Enabling Security Uniformly Across Cloud Systems

Sean Thorpe
University of Technology
237 Old Hope Road
Kingston, Jamaica
thorpe.sean@gmail.com

Indrajit Ray
Colorado State University
Fort Collins, USA
indrajit@cs.colostate.edu

Tyrone Grandison
IBM Almaden Research
Silicon Valley
USA
tyroneg@us.ibm.com

ABSTRACT
Compute cloud interoperability across heterogeneous distributed virtual machines (VM) is an emergent and challenging problem. Security administrators are currently unable to definitively audit cross platform transactions. In order to provide monitored cross platform support, this paper represents a first attempt to model security in cloud systems. A source theoretical policy framework is defined and a formal mapping model articulated that binds to specific VM attribute functional policies, which are used by the cloud environments within which the security administrator has applications deployed. These policies can be used to affect tangible yet flexible access control measures within these abstract environments. Our novel approach refers to the use of a VM attribute specification language that we refer to as the Global Virtual Machine Attribute Policy Auditor (GVMAPA).We use GVMAPA to express the multilevel security, hierarchical attribute based policies and organizational constraints such as separation of duty that are urgently needed within this VM administrator controlled environment. Our work is inspired by previous work in [7, 11, 12, 13] and referenced based on NIST guidelines.

General Terms
Security, Specification, language, standardization

Keywords
Access controls, Virtual Machine, Attribute, Policy, Auditor.

1. INTRODUCTION
We envisage fine grained access control (AC) systems to allow a security administrator to specify relations between AC attributes for both homogenous and heterogeneous dynamic virtual compute clouds. With this capability, an AC system is able to maintain hierarchical orders of the attributes of the AC elements (subjects, actions, objects) as observed on most static grids. By the same extension we should be able to scale this concern to a virtual compute cloud and suitably provide level of assurances (LOA) against the use of the attribute relations. This concern underscores the relevance for our GVMAPA within the global system administrator environment.

The expression of privilege inheritance relations is essential for many popular AC models such as **Bell-La Padula** [1] and **Biba** [2] (BLPB) of **Multilevel Security** (MLS) [3], and **Hierarchical Role Based Access Control** (HRBAC) [4] as well as constraint policies such as **Separation of Duty** (SOD) [5].

The syntactic and semantic supports of attribute relation (AR) specifications in AC mechanism or languages allow not only accurately specifying but also efficiently enforcing the relation-based AC models and policy constraints. The specific advantages of such capabilities include:

- Specifying hierarchical relations for the inherit or inherited privileges of subjects, actions, and objects in AC policies. For example, if subject *X* is **related** to subject *Y* then subject *X* **inherits** all the access privileges of subject *Y.*

- Efficient management of Ac rules, such that AC policy administrator can modify privileges based on attribute groups and relations without leaking access permissions. Also, through a GUI it is possible to display all the linkages of existing related attributes, thus providing a complete view of the current
privilege assignments.

Performance enhancement for evaluating access requests, because the AC system does not have to go through all the AC rules to collect attribute information fort the grant decision if higher level attributes of the request can be found to match the rule. As measure of the functionality for our Virtual Machine attribute policy engine there is the need for an authentication algorithm hat captures the assurance levels in identifying a VM user. In this capacity the policy engine authorization and decision making exhibits considerations as a clear LOA interaction agreement.
We seek to demonstrate the novel virtues of an AR mechanism from a relation-based AC mechanism – for a Global Virtual Machine Attribute Policy Auditor (GVMAPA), which includes a server engine called Policy Server (PS) and a policy management system. We scale this Policy Machine definition [7] [8] for reuse and extension within our GVMAPA. PS and GVMAPA together enable enforcement of multiple access control policies within a single, unified virtual machine system administered environment. *GVMAPA* composes and combines access control policies from a relative set of atomic properties completely expressed with mappings and interrelationships of the ARs on three basic elements – *Subject Sets, object Sets,* and *Operation Set.* Mappings and interrelationships of ARs are enforced with a database and a fixed set of functions.
The components of the GVMAPA Services architecture include: identity logging, monitoring, aggregation, requirement and registration. All these services are collaborative in nature, and demands considerable synchronization constraints as a measure of

the system's effectiveness. We'll speak to the component detail architecture of the GVMAPA as the subject of an independent paper.

This current paper however contains six sections. Section I introduces the AR of AC policies adopted for a VM. Section II explains AR Implementation mechanisms for the GVMAPA scheme. Section III introduces the architecture and functions of GVMAPA. Section IV specifies the GVMAPA mechanism for specifying ARs for AC models and policy constraints. Section V compares GVMAPA mechanisms with related work. Section VI is the conclusion and future work

2. AR IMPLEMENTATION DESIGN

GVMAPA provides an AC policy specification language as well as generic architecture components [Policy Decision Point (PDP), Policy Enforcement Point (PEP)] for the AC enforcement functions. The regular expressions specification constraints for this GVMAPA are adopted from work in [7, 8, 11].

(1) PS: T+ P + PCA + O
(2) *T: S + R + A + E*
(3) *P: T + RL +RCA + O*
(4) RL: T + C + E

where *PS* is the *Policy Set, T* is the VM *Target, P* is the VM *Policy, PCA* is the VM *Policy Combination Algorithm, O* is the *Obligation, S* is the *Subject, R* is the *Resource, A* is the *Action, E* is the *Environment, RL* is the *Rule, RCA* is the *Rule Combining Algorithm, C* is the *Condition,* and *E* is the *Effect* for the policy language scheme.

Regular expressions (2) and (4) are used for composing AC rules by the basic AC elements: subjects, resources, actions and environmental variables. Regular expressions (1) and (3) are for associating (2) and (4) in two different levels. There is no grammar for the expression of ARs in these four regular expressions unless specified by enumerating every relation between attributes. Additionally, we can let the Policy language allow functions to be implemented to handle ARs in PEP or an extended function. And those two methods are ad-hoc efforts without formal and structural definition in the scheme. In comparison, we will introduce an AC mechanism that provides a well-defined framework for the specification of attribute relations in Section III.

We demonstrate that GVMAPA has sufficiency of the elements in the language scheme for the purpose of explaining the ARs by the basic AC elements (i.e., subject, action, and object).

A. Specification of HRBAC Policies

BLPB policies require assigning **classes (**ranks) for VM attributes to subjects and objects. We also adopt from [7] the Formal definitions *Rs* = {…(Sa*i, Saj*)…..} and *Ro* ={…(Oa*i, Oaj*)…}, where *Rs* is a set of ARs for subject classes: for instance, *Sai* is the "Top Secret" class and *Saj* is the "Secret" class. *Rs* defines the "no read up" property of BLPB. In the same manner, *Oai* and *Oaj* define the object classes and property. Instead of classes, HRBAC model uses *Sai* and *Saj* to define the hierarchical relation of privilege inheritance from Role *Saj* to Role *Sai*; for example, Role "Professor" inherits all Role attributes of "Student" privileges in a grading system.
To specify and enforce these relations for our GVMAPA language, AC policy authors need to specify all the possibilities including direct and indirect relations between the classes of VM attributes. In the worst case, it requires O(n^2) number of (2) type of statements to describe the relations for *n* number of classes of attributes in the policy. Further, there is no clear semantic support for checking the correctness (e.g. cyclic assignment) of the specifications.

B. Specification of GVAMAPA Separation of Duty Policies (SOD)

When required to enforce SOD polices to prevent conflicts of interest or to control business processes, the access state of the AC system is dynamically dictated by some system variables. For example, a SOD policy constrains a subject's privileges (action and object pairs) not to exceed a predefined number, so that no subject should be assigned more than *k* privileges. Equally under this policy guarantees that no less than *k* number of subjects can perform all of a set of privileges (i.e., requires at least *k* number of subjects to perform all of them).
GVMAPA also seeks to enforce and maintain counters for monitoring the number of privileges consumed by each subject currently in the system.
In this context, the *obligation* and *environment* elements are used to update and retrieve (read in) the external counters, respectively.
In order to establish this however, we argue the need for a VM level of assurance (VM LOA) effecting attributes within a hierarchical structure (VMLOA-AHS) that can accommodate multiple attributes and categorize them into different VM groups along with their relationships. These abstractions demonstrate a composite effect by mapping the multiple attributes into a generic value.
The approach here is to establish a link attribute access control method which is risk averse. VM attribute properties are dynamic and hence we need to design and develop an adaptive authentication solution with different authentication methods and varying levels of attribute assurances. We consider a hierarchical based VMLOA-AHS policy combination algorithm that reflects these concerns. In the interest of space we'll discuss this algorithm as a part of our next paper.
The regular expressions (4) are needed for referencing the environment variables (e.g., external counters) and statements in (3) are used to store updated variables. However, the challenge is to accurately maintain the constraint variables (the number *k* in our examples), because a subject's access request can be granted from more than one type (4) statement. And (4) may be encompassed in (1) (2) or (3) statement, which provides no syntax for maintaining the ARs between (4) s. For example, a subject may be granted access both from Role *X* and Role *Y* to an object, and there is no way to specify the fact that *X* inherits *Y,* therefore, the privilege *k* for this subject is counted twice (which is supposed to be once) from both *X* and *Y* attributes in the same access session

3. ATTRIBUTE POLICY MACHINE

In pursuit of standardized access control mechanism for the virtual cloud environment , separation of access control policies from mechanisms which allow enforcement of multiple attribute policies within a single VM is critical for rule based policy audits for these abstract domains. Additionally considerations for certificate authorities through "Kerberos cloud provisions" should be integrated as apart of the policy audit. The GVMAPA architecture is composed of the *Policy Server* (PS) for PDP and PEP.

PS includes both processes and a database as components, and the General Policy Attribute Management System. The PS receives subject requests and performs the authorization process by referencing information from the PS database; it then generates a boolean value (grant or deny) as a result. The General Policy Attribute Management System is the interface for GVMAPA administrators to configure and compose policies and to manage the PS database. GVMAPA categorizes subjects (users), objects (resources), and their attributes into policy classes, and appropriately enforces subsets of the policies in response to a subject's access request.
The following fundamental data sets for the GVMA*PA* processing are stored in the *PS* database:

S: The set of GVMA*PA* subjects (users) under the *GVMAPA's* control
SA: The set of subject attributes of *S*
OP: The set of operations (access rights) permitted by the *GVMAPA.*
O: The set of objects under the GVMAPA*'s* control
OA: The set of object attributes of *O*
PC: The set of policy classes the GVMAPA is implementing

GVMAPA allows inheritance relations among subject attributes, and object attributes such that an element inherits the privileges from the elements that it is inherited from. The inheritance relation must not have cycles to be legitimate. A set of elements in an inheritance relation from one function to another function can be formally described by the union transitive closure of the two functions: ∗y ∈a(x)*b(y)* denoted by the symbol "x→a*b*". For example, all inherited subject attributes *SA*s of subjects can be denoted by *s*→ssa*sasa,* and all inherited object attributes *OA*o of an object *o* is *o*→ooa*oaoa*.

The atomic authorization process of *GVMAPA* is based on the above model and notation; the following formal definitions describe the *PS* authorization process:

For *s*∈ *S,* *op*∈ *OP,* *o*∈ *O,* *pc*∈ *PC, Grant_instance_of_policy(s,op,o,pc)* = True ™ ∈ *sa*∈ *SA* and ∈*oa*∈ *OA,* such that

1) sa∈ (s→ssa *sasa),* 2) *oa*∈ (o→ooa *oaoa),* 3) *sa*→op *oa, 4) pc* ∈ *sa*→sapc*pcpc,* and

5) *pc*∈ *oa*→oapcc*pcpc.*

GVMAPA only requires mapping the relations between elements to decide the permission of a subject's request. Through this mechanism, *GVMAPA* provides syntactic and semantic support of the AR specification.

4. ATTRIBUTE RELATION MODEL

This section outlines how *GVMAPA* specifies the HRBAC policies and Separation of duty (SOD) constraints by the AR assignments from the *PS* database and relation mapping functions. Subsection A outlines the implementation of a simple VM BLBP Model, and Subsection B shows the VM specification of SOD constraints as illustrated in Section II.

A. Specification of HRBAC Policies

GVMAPA can emulate its subject and object ARs. The subject security classes (labels) can be represented in GVMAPA's subject attributes. Further, the objects security classes (labels) can be represented in GVMAPA's object attributes and the subject attributes are linked to the object *attributes* through *operations.* For example, to implement the Bell-La Padula model, *GVMAPA may* construct two sets of relations for each of the *subject attributes* and *object attributes*.

The attribute with lower-case *r* in the attribute label of *subject attribute* and *object attribute* is for the read *privileges,* which are for the basic **confidential rule.** The attributes with lower-case *w* in the attribute label are for the **star** property of Bell- La Padula rules.

TS is *subject/object attribute* label for "Top Secret" *subject/object* class, *S* is for "Secret" class, and *C* is for "Confidential" class. *W* is for write privilege, *R* is for read privilege for each class (for example, *TSR* or *CW*).

Each subject/object belonging to a class is assigned to both labels w and r subject/object attribute w and r subject/object attribute (for example, *TSr* and *TSw*). Assume that class *TS* dominates class *S*, and class *S* dominates class *C*; Subjects with the *Cw subject attribute* can write objects with the *object attribute Cw*, *Sw* and *TSw*. *Sw* can write *Sw* and *TSw*. *TSw* can only write *TSw*. *TSr* can read *TSr, Sr*, and *Cr*. *Sr* can read *Sr and Cr.*
Cr* can only read *Cr.* Note that a *subject/object* must be assigned to the same *r* and *w* group of *subject/object attributes of (TS, S or C). For example, a subject should be assigned to the *Cw* subject attribute if she was assigned to the *Cr* subject attribute and vice versa.
Similar to BLBP models, the hierarchy of privilege inheritance for HRBAC can be directly specified by the *subject attributes* of *GVMAPA,* such that if *subject attribute x* dominates *subject attribute y,* then subject with role *x* inherits all the access privilege of subjects with role *y.* As the AR need only be assigned to directly related attributes, it only requires O(n) relation assignments if there are *n* classes for BLPA, or role inheritance relations for HRBAC. Thus the complexity is many times more efficient compared to assignment statements.

Note that in this paper, we are merely **specifying** the AR required AC models and constraints. The process complexity (efficiency) for the enforcement of these VM adopted attribute models and constraints are still the subject of ongoing work.

B. Specification of Separation of Duty (SOD) Policies

To enforce SOD, it is necessary to maintain all subject/object attribute relations for any subject or object if multiple attribute assignments are allowed. Hence, in addition to the basic relation mapping functions, to retrieve current mappings of ARs in the system, the function *sa_opoa(sa)* returns all (op, *oa)* pairs mapped to the *sa.*

For example, a SOD constraint specifies that no subject should be assigned to more than *k* VM privileges of a given set. Note that when *k* =1, this policy is a Privilege to Privilege Conflicts Policy (PPC), i.e. a set of privileges (OP ∈*OA)* should not be assigned to the same subject. *GVMAPA* implements this policy by calculating the number of subject attributes the requesting subject is dominating or inheriting associated with the constrained privileges and the number cannot exceed *k.*

These SOD constraints seek to eliminate duplicate VM privileges from the PS. Without these considerations, the complexity in specification is non trivial.

5. RELATED WORK

We like the authors in [7] [9] adopt NIST guidelines for a proposed Flexible Access Control Model (FACM), which provides user-friendly notations and presentation of ARs and constraints. However, the main usage of the graph representation is to help in the specification, design, rather than as a pure computational model, unlike GVMAPA, which provides computational functions in the PS server, and allows policy authors to specify AC rules by directly mapping ARs into rules semantic [10] proposed a Logical Framework for Reasoning about Access Control Models (ACM) based on the C-Datalog program, which provide a precise mathematical foundation for reasoning about ARs. However, in addition to its logical programs are not being intuitive to most users, ACMP does not provide views of access instance and relations between attributes, unlike *GVMAPA,* which is intent to allow administrators to check/filter the relations at the point of view of any selected access element. This capability otherwise requires tracing through AC rules, and it is hard to achieve with the increased number of entries in the Access control models program

6. CONCLUSION AND FUTURE WORK

The flexibility and expressiveness of an attribute based specification is intent on making virtual machine policy languages suitable AC mechanisms for these new compute cloud networks. We recognize that our work is still in its infancy because, *GVMAPA* is not an implementation language, and it is free from the syntactic and semantic complexity of such. When describing hierarchical relations between attributes or policies, *GVMAPA* only requires adding links between them, therefore, avoiding the time delays due to the sequence of overhead algorithms. In supporting the enforcement of SOD policy constraint rules, *GVMAPA* provides an infrastructure that allows the efficient specification of rules to collect the attributes for the VM policy.

Bearing in mind that the VM is merely a logical document for which we can keep logs on our physical disk. This feature is especially important when adding and deleting rules in the AC policies. Subsequent work focuses on using attribute policies highlighted in this paper to design synchronized virtual disk logs required by GVMAPA to establish forensic attribute consistency for all VM identities within the system environment.

7. REFERENCES

[1] Bell D.E. and Lapadula L. J., “Secure Computer Systems Mathematical Foundations and Model,” M74-244, MITRE Corp., Bedford, Mass., 1973 (also available as DTIC AS-77153

[2] Biba K. J., “Integrity Considerations for Secure Computer Systems,” ESD-TR-76-372, USAF Electronic Systems Division (also MTR3153, MITRE Corp.), Bedford, Mass.,1977 .

[3] NCSC, “Trusted Computer System Evaluation Criteria,” National Computer Security Center, 1985.

[4] Ferraiolo et al, “Role-Based Access Control (RBAC): Features and Motivations,” Proc. of the 11th Annual Conference on Computer Security Applications, Calif, pp 241-248, 1995.

[5] Jajodia et al, “A logical language for expressing authorizations,” Proc. IEEE Symp. On Research in Security and Privacy, Oakland, Calif, pp 31- 42, May 1997

[6] OASIS, “Extensible Access Control Markup Language(XACML), TC”, www.oasisopen.org/committees/tc_home.php/wg_abbrev=xacml

[7] Hu et al, “The Policy Machine For Security Policy Management,” Proc. ICCS Conference, San Francisco, 2001

[8] Ferraiolo et al, “Composing and Combining Policies under the PolicyMachine,” ACM SACMAT, 2005.

[9] Coetzee M. and Eloff J. H. P., “Virtual Enterprise Access Control Requirements,” Proc. of SAICSIT, pp. 285-294, 2003.

[10] Bertino et al, “A Logical Framework for Reasoning about Access Control Models,” ACM Transactions on Information and System Security, Vol. 6, No. 1, pp 71–127, February, 2003

[11] Grandison et.al “ Fomal Definition on Models of a Cloud Computing” “ IEEE Services , July 2010.

[12] Thorpe “ Contextual Models of trust for Digital Identities using UML 2.0”, IEEE Information Assurance and Security , August 2010.

[13] Ray et.al “ Global Virtual Machine Attribute Policy Auditor”, CSU Discussion Forum, Sept. 2010.