SP_7, SP_4, SP_6 and SP_1 to proceed to the evaluation of our approach, as a starting point of our on-going research on multi-layer assurance dependencies policies.

C. *Scenario based assessment*

To demonstrate the approach we distinguish two specific use cases: the fundamental general tree model (illustrated in Figure 2- a) and the advanced tree model (illustrated in Figure 2- b). Both models illustrate a service through a general tree model, where each individual node represents a standalone entity of the particular service that is being evaluated. Furthermore, we use our set of identified security properties to demonstrate our approach by distinguishing the four most relevant properties SP_7, SP_4, SP_6 and SP_1 assigning them as SP_4, SP_3, SP_2 and SP_1 respectively. We implemented a random bit vector generator that generates four bit sets, regardless of the use case, and associates them with individual SPV for a particular object.

For the evaluation of our first use case scenario we illustrate a general tree model for each COE_i, $i \in \{B, C, D \ldots N\}$ generated SPV [SP_4, SP_3, SP_2, SP_1], as shown in Table 4 (a). We use the traversal post order method to recursively assess the use case scenarios.

Table 3: Security Properties, Vulnerabilities & Threats

ID	Security Property	Category	Vulnerability	Threats	Dependencies
SP_1	User Authentication and Identity assurance level	Identity Assurance	Loss of human-operated control point to verify security and privacy settings	Data Breaches Data Loss Shared Technology Vulnerabilities	None
			Insufficient authentication security, e.g., weak authentication mechanisms, on the Cloud management interface	Account or Service Traffic Hijacking Insecure Interfaces and APIs Malicious Insiders	
SP_2	Data deletion quality level	Data Disposal	Data recovery vulnerabilities, e.g., unauthorised access to data in memory or on disk from previous users	Data Breaches Account or Service Traffic Hijacking Insecure Interfaces and APIs Malicious Insiders Insufficient Due Diligence	None
SP_3	Storage Freshness	Durability			
SP_4	Data alteration prevention / detection	Integrity	Poor/ no integrity checks of the billing information	Data Breaches Insecure Interfaces and APIs Insufficient Due Diligence	SP_1, SP_2, SP_3
SP_5	Storage Retrievability	Durability	Poor/ no backup & restore strategy is in place to prevent the loss of billing information, e.g., in the case of a system failure	Data Breaches Insecure Interfaces and APIs Insufficient Due Diligence	SP_4
SP_6	Data leakage detection / prevention	Data Leakage	Poor/ no encryption of the VM data through a wide-area migration process	Data Breaches Malicious Insiders Shared Technology Vulnerabilities	SP_5
SP_7	Cryptographic module protection level	Key Management	Unmonitored and unencrypted network traffic between VMs is possible, e.g., for VMs on the same node through virtual network Unencrypted physical storage, which is the underlying for allocated virtual storage of the VMs	Insufficient Due Diligence Shared Technology Vulnerabilities Data Breaches Malicious Insiders	None
SP_8	Percentage of Up Time	Availability	Poor/ no implemented QoS (Quality of Service) services, e.g., to guarantee connection bandwidth required by the Cloud user. Only one ISP connection is considered for operation	Insufficient Due Diligence Shared Technology Vulnerabilities	SP_6
			Poor/ no failover mechanism, e.g., in case of losing one out of two ISP connections Missing redundant power connection leads to a higher risk of losing power	Denial of Service	

Due to the simplicity of the first use case scenario the traversal post order method only determines the sequence of the evaluated objects, which is {B, C, D … N} since we have a one-level deep tree. Consolidating this with our procedural algorithm from Figure 3 we conduct bitwise conjunction. In particular we start by conducting the procedure illustrated in equation (8) and implemented in our algorithm in Figure 3 on the SP_4. The result of this is 0. This indicates that according to Table 1 we discard all potential combination that fulfil SP_4 (upper eight combination 8-15) and reduce to 3-bit vector set for further evaluation. Our next sequential step, applies the same process on SP_3 resulting also to 0, which also lead to the same outcome, but reducing it into 2-bit vector. The next iteration for the SP_3 resulted to 1 that maps the remaining bit vector sets towards the assurance level two, therefore making the last bit irrelevant for the assurance since both potential outcomes (0 and 1) would lead towards assurance level two. Hence, the final vector, according to Table 1, associates the underlying Associated component set (ACS) of the root node with AL=2 is SPV = [001X]. This process is derived for each AC until we derive the final SPVi for each ACi. The final aggregation towards the root is defined with equation (8), which leverages the policies of Table 2 to decide whether both conditions of the DAL_{VS} and AL_{VS} are satisfied to determine the root assurance level (the equations (9, (10 and (11 have to be fulfilled.), In this particular case this is $CoE_A(AL)=2$. However, in case of multiple ACs it has to be also checked weather for each AC the minimum CAL is satisfied to fulfil a particular AL, as stated in Table 2 and defined by equation (12).

Table 4: Randomly generated SPV per individual CoE for demonstrating our algorithm Figure 3, via the use cases from Figure 2. Left table (a) is referring to the first use case scenario, Figure 2 (a), and table (b) refers to the second use case scenario Figure 2 (b).

(a)

	SP_4	SP_3	SP_2	SP_1
CoE_A	0	1	1	1
CoE_B	0	1	1	0
CoE_C	0	0	1	0
CoE_D	0	0	1	1
CoE_N	1	0	1	0

(b)

	SP_4	SP_3	SP_2	SP_1
CoE_A	0	1	1	0
CoE_B	1	0	0	0
CoE_C	1	0	1	0
CoE_D	1	0	1	1
CoE_E	0	1	0	1
CoE_F	0	1	1	0
CoE_G	1	0	0	1
CoE_H	0	1	1	0
CoE_I	1	0	0	0
CoE_J	1	0	0	0
CoE_K	1	0	1	1
CoE_L	0	1	0	0
CoE_M	0	1	1	1
CoE_N	0	1	0	1

To evaluate the second use case, i.e. the advanced tree model (see Figure 2- b), we generate for each COE_i, $i \in \{A, B, C, D \ldots N\}$ SPVi Table 4 (b). Due to the fact that the first use case tree is a subset of the tree in the second use case, we can apply the whole process conducted in the first use case scenario iteratively, until we aggregate the assurance towards the root. Therefore, in order to avoid redundancy we will just refer to the process explained in the first use case and extend it accordingly. The traversal post order method in the second use case, Figure 2 .b has the following sequence {D, F, L, M, N, G, B, H, C, I, J, K, D, A}. Therefore we marked 5 steps in Figure 2 b that illustrate this procedure. The first step will aggregate the assurance for ACS_G = (COE_i, $i \in \{L, M, N\}$) with the ALCP procedure which results in $CoE_G(AL)$ = [010X]. Then, as second step, when the Assurance level of CoE_G has been reached we aggregate the assurance of ACS_B = (COE_i, $i \in \{E, F, G\}$), e.g. $CoE_B(AL)$ = [0110]. The third step determines the assurance level of CoE_C directly according to one child node CoE_H, $CoE_H(AL)$=[0110]. The fourth step aggregates the assurance level of ACS_D = (COE_i, $i \in \{I, J, K\}$), $CoE_D(AL)$=[1001]. Finally the last step of the assessment process is to aggregate the assurance level of ToE_{AL} = ACS_A = (COE_i, $i \in \{B, C, D\}$), where $CoE_D(AL)$=[0110], by fulfilling the equations (9, (10 and (11 leads to the overall assurance of AL=4. However, just as in the first use case scenario, if dealt with multiple assurance class we have to use equation (12 to derive the final consolidated AL for a particular CoE.

V. Conclusion and Future work

In this paper we present an assurance methodology for Cloud security properties. This will support Cloud users in simplifying the assessment on whether a specific security level (i.e. assurance level) of a service can be maintained despite churn in the substitute components. The method supports multi-tenant environments and multi-layer environments. The scheme has been applied to two scenarios. This theoretic evaluation method shows efficient application of the proposed assurance assessment method over the use case where we demonstrate how services can be assessed according to a set of security properties with a defined set of policies.

Based on this work the next steps will provide a complete assurance class and security property catalogue that comprehensively covers the different aspects of Cloud environments. Furthermore, we are planning to use real-world applications from the SECCRIT and CUMULUS projects and benchmark them using the introduced scheme. As far as the model itself is concerned we will also further investigate the use of a binary tree model instead of the currently used general tree model, since we can easily transform a general tree to a binary tree model in order to empirically evaluate the performance of our algorithm.

Acknowledgment

The research presented in this paper has been funded by the European Commission, in the context of two Seventh Framework Programme (FP7) projects, the SECCRIT (Grant No. 312758) and the CUMULUS (Grant No. 318580).

References

[1] CSA, “The notorious nine: Cloud computing top threats in 2013”, v.1.0, Cloud Security Alliance, February 2013, available from: http://cloudsecurityalliance.org/research/top-threats/ [retrieved: April 2014]

[2] CUMULUS project, http://www.cumulus-project.eu/

[3] P. Mell and T. Grance: The NIST Definition of Cloud Computing. Technical Report Special Publication 800-145, National Institute of Standards and Technology (NIST), September 2011.

[4] SECCRIT project, https://seccrit.eu/

[5] CUMULUS Deliverable “D2.1 Security-aware SLA specification language and cloud security dependency model v1.01”, September 2013. Available from http://www.cumulus-project.eu/.

[6] Common Criteria (CC) for Information Technology Security Evaluation, CCDB USB Working Group, 2012, part 1-3. [Online]. Available: http://www.commoncriteriaportal.org.

[7] Scholler, M., Bless, R., Pallas, F., Horneber, J., & Smith, P. (2013, December) “An Architectural Model for Deploying Critical Infrastructure Services in the Cloud“, Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on (Vol. 1, pp. 458-466). IEEE.

[8] Florian, M., Paudel, S., & Tauber, M. (2013, December), “Trustworthy evidence gathering mechanism for multilayer cloud compliance”, Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for (pp. 529-530). IEEE.

[9] S. Paudel, Tauber, M., and Brandic, I., “Towards Taxonomy based Software Security Standard and Tool Selection for Critical Infrastructure IT in the Cloud”, The 8th International Conference for Internet Technology and Secured Transactions (ICITST-2013), 2013.

[10] A. Hudic, T. Hecht, M. Tauber, A. Mauthe and S. E. Cáceres, "Towards Continuous Cloud Service Assurance for Critical Infrastructure IT", The 2nd International Conference on Future Internet of Things and Cloud (FiCloud-2014), 2014

[11] M. Krotsiani, G. Spanoudakis, and K. Mahbub, “Incremental certification of cloud services,” in SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies, 2013, p. 7280.

[12] Cloud Security Alliance, Cloud Controls Matrix, Available from: https://cloudsecurityalliance.org/research/ccm/

[13] Payment Card Industry Data Security Standard (PCI DSS) Cloud Computing Guidelines, Available from: https://www.pcisecuritystandards.org/security_standards/documents.php?document=dss_cloud_computing_guidelines

[14] COBIT, IT Assurance Guide: Using COBIT, Control Objectives for Information and related Technology, 2007, information Systems Audit and Control Association.

[15] IT Baseline Protection Catalogs, Available from: http://www.bsi.de/gshb/index.htm

[16] National Institute of Standards and Technology, "Information Security Handbook: A Guide for Managers," NIST Special Publication 800-100, October 2006.

[17] ITAF, Information Technology Assurance Framework, 2nd ed., Information Systems Audit and Control Association, 2013.

[18] ENISA, Cloud Computing Information Assurance Framework, 1st ed., European Union Agency for Network and Information Security, 2009. Available from:http://www.enisa.europa.eu/

[19] Top Threats Working Group. "The Notorious Nine: Cloud Computing Top Threats in 2013." Cloud Security Alliance (2013).

[20] Jerry Busby, Lucie Langer, Marcus Schöller, Noor Shirazi and Paul Smith Deliverable: 3.1: "Methodology for Risk Assessment and Management", 2013, Available online: https://www.seccrit.eu/upload/D3-1-Methodology-for-Risk-Assessment-and-Management.pdf

[21] Patel, Pankesh, Ajith H. Ranabahu, and Amit P. Sheth. "Service level agreement in cloud computing." (2009).

[22] Buyya, Rajkumar, Chee Shin Yeo, and Srikumar Venugopal. "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities." HPCC'08. 10th IEEE International Conference on. Ieee, 2008.

[23] OpenStack, http://www.openstack.org/

[24] CloudStack, http://cloudstack.apache.org/