A Formal Model for Isolation Management in Cloud Infrastructure-as-a-Service

Khalid Zaman Bijon[1], Ram Krishnan[2], and Ravi Sandhu[1]

[1] Institute for Cyber Security & Department of Computer Science,
[2] Institute for Cyber Security & Department of Electrical and Computer Engineering
University of Texas at San Antonio

Abstract. Datacenters for cloud infrastructure-as-a-service (IaaS) consist of a large number of heterogeneous virtual resources, such as virtual machines (VMs) and virtual local area networks (VLANs). It takes a complex process to manage and arrange these virtual resources to build particular computing environments. Misconfiguration of this management process increases possibility of security vulnerability in this system. Moreover, multiplexing virtual resources of disjoint customers upon same physical hardware leads to several security concerns, such as cross-channel and denial-of-service attacks. Trusted Virtual Datacenter (TVDc) is a commerical product which informally presents a process to manage strong isolation among these virtual resources in order to mitigate these issues. In this paper, we formally represent this TVDc management model. We also develop an authorization model for the cloud administrative-user privilege management in this system.

Keywords: isolation, virtual resource management, cloud computing.

1 Introduction

Cloud service providers (CSPs) of infrastructure-as-a-service (IaaS) offer a number of heterogeneous virtual resources, e.g. virtual machine (`VM`), to the cloud customers. Management process of these virtual resources is very complex since the datacenter of a CSP may contain thousands of these resources from different customers and it also increases likelihood of misconfiguration which makes this system vulnerable. On the other hand, multi-tenancy — sharing a hardware resource among different customers' workloads enables cost reductions and economic benefits of the CSPs. However, this situation may also lead to different security vulnerabilities for the customers' workloads such as cross-channel attacks and denial-of-services.

Recently, trusted virtual datacenter (TVDc) [3] proposed an isolation management process in cloud IaaS that addresses these multi-tenancy and management issues. In TVDc virtual machines and their associated resources, such as virtual bridge and virtual local access network (VLAN), are grouped into trusted virtual domains (TVDs). Each TVD, represented as a security clearance (also

referred to as color), enforces an isolation policy towards its group members. More specifically, resources are only allowed to interact with each other if they are assigned to same color. For instance, VMs with same color can communicate and a VM can run on a hypervisor only if this hypervisor has the same color as that VM. The main goal of this process is to isolate customer workloads from each other. As described in [3], the purpose of this isolation is to reduce the threat of co-locating workloads from different customers by preventing any kind of data flow among these workloads where the data might includes sensitive information of the customers or any virus or malicious code. Again, this simple management process also reduces the incidence of misconfiguration of the virtual-resource management tasks. TVDc [3] also develops a hierarchical administration model based on trusted virtual domains.

In this paper, we develop a formal model for TVDc which we call Formalized-TVDc (also referred as *F-TVDc*). We leverage the attribute based system proposed in [16] in order to represent different properties of the virtual resources, such as color. Then, resources with similar attributes will be arranged together to built a particular computing environment. For instance, a VM can run on a hypervisor (host) if the host has same color of the VM. This formal model consists authorization models for three types of administrative-user operations. We also derive enforcement process for the constraints discussed in [3].

2 Background: Trusted Virtual Datacenter (TVDc)

Trusted virtual datacenter (TVDc) [3] manages isolation by defining a trusted virtual domain (TVD) for a set of VMs and their associated resources that constitute a common administrative unit. The boundary of a TVD is maintained by assigning the TVD identifier to the respective VMs and resources. A TVD identifier represents a security clearance (also referred to as a color to emphasize there is no ordering or structure). For instance, a color can represent the virtual resources of a particular customer or virtual resources running specific workloads of a customer. Hence, basically, a color represents a particular context for the assigned VMs and resources. Figure 1, from [3], shows the TVDc view of the virtual resources running two physical data centers and their resources, such as servers, storage, and network components. The TVDc view separates the association of physical resources for each color. For instance, in figure 1, the red color includes VMs 3, 7, 9, and 11, and associated storages.

In order to manage this isolation process, three different administrative roles are proposed in TVDc: IT datacenter administrator, TVDc administrator , and tenant administrator. Administrative users (also generally referred as admin-users) having IT datacenter administrator role are the superusers in this system. Their main task is to discover the physical and the virtual resources and group the discovered resources into TVDcs. They also define the security labels or colors. IT datacenter admin-users further can assign a TVDc group to both TVDc and tenant admin-users, and assign a set of colors to TVDc admin-users in order to delegate isolation management of resources in that specific TVDc group.

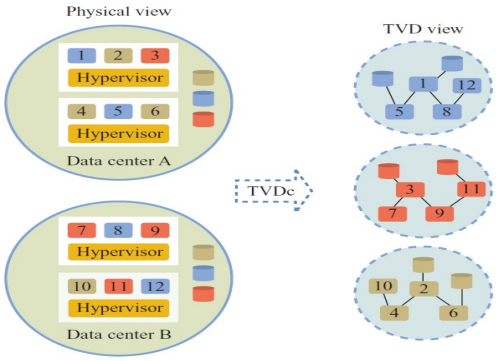

Fig. 1: *TVDc View of the IaaS Cloud Resources [3]*

The tasks of a TVDc admin-user include assigning colors to the resources, belong to the same TVDc group, from her assigned set of colors. She also assigns a color to an admin-user if the admin-user is in the same TVDc group and assigned to the tenant administrator role. The job function of a tenant admin-user is to perform basic cloud administrative operations on the cloud resources, such as boot a `VM` and connect a `VM` to a virtual network, if both the resources and the tenant admin-user are in same TVDc group and assigned with same color.

This color-driven isolation management process supports four different types of isolation which are described as follows:

1. **Data Sharing:** In this model, `VM`s can share data with each other only if they have common colors. In order to constrain this, a `VM` is allowed to connect to a VLAN only if both the `VM` and the VLAN have common colors and, therefore, it is restricted to communicate with `VM`s having same color.
2. **Hyperviosr (`host`) Authorization:** A `host` is assigned to a set of colors and is only allowed to run a `VM` having a color in that set. Therefore, a `host`'s capability to run `VM`s is isolated to its assigned colors.
3. **Colocation Management:** Two colors can be conflicted with each other if context of operation is mutually exclusive. Colors can be declared to be conflicting and two `VM`s with conflicted colors are prohibited from running in same `host`. For instance, `VM`s A and B with color red and blue respectively which have been declared to be conflicting cannot run on same `host`.
4. **Management Constraints:** For management isolation, tenant administrative roles are created where each user having this role is restricted to perform administrative operation within a single color.

3 Formal Isolation Management Model (*F-TVDc*)

In this section, we formalize the isolation management process in cloud IaaS which is informally described in TVDc [3]. We call the resulting model as Formalized-TVDc (*F-TVDc*) for ease of reference and continuity. In *F-TVDc*, different properties of the cloud entities are represented as assigned attributes to them. For

Table 1. Basic Element-Sets

CLR = Finite set of existing colors
VDc = Finite set of existing virtual data centers
AROLE = {itAdmin, tvdcAdmin, tntAdmin}
AU = Finite set of existing admin-users
VM = Finite set of existing virtual machines
VMM = Finite set of existing hypervisors
BR = Finite set of existing virtual bridges
VLAN = Finite set of existing virtual LANs

instance, a virtual machine(VM) attribute *color* represents assigned colors to that VM and an administrative user (admin-user) attribute *adminRole* represents assigned role to that user. For this purpose we utilize the attribute based system [16], specifically its attribute representation for the entities in a system. In this attribute based representation of *F-TVDc*, the admin-users can manage the resources in data-center by assigning proper attributes to them. For instance, a TVDc admin-user can assign a set of colors to a host and, consequently, the host is authorized to run a VM if the assigned color of the VM is an element of the set of colors assigned to the host. *F-TVDc* also formally represents an authorization model for these admin-user privileges.

3.1 Basic Components

The sets that contain basic entities of *F-TVDc* are shown in table 1. In *F-TVDc*, CLR contains the existing colors/clearances in the system. We will see later that the colors from CLR will be assigned to the cloud-resources' and admin-users' respective attributes, such as the *admincolor* attribute of an admin-user. The data-center is divided into multiple virtual data-centers. VDc contains the names of these virtual data-centers. There are three administrative roles: IT administrator (itAdmin), TVDc administrator (tvdcAdmin), and tenant administrator (tntAdmin) which are contained in set AROLE. The set AU contains all admin-users in the system. VMM and VM contains the current existing hypervisors (hosts) and the virtual machines(VMs) in the system. Similarly, existing virtual LANs and virtual bridges are contained in set VLAN and BR respectively.

3.2 Attributes

Attributes characterize properties of an entity and are modeled as functions. *F-TVDc* recognize two types of attribute functions for each entity depending on the nature of the functions values: atomic-valued and set-valued. For instance, an admin-user attribute function *adminRole* can only take a single value that indicates the assigned role to that user. On the other hand, the attribute function *admincolor*, representing the assigned colors to an admin-user, can take multiple values. For convenience we understand attribute to mean attribute function for ease of reference. Attributes of an entity, let's say VM attributes, can be formally defined as follows:

Table 2. Attributes Specification

Entity	Attributes	attType	SCOPE
Admin-User	*adminRole*	atomic	AROLE
	adminvdcenter	set	VDc
	admincolor	set	CLR
Virtual Machine	*vmvdcenter*	atomic	VDc
	vmcolor	atomic	CLR
	host	atomic	VMM
	status	atomic	{Running, Stop}
	bridge	set	BR
Hypervisor (`host`)	*vmmvdcenter*	atomic	VDc
	vmmcolor	set	CLR
	vm	set	VM
Virtual Bridge	*brvdcenter*	atomic	VDc
	brcolor	atomic	CLR
	vm	set	VM
	vlan	atomic	BR
Virtual LAN	*vlanvdcenter*	atomic	VDc
	vlancolor	set	CLR
	bridge	set	BR

– $\text{ATTR}_{\texttt{VM}}$ is the finite set of `VM` attributes, where attType: ATTR $\rightarrow$ {set, atomic}.

– For each $att \in \text{ATTR}_{\texttt{VM}}$, SCOPE_{att} is a finite set of atomic values which determines the range of att as follows:

$$\text{Range}(att) = \begin{cases} \text{SCOPE}_{att} \text{ if attType}(att) = \text{atomic} \\ \\ \mathcal{P}(\text{SCOPE}_{att}) \text{ if attType}(att) = \text{set} \end{cases}$$

where $\mathcal{P}$ denotes the power set of a set.

– An attribute is a function that maps each `VM`$\in$VM to a value in range, i.e.,

$$\forall att \in \text{ATTR}_{\texttt{VM}}.\ att : \mathsf{VM} \rightarrow \text{Range}(att)$$

Similary, attributes of other entities can be defined. Table 2 shows the necessary attributes for the entities in *F-TVDc* which are described as follows:

– Admin-User (`aUser`) attributes: *adminRole* attribute of an admin-user (`aUser`) specifies the assigned administrative role to `aUser`. Note that, an `aUser` can get only one administrative role, hence, *adminRole* is an atomic attribute. Attribute *adminvdcenter* represents the assigned virtual data-center of an `aUser`. If the `aUser` is an IT administrator then its *adminvdcenter* contains all the members in VDc. Otherwise *adminvdcenter* of an `aUser` contains only one element from VDc. Similarly, *admincolor* specifies the assigned colors to

an `aUser`. If an `aUser` is an IT administrator then her *admincolor* contains all the elements of CLR. On the other hand, an `aUser` having tvdcAdmin role can get subset of colors from CLR and a tntAdmin gets only one color. Section 4 represents the operations to assign values of these `aUser` attributes.
- Virtual machine (`VM`) attributes: The `VM` attribute *host* represents the hypervisor (`host`) where a `VM` is running. Attribute *bridge* represents the connected `bridge`s of a `VM`. *vmvdcenter* represents the virtual data-center a `VM` belongs to and *vmcolor* specifies the assigned color to that `VM`.
- Hypervisor (`host`) attributes: The *VM* attribute represents the running `VM`s in a `host`. The *vmmvdcenter* attribute represents its virtual data-centers and *vmcolor* the assigned colors to it.
- Virtual Bridge (`bridge`) attributes: The *VM* attribute of `bridge` specifies the connected `VM`s to a `bridge`. Similarly, *vlan* specifies the `vlan` to which a bridge is connected. Similar to the other entities, *brcolor* and *brvdcenter* represent the virtual data-center and color assigned to a `bridge`. Note these are atomic in this instance.
- Virtual LAN (`vlan`) attributes: The *bridge* attribute of a `vlan` specifies the connected virtual bridges to it. Also, *vlancolor* and *vlanvdcenter* represents the virtual data-center and colors assigned to a `vlan`.

4 Administrative Models

In this section, we discuss administrative operations for the three types of admin-users. Table 3 formally specifies the set of administrative operations for the IT admin-user. The first column specifies the operation name and parameters. The second column specifies the conditions that need to be satisfied to authorize the operation. Attributes and sets that will be updated after an authorized operation are listed in the third column, with the $'$ symbol indicating the value after the update. Administrative operations of Table 3 are discussed below.

- **CreateVDC**: First column of table 3 shows that this function takes two parameters: users u and a virtual data-center vdc. Then, in second column, these parameters need to satisfy the given formula which checks if u belongs to AU, *adminRole* of u is itAdmin and vdc is not present in VDc. If the precondition is satisfied, in column 3, vdc is created by adding it to set VDc.
- **CreateCl** and **RemoveCl**: Using these two operations, an itAdmin can create a new color cl and remove an existing color cl.
- $\mathbf{Add\_Cl_{TVDcAdmin}}$: This function takes three parameters: users u1 and u2, and a color cl. These parameters need to satisfy the given formula in column 2 which checks if u1 has role itAdmin, u2 has role tvdcAdmin and cl is a valid member in CLR. If so, color cl is assigned to tvdcAdmin u2 by adding cl to *tvdcAdmincolor* attribute of u2, as shown in column 3.
- $\mathbf{Rem\_Cl_{TVDcAdmin}}$: Using this operation, an itAdmin removes a color cl from an admin-user having role tvdcAdmin.
- $\mathbf{Assign\_VDC_{Admin}}$: Using this operation an itAdmin user assigns a virtual data-center vdc to attribute *adminvdcenter* of a tvdcAdmin or tntAdmin user.

Table 3. IT admin-user Operations

Operation	Precondition	Updates
CreateVDC(u, vdc) /*Creates a virtual data-center vdc*/	u∈AU∧vdc∉VDc∧ *adminRole*(u)=itAdmin	VDc′=VDc∪{vdc}
CreateCl(u,cl) /*Creates a color cl*/	u∈AU∧cl∉CLR∧ *adminRole*(u)=itAdmin	CLR′=CLR∪{cl}
RemoveCl(u,cl) /*Removes a color cl*/	u∈AU∧cl∈CLR∧ *adminRole*(u)=itAdmin	CLR′=CLR-{cl}
Add_Cl$_{\mathbf{TVDcAdmin}}$(u1,u2,cl) /*Adds cl to tvdcAdmin u2*/	u1∈AU∧*adminRole*(u1)= itAdmin∧u2∈AU∧cl∈CLR∧ *adminRole*(u2)=tvdcAdmin∧ cl∉*admincolor*(u2)	*admincolor*′(u2)← *admincolor*(u2)∪{cl}
Rem_Cl$_{\mathbf{TVDcAdmin}}$(u1,u2,cl) /*Removes cl from tvdcAdmin u2*/	u1∈AU∧*adminRole*(u1)= itAdmin∧u2∈AU∧ *adminRole*(u2)=tvdcAdmin∧ cl∈*admincolor*(u2)	*admincolor*′(u2)← *admincolor*(u2)-{cl}
Assign_VDC$_{\mathbf{Admin}}$(u1,u2,vdc) /*Assigns virtual datacenter vdc to tvdcAdmin or tntAdmin u2*/	u1∈AU∧*adminRole*(u1)= itAdmin∧u2∈AU∧vdc∈VDc∧ (*adminRole*(u2)=tvdcAdmin∨ *adminRole*(u2)=tntAdmin)	*adminvdcenter*′(u2)← {vdc}
Assign_VDC$_{\mathbf{VM}}$(u,vm,vdc) /*Assigns virtual datacenter vdc to a virtual machine vm*/	u ∈AU∧vm∈VM∧vdc∈ VDc∧*adminRole*(u)=itAdmin	*vmvdcenter*′(vm)← vdc
Assign_VDC$_{\mathbf{VMM}}$(u,vmm,vdc) /*Assigns virtual datacenter vdc to a hypervisor vmm*/	u ∈AU∧vmm∈VMM∧vdc∈ VDc∧*adminRole*(u)=itAdmin	*vmmvdcenter*′(vmm)← vdc
Assign_VDC$_{\mathbf{BR}}$(u,br,vdc) /*Assigns virtual datacenter vdc to a bridge br*/	u ∈AU∧br∈BR∧vdc∈ VDc∧*adminRole*(u)=itAdmin	*brvdcenter*′(br)← vdc
Assign_VDC$_{\mathbf{VLAN}}$(u,vlan,vdc) /*Assigns virtual datacenter vdc to a virtual lan vlan*/	u ∈AU∧vlan∈VLAN∧vdc∈ VDc∧*adminRole*(u)=itAdmin	*vlanvdcenter*′(vlan)← vdc

- **Assign_VDC$_{\mathbf{VM}}$**: A virtual data-center vdc is assigned to the *vmvdcenter* attribute of a virtual machine vm. This value specifies that vm belongs to virtual data-center vdc.
- **Assign_VDC$_{\mathbf{VMM}}$**: Similarly, a virtual data-center vdc is assigned to the *vmmvdcenter* attribute of a hypervisor vmm.

Table 4. TVDc-ADMIN Operations

Operation	Precondition	Updates
Assign_Cl$_{\mathbf{TAdmin}}$(u1, u2,cl) /*Assigns a color cl to a tntAdmin u2*/	u1∈AU∧u2∈AU∧$adminRole$(u1)= tvdcAdmin∧$adminRole$(u2)= tntAdmin∧ $adminvdcenter$(u1)= $adminvdcenter$(u2)∧cl∈ $admincolor$(u1)∧cl∉$admincolor$(u2)	$admincolor'$(u2)← {cl}
Rem_Cl$_{\mathbf{TAdmin}}$(u1, u2,cl) /*Removes the color cl from a tntAdmin u2*/	u1∈AU∧u2∈AU∧$adminRole$(u1)= tvdcAdmin∧$adminRole$(u2)= tntAdmin∧ $adminvdcenter$(u1)= $adminvdcenter$(u2)∧cl∈ $admincolor$(u1)∧cl∈$admincolor$(u2)	$admincolor'$(u2)← ϕ
Add_Cl$_{\mathbf{VMM}}$(u,vmm,cl) /*Adds a color cl to hypervisor vmm*/	u∈AU∧vmm∈VMM∧cl∈ $admincolor$(u)∧$adminRole$(u)= tvdcAdmin∧$adminvdcenter$(u)= $vmmvdcenter$(vmm)	$vmmcolor'$(vmm)← $vmmcolor$(vmm)∪ {cl}
Assign_Cl$_{\mathbf{VM}}$(u,vm,cl) /*Assigns a color cl to virtual machine vm*/	u∈AU∧vm∈VM∧cl∈ $admincolor$(u)∧$adminRole$(u)= tvdcAdmin∧$adminvdcenter$(u)= $vmvdcenter$(vm)	$vmcolor'$(vm)← {cl}
Assign_Cl$_{\mathbf{BR}}$(u,br,cl) /*Assigns a color cl to bridge br*/	u∈AU∧br∈BR∧cl∈ $admincolor$(u)∧$adminRole$(u)= tvdcAdmin∧$adminvdcenter$(u)= $brvdcenter$(br)	$brcolor'$(br)←{cl}
Add_Cl$_{\mathbf{VLAN}}$(u,vlan,cl) /*Adds a color cl to virtual LAN vlan*/	u∈AU∧vlan∈VLAN∧cl∈ $admincolor$(u)∧$adminRole$(u)= tvdcAdmin∧$adminvdcenter$(u)= $vmmvdcenter$(vlan)	$vlancolor'$(vlan)← $vlancolor$(vlan)∪ {cl}

- **Assign_VDC$_{\mathbf{BR}}$**: This operation assigns a virtual data-center named vdc to *brvdcenter* attribute of a virtual bridge br.
- **Assign_VDC$_{\mathbf{VLAN}}$**: A virtual data-center, vdc, is assigned to the *vlanvdcenter* attribute of a virtual LAN vlan.

Similarly, table 4 shows the operations for TVDc admin-users. The TVDc admin-users are responsible to assign colors to the tntAdmins and the resources in data-centers where the TVDc admin-users are authorized to exercise their priviledges. The description of these operations are as follows:

- **Assign_Cl$_{\mathbf{TAdmin}}$**: A tvdcAdmin u1 assigns a color cl to a tntAdmin u2. Authorization of this operation needs to satisfy the precondition that u1

Table 5. Tenant-ADMIN Operations

Operation	Precondition	Updates
Boot(u,vm,vmm) /*Boots a virtual machine vm in a `host` vmm*/	$vmcolor(vm) \in admincolor(u) \wedge admincolor(u) \cap vmmcolor(vmm) \neq \emptyset \wedge adminvdcenter(u) = vmvdcenter(vm) \wedge adminvdcenter(u) \in vmmvdcenter(vmm) \wedge vmcolor(vm) \in vmmcolor(vmm) \wedge vm \in$ VM $\wedge vmm \in$ VMM $\wedge$ Evaluate_CLocConst(vm,vmm) $\wedge u \in$ AU $\wedge adminRole(u) =$ tntAdmin $\wedge status(vm) =$ Stop	$host'(vm) \leftarrow vmm$ $vm'(vmm) \leftarrow vm(vmm) \cup vm$ $status'(vm) \leftarrow$ Running
ConVmToBr(u,vm,br) /*Connects virtual machine vm to a virtual bridge br*/	$u \in$ AU $\wedge vmcolor(vm) \in admincolor(u) \wedge brcolor(br) = vmcolor(vm) \wedge brcolor(br) \in admincolor(u) \wedge br \in$ BR $\wedge vm \in$ VM $\wedge adminvdcenter(u) = vmvdcenter(vm) \wedge adminvdcenter(u) = brvdcenter(br) \wedge adminRole(u) =$ tntAdmin	$bridge'(vm) \leftarrow br$ $vm'(br) \leftarrow vm(br) \cup \{vm\}$
ConBrToVLAN(u,br,vlan) /*Connects a virtual bridge br to a virtual LAN vlan*/	$u \in$ AU $\wedge brcolor(br) \in admincolor(u) \wedge brcolor(br) \in vlancolor(valn) \wedge vlancolor(vlan) \cap admincolor(u) \neq \emptyset \wedge br \in$ BR $\wedge vlan \in$ VLAN $\wedge adminvdcenter(u) = vlanvdcenter(vm) \wedge adminvdcenter(u) = vlanvdcenter(br) \wedge adminRole(u) =$ tntAdmin	$bridge'(vlan) \leftarrow bridge(vlan) \cup \{br\}$ $vlan'(br) \leftarrow vlan$

and u2 are in the same virtual data-center. Also, the *admincolor* attribute of u2 must contain cl.

- **Rem_Cl$_{\mathbf{TAdmin}}$**: Using this operation a tvdcAdmin removes a color from tntAdmin.
- **Add_Cl$_{\mathbf{VMM}}$**: This operation adds a color cl to a `host` named vmm if vmm and tvdcAdmin are in same virtual data-center. Note that, a `host` can contain multiple colors.
- **Assign_Cl$_{\mathbf{VM}}$**, **Assign_Cl$_{\mathbf{BR}}$**, and **Add_Cl$_{\mathbf{VLAN}}$**: Using first two operations operations a tvdcAdmin u assigns a color cl to a virtual machine vm and a bridge br respectively. Also, using **Add_Cl$_{\mathbf{VLAN}}$** the tvdcAdmin adds a color to the *vlancolor* attribute of a virtual LAN. Note that, *vlancolor* attribute can contain multiple colors since a virtual LAN can be connected to multiple virtual bridges.

Now, table 5 shows the administrative operations for tenant admin-users and preconditions to authorize these operations. The operations of the tenant admin-users are to manage cloud resources within their assigned TVD groups, i.e., colors. The operations are described as follows:

– **Boot**: Using this operation a tenant admin-user u boots a `VM` vm in a `host`vmm. Table 5 shows the necessary precondition in order to authorize this operation. The precondition verifies if the u has same color of the vm which is basically an implementation of the management isolation constraint shown in section 2. In addition to that, the precondition also checks if these three entities belong to the same data-center. It also verifies if the `host` vmm's *vmmcolor* attribute contains the color of the vm's assigned color in *vmcolor* which is an implementation of `host` authorization isolation which is also shown in section 2. The authorization process of this operation also calls Evaluate_CLocConst function to satisfy the co-location management constraint, also given in section 2, for the vm with other running `VM`s in vmm. The algorithm 1 shows the evaluation process of Evaluate_CLocConst. Upon successful checking of these conditions the vm is scheduled to the vmm.
– **ConVmToBr**: It connects a `VM` to a virtual bridge . A `VM` can only connects to `bridge` if they have same color and they belongs to the same data-center.
– **ConBrToVLAN**: Using this function a tenant admin-user connects `bridge` to a VLAN. They can be connected if color of the `bridge` is present in the *vlancolor* attribute of the VLAN.

Algorithm 1 Colocation Constraints Verification

```
1: procedure EVALUATE_CLOCCONST(reqVm,vmm)
2:     Flag=True
3:     for vm∈VM do
4:         if host(vm)=vmm then
5:             for conele∈ConflictColor do
6:                 if vmcolor(reqVm)≠ vmcolor(vm) then
7:                     if vmcolor(vm)∈conele∧vmcolor(reqVm)∈conele∧status(vm)=Running then
8:                         Flag=False
9:                         return Flag
10:                    end if
11:                end if
12:            end for
13:        end if
14:    end for
15:    return Flag
16: end procedure
```

Algorithm 1 shows the evaluation algorithm of the co-location constraints. It takes two inputs: requested `VM` (reqVm) and the `host` (vmm). For each `VM` running in the vmm, this algorithm verifies if there is any conflict between the assigned color to the *vmmcolor* attribute of `VM` with the assigned color to the *vmmcolor* of reqVm. Attribute values can have different type of conflicts that can represent various relationships among these such as mutual-exclusion, pre-condition, etc. A generalized approach to represent the various types of attribute conflict-relations are shown in Bijon et al [4, 5]. Here, the conflicting values, i.e. colors, of the attribute *vmmcolor* are stored in a set called ConflictColor where each element in the set contains a set colors that are conflicting with each-other. Formally this set is defined as follows,

ConflictColor = $\{\text{conele}_1, \text{conele}_2, ..., \text{conele}_n\}$ where $\text{conele}_i \subseteq \mathsf{CLR}$

If algorithm 1 identifies no conflicts between reqVm and all running `VM` in vmm, it returns True. Otherwise, it returns a False.

5 Related Work

Presently, cloud providers, such as Amazon Elastic Compute Cloud [1], are highly multiplexed shared resources among different clients for achieving on demand scalability and cost effectiveness, although, it raises several issues making security and performance predictability a key concern for the customers [11]. For instance, Ristenpart et al [19] shows that it is possible to initiate a side-channel attacks from a `VM` to another co-locating `VM`s in same server. Rocha et al [20] shows different attacking scenarios including stealing cleartext password, private keys, etc. when tenants run their workloads without proper control in cloud. Moreover, complex and limited virtual resource management mechanism in cloud increases the likelihood of the risk of possible misconfiguration. Seawall [21] shows that arbitrary sharing of network, in cloud, may cause denial of service attack and performance interferences. Wei et al [23] shows that uncontrolled snapshots and uses of images cause security risk for both creator and user of it. Sivathanu et al [22] presents an experimental analysis on I/O performance bottleneck when virtual storages are placed arbitrarily in physical storage. Also, several performance and security issues, are basically resulted from unorganized management and multiplexing of resources, are summarized in [12, 14, 15].

In this paper we develop a formal model for isolation management of both admin-users and cloud resources that provides a simple but effective resource management mechanism for cloud. This isolation management is informally addressed in Trusted Virtual Datacenter(TVDc) [2]. TVDc builds upon Trusted Virtual Domains [2,3], which provides strong isolation that significantly enhance the security and management capabilities in cloud IaaS environment. Cabuk et al [8] also utilize Trusted Virtual Domains for the orchestration of various cloud resources. Lohr et al [17] propose isolation management in cloud system which is designed for medical services. This work proposes that the end-user, e.g. a patient, should be able to divide the execution environment for applications into separated domains isolated from each other so that medical data of a patient can only resides within her personal domain. Also, Bleikertz et al [6] develop an assurance language for isolation management in cloud computing environment. Prior literature also contains several processes on users authorization and access control models for cloud IaaS including [7,9] and, also, various processes for virtual resource management, e.g., virtual machine placement algorithms [10,13,18] for improving different aspects, e.g. high performance, load balancing.

6 Conclusion

In this paper, we formally represent an isolation management process of virtual resources in cloud IaaS. We utilize attribute-based system [16] in order to represents different properties of cloud resources, such as colors (security clearance), as assigned attributes to them. Then, resources are organized together based on similar attribute values. We also develop a process to enforce co-location constraints for the `VM`s having conflict with each other which prohibits them to be scheduled in same physical server.

Acknowledgement. This work is partially supported by the NSF (CNS-1111925) and AFOSR MURI grants (FA9550-08-1-0265).

References

1. Aws identity and access management. *https://http://aws.amazon.com/iam/*.
2. S. Berger et al. TVDc: managing security in the trusted virtual datacenter. *ACM SIGOPS Operating Systems Review*, 42(1):40–47, 2008.
3. S. Berger et al. Security for the cloud infrastructure: Trusted virtual data center implementation. *IBM Journal of R&D*, 53(4):6–1, 2009.
4. K. Z. Bijon, R. Krishman, and R. Sandhu. Constraints specication in attribute based access control. *SCIENCE*, 2(3):pp–131, 2013.
5. K. Z. Bijon, R. Krishnan, and R. Sandhu. Towards an attribute based constraints specification language. In *Social Computing*, pages 108–113. IEEE, 2013.
6. S. Bleikertz and T. Groß. A virtualization assurance language for isolation and deployment. In *Policies for Distributed Systems and Networks*. IEEE, 2011.
7. S. Bleikertz, A. Kurmus, Z. Nagy, and M. Schunter. Secure cloud maintenance - protecting workloads against insider attacks. In *Proc. of the ASIACCS*, 2012.
8. S. Cabuk et al. Towards automated security policy enforcement in multi-tenant virtual data centers. *Journal of Computer Security*, 18(1):89–121, 2010.
9. J. M. A. Calero et al. Toward a multi-tenancy authorization system for cloud services. *Security & Privacy, IEEE*, 8(6):48–55, 2010.
10. L. Cherkasova et al. Comparison of the three cpu schedulers in xen. *SIGMETRICS Performance Evaluation Review*, 35(2):42–51, 2007.
11. B. Claybrook. Comparing cloud risks and virtualization risks for data center apps. *http://searchdatacenter.techtarget.com/tip/ 0,289483,sid80 gci1380652,00.html*.
12. W. Dawoud, I. Takouna, and C. Meinel. Infrastructure as a service security: Challenges and solutions. In *IEEE INFOS*, pages 1–8, 2010.
13. A. Gupta et al. Hpc-aware vm placement in infrastructure clouds. In *IEEE Intl. Conf. on Cloud Engineering*, volume 13, 2013.
14. K. Hashizume et al. An analysis of security issues for cloud computing. *Journal of Internet Services and Applications*, 4(1):1–13, 2013.
15. A. Jasti, P. Shah, R. Nagaraj, and R. Pendse. Security in multi-tenancy cloud. In *IEEE ICCST*, pages 35–41, 2010.
16. X. Jin, R. Krishnan, and R. Sandhu. A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC. In *DBSec*, 2012.
17. H. Löhr et al. Securing the e-health cloud. In *Proceedings of the 1st ACM International Health Informatics Symposium*, pages 220–229. ACM, 2010.
18. K. Mills, J. Filliben, and C. Dabrowski. Comparing vm-placement algorithms for on-demand clouds. In *IEEE CloudCom*, pages 91–98, 2011.
19. T. Ristenpart et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proc. of the ACM CCS*, 2009.
20. F. Rocha and M. Correia. Lucy in the sky without diamonds: Stealing confidential data in the cloud. In *Proc. of the IEEE DSN-W*, 2011.
21. A. Shieh et al. Sharing the data center network. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, 2011.
22. S. Sivathanu, L. Liu, M. Yiduo, and X. Pu. Storage management in virtualized cloud environment. In *IEEE CLOUD*, pages 204–211, 2010.
23. J. Wei et al. Managing security of virtual machine images in a cloud environment. In *Proc. of the ACM workshop on Cloud computing security*, 2009.