Towards Multi-policy Support for IaaS Clouds to Secure Data Sharing

Ying Fairweather
Computer Science Department
New Mexico Institute of Mining and Technology
Socorro, NM USA
ywa01@nmt.edu

Dongwan Shin
Computer Science Department
New Mexico Institute of Mining and Technology
Socorro, NM USA
doshin@nmt.edu

***Abstract*—Infrastructure as a service (IaaS) is a cloud service model that provides storage and computation services for users at a low price. A recent report from Gartner indicates that IaaS will be the fastest growing area among all of the cloud service models in the near future, and thus it is strongly envisioned that multiple companies will use IaaS clouds to share information among them. However, the current access control mechanisms in IaaS platforms do not have the ability to enable flexible data sharing among companies while addressing security problems such as information and privacy leaking. In this paper, we propose two IaaS cloud reference architectures that enforce cloud-level Chinese Wall security (CWS) policy to prevent information leaking among companies. The new architectures are also able to support customized domain level access control policies such as role-based access control (RBAC), privacy-preserving information retrieval, and single sign on (SSO). The reference architectures were implemented using Eucalyptus and its data storage service called Walrus; therefore, our approach can also be applied to commercial clouds like Amazon S3. The result of performance analysis has shown that our architectures are feasible, scalable, and efficient.**

Keywords-Infrastructure as a service (IaaS); Access Control; Chinese Wall Security Policy; Identity Management; Secure Information Retrieval

I. INTRODUCTION

Cloud computing promotes the availability of computing resources, which can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. Infrastructure as a Service (IaaS) is a service model of cloud computing that provides users with infrastructure services such as computation and data storage. In 2010, a research report from Cisco forcasted that the service revenues from IaaS would be $15.6 billion in 2013, out of $35.4 billion from all types of cloud services [2]. A recent report from Gartner confirmed the growing interest and importance of IaaS with a confidence that IaaS would be the fastest growing cloud service model in 3 years [3].

IaaS is an ideal solution for small and medium sized businesses and companies, considering that the cost of cloud storage is much lower compared to the cost of purchasing physical storage. Hence, those companies have shown keen interest in storing their resources in clouds and it is strongly envisioned that they will use IaaS clouds to share information among them as well. When multiple companies are involved in sharing data for collaboration, simple access control mechanisms like Access Control Lists (ACLs) are not sufficient. Various issues like conflict of interest (COI) among companies and the user privacy and identity control in a cross-domain environment should also be taken into consideration when access control services are designed for IaaS clouds. However, popular IaaS providers like Amazon and Microsoft currently fail to provide such mechanisms [4][5][6]. Therefore, there is an urgent need for a more sophisticated IaaS authorization model that allows for secure information sharing among companies.

In this paper, we propose two IaaS cloud reference architectures that aim at a higher security level in the cloud. We propose to employ the concept of domain and Chinese Wall Security Policy (CWSP) to the architectures and a popular open-source IaaS cloud platform, Eucalyptus [7], is extended to implement those two architectures. Our main goals include flexible and secure information sharing between companies, protecting users' privacy, and reducing cloud/database administrator' privilege to mitigate admin-based insider attacks.

A. Motivation Examples

To demonstrate that our approach is applicable to real-world problems, we will start by describing two scenarios.

Scenario 1: Alice is the owner of a financial consulting company, BestFinance. In BestFinance there are 1000 consultants working for 10000 companies. Each company has an individual physical machine that stores a tremendous amount of data, some of which are top secrets. Due to CWSP, one consultant cannot work for two companies that have a conflict of interest (COI). To enforce this policy, a system administrator is hired to manage the access rights of consultants. Customer companies' COI properties and a track of consultants' access history are stored in the database.

When a consultant requests information of a customer company, the system administrator retrieves the company's COI property and the consultant's access history from the database. He will grant access to the consultant if there is no violation of CWSP. The system administrator will also update the consultant's access history in the database immediately.

This work was partially supported at the Secure Computing Laboratory at New Mexico Tech by the grant from the National Science Foundation (NSF-

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254127

As the number of customers increases, the maintenance of the machines is becoming more costly. Alice also heard about admin-based insider attacks, when administrators take advantage of their privileges and leak data to competitor companies. Some customers suggested having their own policy in which they can further control consultants' access of their files. So Alice is looking for a solution that provides lower cost, mandatory access control based on COI, flexible company-based policies, as well as higher security against insider attacks.

Scenario 2: Bob is a cloud service provider for millions of companies from all over the world. It is unavoidable to store several companies' files on the same physical machine. He has a good size of cloud administrators doing periodic maintenance, which involves accessing files of each customer company. To avoid possible information leaking caused by administrators, he assigns each administrator some companies that do not have a conflict of interest, and gives them access to the specific companies' data.

At the same time, some companies want to share information with users' from partner companies, but they do not want to create a user account for each company. He is looking for a solution that automatically prevents information leaking caused by cloud administrators and supports flexible information sharing between companies in the cloud.

We believe that what Alice and Bob need is a cloud service model which protects users from admin-based insider attacks and supports flexible information sharing between companies/users while maintaining the necessary security level by applying cloud and company level access control policies.

B. Contributions and Organizations

In this paper, we propose two feasible IaaS cloud reference architectures, one with centralized identity management and the other with decentralized identity management. The CWSP is enforced on both architectures to support secure information sharing at a company-to-company level. The new architectures feature company domains in the cloud, flexible company-defined access control on the domain level, privacy-preserving information retrieval, Single Sign On (SSO), user-friendly web interfaces, and protection against admin-based insider attacks.

Our contributions in this paper are as follows: first, we combined cloud level CWSP and company level policies like RBAC. This provides a fine-grained access control mechanism that is more applicable for information sharing among multiple companies. To the best of our knowledge it is the first attempt to combine these two types of policies in an IaaS model. Second, by managing the user identities with SSO and using a privacy-preserving component, we protect the user's sensitive information from leaking to untrusted parties and database admins. Fourth, by having CWSP based access control and the privacy-preserving component, we reduced admins' privileges thus mitigating the risk of an admin-based insider attacks.

The rest of the paper is organized as follows. Section 2 discusses background and related work. Section 3 describes the features and design of our approach, including the architectures of two models and their workflow. Section 4 presents the implementation of the two reference architectures using Eucalyptus and analyzes the performance of the implemented architectures. Section 5 concludes the paper.

II. BACKGROUND AND RELATED WORK

A. Eucalyptus and Access Control in IaaS

Eucalyptus is a cloud platform that is equivalent to Amazon's commercial cloud services such as Elastic Compute Cloud (EC2) and Simple Storage Service (S3) [7].

In this paper we used an open-source version of Eucalyptus for a proof-of-concept implementation and extended for our purposes. There are five major components in Eucalyptus:

- Cloud Controller (CLC): The entry-point of the Eucalyptus cloud. It handles user request from an Amazon EC2 compatible command line tool, as well as a web interface. The original web interface is mostly meant for administration purposes. Our modification is mostly made on this component.
- Walrus: The component where user data are persistently stored. In this paper we are interested in the data stored in Walrus, other than computation resources like Virtual Machines (VMs). This component is equivalent to Amazon S3.
- Cluster Controller (CC): Responsible for computing nodes scheduling and network control within the cluster.
- Storage Controller (SC): Controls blocked-based storage, equivalent to Amazon ESB.
- Node Controller (NC): Controls VM related activities.

Role-based access control (RBAC) is one of the most popular access control policies for computer systems. Shin et al. discussed how to inject RBAC supports into IaaS clouds and used Eucalyptus as a proof-of-concept implementation example for their approach called dCloud. The dCloud supports the concept of domains, enabling the easy establishment of private IaaS clouds within a public cloud such as Amazon EC2 [8]. In dClound, each of the domains is able to manage its own security policy including RBAC. Our approach is based on and extends dCloud. Tsai also proposed to apply RBAC in the cloud platforms but with emphasis on how to efficiently build the role hierarchy [9].

Discretionary access control policies are also proposed for cloud platforms, and they require the data owner to either attach an ACL for each file [10] or define an access structure for each user [11]. Most of the popular IaaS providers like Amazon, Google, and Microsoft use ACLs. Google also supports a signed URL, which provides "valet-key" type access for anonymous users. Amazon and Microsoft allow users to control the access right based on groups, but none of them provides a finer-grained access control mechanism that takes COI into consideration. In addition, all the existing implementations apply a single access control mechanism at a cloud level or a domain level, but not both.

B. Chinese Wall Security Polices

The Chinese Wall Security Policy was first introduced in 1989 [12]. The mathematical foundation and an enforcement mechanism were proposed for this well-known commercial security model that has attracted a lot of interest from computer security communities thereafter. Later, Lin argued that the notion of COI classes by Brewer and Nash is only applicable to the very specific circumstance (for instance, when A has a conflict of interest with both B and C, B and C must have a conflict of interest), claiming it to be conservative when it comes to generalized applications [13]. Therefore, he proposed an aggressive CWSP model that can be applied to general applications, where the COI relationship is not transitive among entities. Subsequently, Sandhu proposed that one object should be able to be associated with different datasets within different COI classes [14][15]. The generalization of the original Brewer-Nash model was also discussed in [16].

In recent years, the interest of the CWSP has been drawn to the cloud computing environments. A centralized control mechanism based on the CWSP was proposed to eliminate inter-VM attacks in [17]. In their approach, they used the CWSP and graph theory to achieve the physical isolation of VMs owned by companies that have a conflict of interest. Similarly, [18] enforced the CWSP on IaaS to ensure that a user cannot run two VMs that are in different COI classes. The CWSP has also been applied to software as a service (SaaS) and platform as a service (PaaS) cloud models in [19] and [20]. None of the above targeted the data storage in IaaS, which we believe is more interesting to attackers.

C. Secure Information Retrieval

Users' sensitive data such as credentials and access history must be protected. This type of information is usually stored in a database as plaintext. CryptDB is a database proxy developed by MIT CSAIL [21]. It takes plain SQL queries from users and uses several encryption algorithms to encrypt data and store them in the database. When CryptDB receives a SQL query, it retrieves the encrypted data and sends back the decrypted data. In this way, a database administrator cannot get any useful data by looking at the tables.

In our approach we do not need complex SQL queries, so a simplified secure information retrieval module is used. Since the information retrieval is also done in the user's trusted party, an untrusted third party can never obtain the sensitive information. The encryption of data in the cloud was also discussed in [22] and [23], but since we already have an access control mechanism on the data, the encryption is trivial.

D. Insider Attacks

An insider attack happens when users with privileges in the system decide to exploit the privileges and perform malicious tasks. Among all types of insider attacks, admin-based insider attacks are considered the worst since they exploit the administrative privileges. Several methods were proposed to detect an insider attack [24] [25], but reducing the probability of such an attack is more effective. Bleikertz el al. introduced a solution to prevent admin-based insider attacks during maintenance by assigning them different levels of privileges and using separation-of-duty [26].

In our approach, we address this type of attack using a CWSP that controls administrator's access activity and a secure information retrieval module that reduces administrator's privileges on the database. In reality there could be many different types of admin-based attacks. Hence, we limit our solution for a specific type, which is caused by exploiting the administrative privileges to access data in the cloud storage.

III. Our Approach

In order to support secure data sharing in clouds, we propose two IaaS cloud reference architectures, centralized and decentralized ones, in this section.

A. Domains and Identity Management

Our new architectures adopt the concept of domain from dCloud [8]. Each company in the cloud has its own domain. Every domain has an interface to provide access to the data that belong to the domain and also a database to store the information about the resources and policies of the domain. For the ease of database management, the database should be hosted by each company directly and is accessible by the domain's interface.

In the rest of the paper, we will use the following definitions:

- **Home domain**: the domain to which the user belongs.
- **Identity provider**: the domain/cloud where the user's identity is stored. In a centralized architecture, this is the cloud, and in a decentralized architecture, this is the same as the user's home domain.
- **Service provider domain**: a domain from which the user wants to access data.
- **Collaboration domain**: a domain with which the service provider domain wishes to share information.

When a user of domain A wants to access data located at the service provider domain B, the domain B will have to verify the user's identity first. It is by no means necessary or safe for the user to reveal his/her credentials to a third party. Alternatively, the user can register a new account with all of the service provider domains. This solution works well when the number of service provider domains in the cloud is small, but will cause many management and security issues when the number of service provider domains rapidly grows. In order to address this concern, we adopted a SSO solution based on the existing trust relationships in the cloud:

- The users trust the cloud. Their credentials and access history are stored in the cloud. This is a common trust relationship for cloud platforms such as Amazon or Google.
- The users trust their home domains. Their credentials and access history are stored in the home domain.

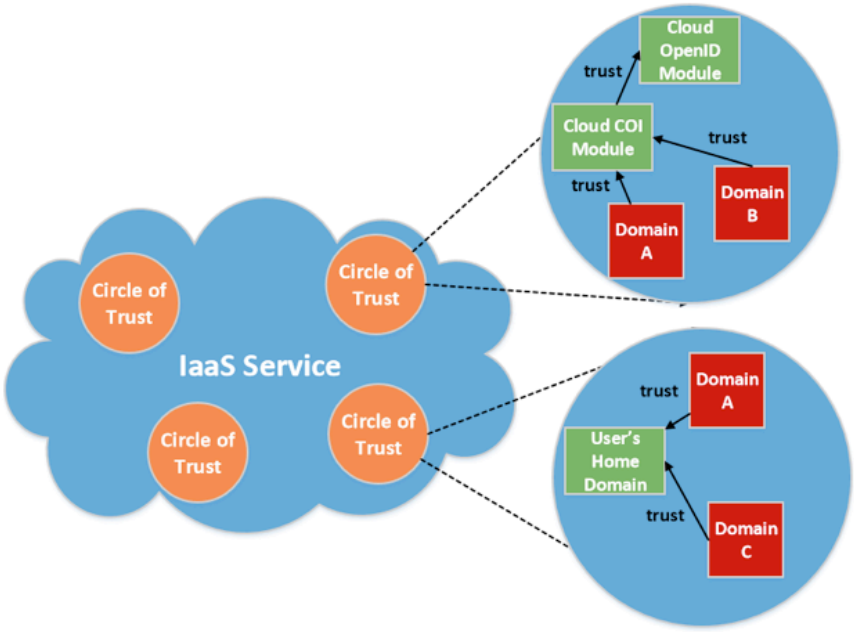


Figure 1. Circle of trust

- The service provider domains trust the cloud. They are storing their confidential data in the cloud. This is also a common trust relationship witnessed often in the federated identity management (FIM) solutions.
- The service provider domain trusts the collaboration domains. They are sharing their confidential data with the collaboration domains.

Based on the four trust relationships above, we can infer that a user is willing to appeal their credentials to the cloud and home domain, and a service provider domain trusts the authentication and access control decisions made by the cloud or a collaboration domain. The user's home domain, and all the service provider domains that treat this home domain as a collaboration domain form a circle of trust.

Fig. 1 shows the circles of trust in an IaaS service cloud. The expanded circle on top shows the centralized architecture. The cloud is trusted by all the domains inside the cloud. We designed a COI module as the CWSP enforcement mechanism and an openID module for authentication. The bottom circle is in the decentralized architecture. The user's home domain, which takes care of both the CWSP enforcement and authentication, is trusted by the service provider domains.

In our approach, users only need to authenticate themselves to the cloud or their home domain. After the authentication and CWSP related access control, a token will be issued for the user and sent to the service provider domain. The token contains essential information about the request, and is signed by the sender to guarantee its integrity and non-repudiation. Upon successful verification of the token, the service provider domain will display available resources to the user.

Besides credentials, there are also other sensitive information that a user may not want to share, i.e., the user's access history. The cloud COI module and the home domain's web UI also work as a privacy-preserving component that supports secure information retrieval. All the information related to users will be encrypted and stored in the database. The component only displays the decisions made based on that information, rather than exposing it directly to a third party. In this case, no admins or third parties can have access to the actual information, thus the users' privacy is protected.

B. Access Control and Security

On a normal IaaS platform, the data owner defines the access control policies on the data and grants the access to individuals and groups. This traditional access control mechanism is suitable for the one-to-one and one-to-N type information sharing, but is not scalable.

As the scale of information sharing grows, a potential problem arises when some datasets conflict with each other in interest. For instance, data owner A may be sharing data with consultant B, without knowing that her competitor C is also sharing data with B. When B has access to data from both sides, a conflict of interest happens. Since the traditional IaaS model lacks protection of such situations, users have to be worried about possible information leaking.

In our new architectures, companies are still allowed to have controls on their own data. But to protect against information leaking, we need to add an upper lever access control mechanism that monitors the access activities in the whole cloud. This finer-grained access control mechanism leverages the CWSP model proposed by Brewer and Nash [12].

By adopting the domain concept, the users and data in the cloud are already grouped into domains based on the company they belong to. To apply the CWSP, we need to further categorize these domains into COI classes based on the company's functionality. Companies that make profits in the same area will fall into the same COI class, like Bank of America and Wells Fargo.

A user in the cloud can be represented as a tuple U = [O, C, H, R], and a data resource can be represented as D = [O, C, T] where:

- O is the home domain of the user/data.
- C is the COI attribute of the user/data. This attribute is inherited from O.
- H = $\{O_1, C_1, O_2, \ldots\}$ is the access history of U. It's a finite set of Os and Cs.
- R = $\{R_1, R_2, \ldots\}$ is a finite set of roles that are assigned to U, where each R_i is a set of permissions associated with the role: $\{P_1, P_2, P_3, \ldots\}$. To simplify the concept, we will treat R as a N x M matrix that represents the user's access right on a finite set of data.
- T = $\{O_1, O_2, \ldots\}$ is a finite set of trusted domains for D.
- $U \rightarrow D$ would mean that U could access D.

Since we use two levels of access control, the cloud level CWSP mechanism can only make decisions about whether the user should be able to connect to the service provider domain's interface with a generated token. The service provider domain will not recognize any token that came from a domain not included in the collaboration domain set, or a token that was tampered with.

DEFINITION 1 (Circle of Trust): To access the data, the user has to belong to a domain that is trusted by the data resource's home domain.

$$O_U \in T_D \tag{1}$$

Brewer and Nash's rule claims that this access is only granted if [12]:

- The data requested is in the same company dataset as another data object that has been accessed by the user before.

Or

- The data requested belongs to a COI class in which none of the domains has been accessed by the user before.

DEFINITION 2 (Chinese Wall Security Policy): To access the data, the user's access history must satisfy Brewer and Nash's rule.

$$O_D \in H_U \,||\, C_D \notin H_U \tag{2}$$

After the user passes the CWSP access control mechanism, they will be able to access the data if they have the access permission. The domain level access control policies are managed by the home domain of the data. We use one of the popular access control policies, RBAC, as the domain level policy.

DEFINITION 3 (RBAC): To access the data, the user has to obtain the access permission on the data first. We can denote the user's access permission on the data D, which is an entry in a matrix that contains all data resources, as: $R_U \times D$.

$$R_U \times D = 1 \tag{3}$$

In conclusion, a user U can access a data resource D if and only if all the three formulas above are satisfied.

$$U \rightarrow D \text{ iff } (O_U \in T_D) \,\&\, (O_D \in H_U \,||\, C_D \notin H_U) \,\&\, (R_U \times D = 1) \tag{4}$$

In the centralized architecture, the COI information of each company is managed by cloud admins. While in the decentralized architecture, such information is managed by domain admins from each domain that serves as an Identity Provider. The classification of companies into COI classes is based on what area a company makes profits in and what companies it has a conflict of interest with, and should be agreed among all the companies in the cloud. The domain level access control policies are managed by domain admins from each domain that serves as a Service Provider.

In addition, cloud admins are also grouped into the default "cloud" domain. It is not a domain that is associated with a company, but just a virtual domain that does not conflict with any companies in the cloud. We also gain the benefit of tracking and controlling admins' access activities.

By employing the CWSP and the secure information retrieval component, the admins' privileges are reduced, so is the risk of an admin-based insider attack.

C. System Architecture and Work Flow

Fig. 2 shows the system architecture. Both architectures have a database that stores the user's access history, an authentication module to verify the user's identity, and a COI module to check the user's access history and generate COI tokens for the user. The only difference is, in the centralized architecture, the cloud is the only party that can work as an identity provider (only one circle of trust); in the decentralized architecture, every home domain is an identity provider (multiple circles of trust).

We also modified the authentication module in the centralized architecture to an openID-supported module within the cloud. This module can be easily replaced with popular openID providers such as Google and Microsoft. The database where the user credentials are stored and the modules inside the service provider domain are not shown in the figure.

Here we take Alice's consulting company as an example, the typical workflow described in Fig. 2 is:

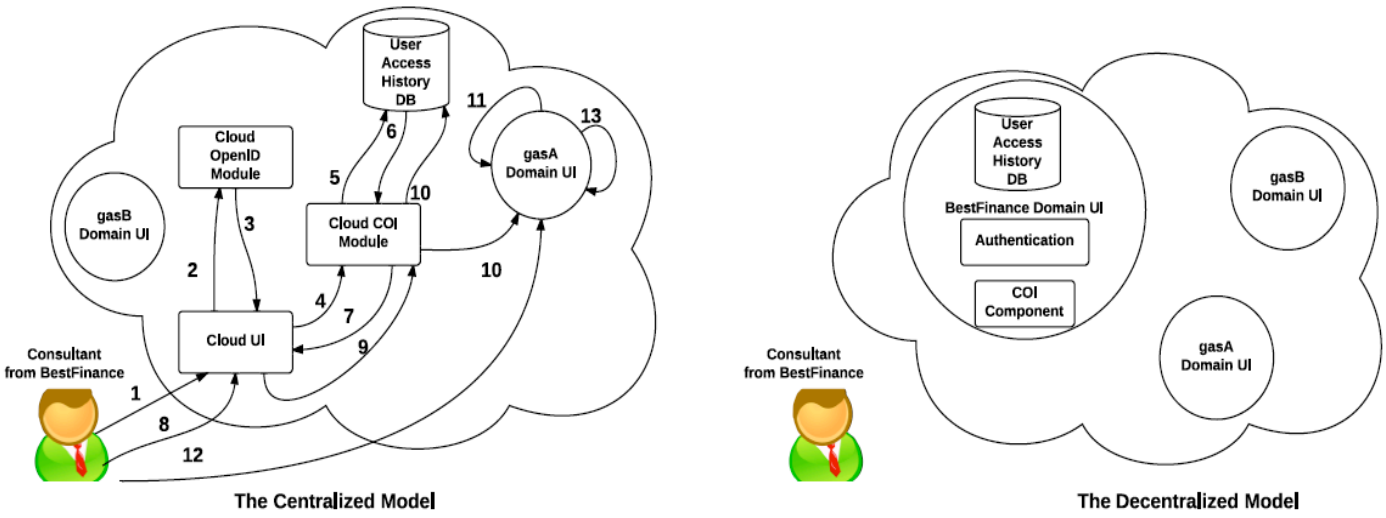


Figure 2. Architecture of the two IaaS cloud reference architectures (left: centralized, right: decentralized). The centralized architecture also shows the work flow when a user tries to access data from gasA domain.

1) The user connects to the cloud web UI.

2) The authentication is handled by the cloud openID component.

3) The openID component sends back the result.

4) Upon successful authentication, the cloud UI consults the cloud COI component about what domains this user should have access permissions to.

5) The COI component queries the database.

6) The encrypted user data is sent back from the database.

7) Based on the access history, the COI component sends back a list of domains that the user should have access permissions to. These domains are displayed to the user.

8) The user browses the list and picks the domain they want to access.

9) The cloud UI sends the information about the request to the COI component.

10) The COI component generates a signed token for the user and updates the user access history in the database. The user is redirected to the service provider domain.

11) The service provider domain verifies the token that came with the user. If the token is from a trusted domain, is original, and hasn't expired, the service provider domain UI will display a list of the data resources in the domain.

12) The user browses the resource list and picks the data resource they want to access.

13) The service provider domain UI checks the user's permissions. If the user has access permissions on the requested data resource, it will be returned to the user.

In the decentralized architecture, the cloud UI is replaced by the home domain UI, and a normal authentication component takes the place of the openID module.

IV. System Implementation And Performance Analysis

We implemented two IaaS cloud reference architectures that have the functionalities described in the last section using Eucalyptus and conducted performance analysis to check the feasibility of the architectures. The analysis proved that our approach is efficient, feasible, and scalable.

A. *Implementation*

We implemented our approach using Eucalyptus 1.6.2. In our centralized architecture, all users belong to the cloud. Therefore, they need to go through the Eucalyptus web UI to access data from domains. The user information needed for authentication and access control is also stored in the cloud. We provide normal users and admins with different functionality. The interface for normal users contains only the basic information that a user needs. A user can browse the service provider domains that are open to them and choose one to access, as shown in Fig. 4. The admin interface enables more complex tasks such as managing domains and users, as shown in Fig. 3.

In the decentralized architecture, the users are assigned to the domains they actually belong to. The domain web UI will only take care of the users within the domain. There is no modification to the original Eucalyptus cloud. The web UIs for home domains are standalone web applications that were written in Java. Due to the similarities of the two UIs, we are not showing the UI for the home domain in this paper.

We also built Java based web UIs for the service provider domains. Instead of the web UIs hosted directly in the cloud, VM images were made for each UI. To enable the interface of a service provider domain, we simply run a VM instance with the specific VM image that has the required web UI. In this way, the shut down and removal of a web UI becomes easy.

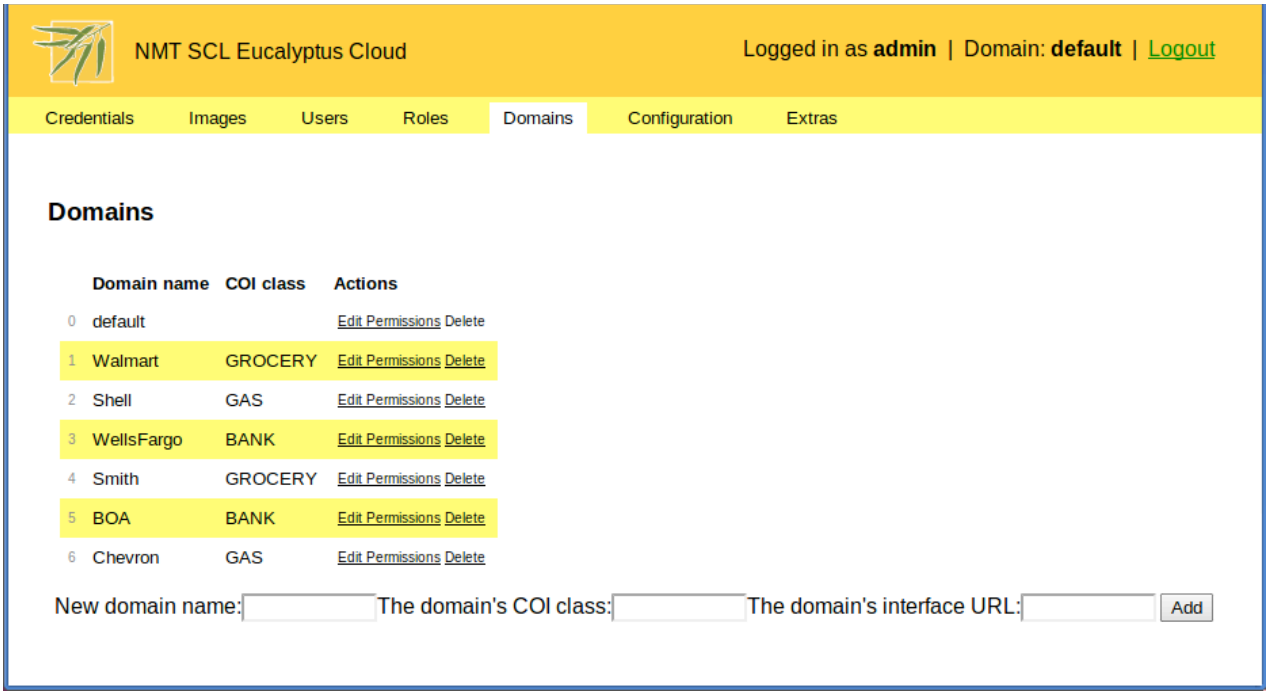


Figure 3. The modified Eucalyptus web UI in the centralized architecture. The logged in user is a cloud admin.

We used MySQL database to store some of the user information and all of the domain level policies. The sensitive user information is digested by SHA-256 algorithm. In the centralized architecture, we modified the table structure in Hibernate to store the user information. A COI token generated by home domains contains the user's username, the home domain, the service provider domain, the user's IP and the expiration time. We used a 1024-bit RSA key pair. The information contained by the token is digested using SHA-256 algorithm, signed by the home domain's private key, and then appended to the token.

The domain level access control we used is RBAC. Since RBAC is not our focus in this project, we didn't use complicated role structures. Within each service provider domain, a user has only one role, which has permissions on a few data resources. When our architecture is applied to the real world, it is totally the company's freedom to choose whichever access control policy benefits them the most.

The openID module in the centralized architecture is only a demonstration about how openID protocol can be applied to our architecture [33]. The openID component is currently a module inside the cloud, it can also be replaced with the openID services provided by industry players. In the decentralized architecture, we only used a traditional username/password authentication method.

B. Testbed Setup

For the testbed setup, we used 2 physical machines and 6 virtual machines. The two physical machines have the following configurations: Intel Pentium 4 3 GHz, 76GB storage, 4GB RAM and 100 Mbit Ethernet. One of them is used to run the CLC (which includes the web UI), Walrus, SC, CC components of Eucalyptus, the other is hosting the decentralized home domain web UI for domain IdP1. The 6 virtual machines are managed by Virtual Box [34]. Each of the virtual machines has the configuration of 50GB storage, 2GB RAM and 100 Mbit Ethernet.

To test the feasibility of our architectures, we created 6 service provider domains belonging to 3 COI classes: Bank Of America domain and Wells Fargo domain in the BANK COI class, Shell domain and Chevron domain in the GAS COI class, and Smith's domain and Walmart domain in the GROCERY COI class. Each domain's web UI is hosted on a separate VM.

We also created 6 users that each belongs to one of the above domains. If the CWSP mechanism works correctly, a new user should be able to see all the other domains that does not conflict with his/her home domain in the "Available Domains" tab; and whenever the user chooses to access a domain that belongs to a COI class he/she never accessed before, the available domains will be updated next time he goes to the home domain web UI. We expect all the domains that have a conflict of interest with the user's home domain or the domains in the user's access history to disappear from the "Available Domains" tab.

In the test, we assigned each of the users a different set of domains that he/she would visit and observed the change of their available domains. From Fig. 3, we can see that there are six service provider domains in the cloud. In Fig. 4, the left side shows when the user "test6" first accessed the Eucalyptus web UI, the "Available Domains" tab displayed all the domains expect for Chevron, which has a conflict of interest with the user's home domain, Shell. We then made this user access the

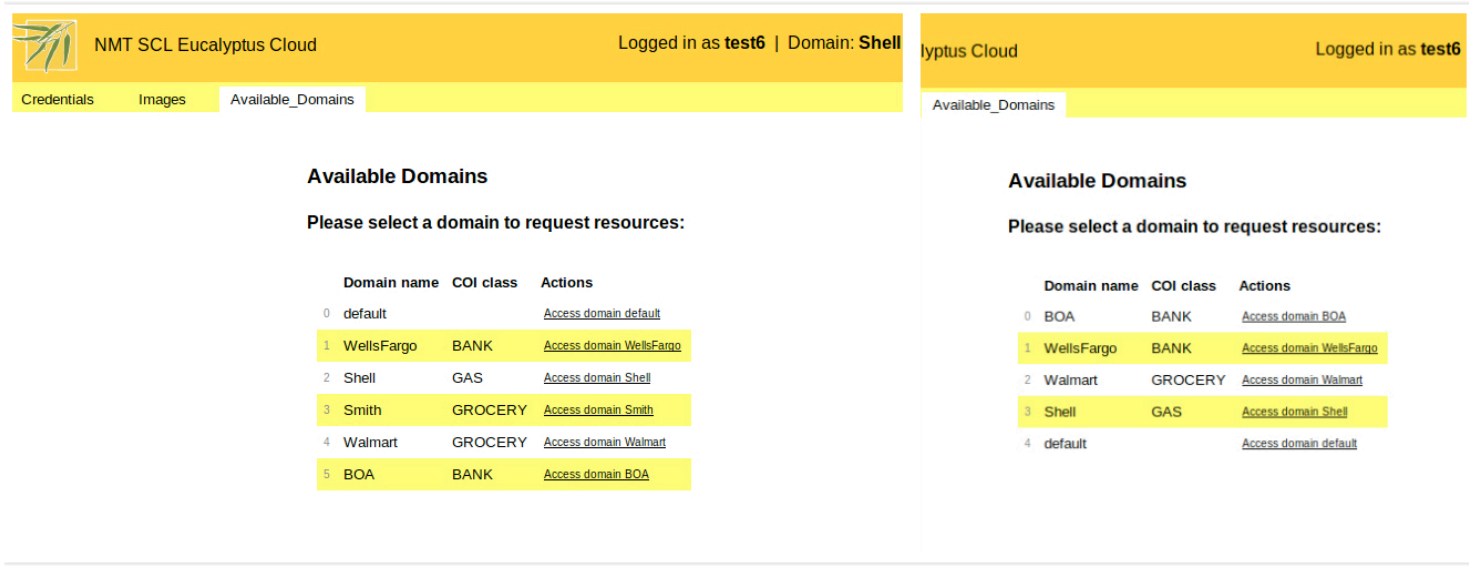


Figure 4. The left side is the modified Eucalyptus web UI in the centralized architecture when the logged in user is a normal cloud user; The right side is the updated "Available Domains" tab after the user accessed the Walmart domain.

domain Walmart. When the user came back to the Eucalyptus web UI, the domain Smith, which has a conflict of interest with the domain Walmart, disappeared.

The results of the test showed that both of the architectures behave correctly when making decisions based on the CWSP.

C. *Efficiency and Scalability*

To calculate the overhead introduced by the new CWSP mechanism, we compared the times taken over three different architectures (the original Eucalyptus terminal commands, the centralized architecture, and the decentralized architecture) when the number of user increases from 1 to 10 to 100 and to 1000. We also broke the total processing time taken by each architecture into four important parts: authentication, COI checking, COI token generation and token validation. In this way, we are comparing the time spent on the server side of each architecture only, without worrying about the time difference introduced by users' typing speed and response time.

One of the 6 VMs was used to host the web UI of a service provider domain, and the rest were used to generate concurrent user requests. We used a Java program and the developer's tool on Chrome to measure the time for the whole operation.

As the number of user increases, the overall time of all the three modules did not show dramatic changes. In the centralized architecture, the time taken for COI checking was slightly longer, due to the busy traffic for the database. But the overall operation time when there were 1000 users was only about 300ms longer than when there was only one user. This shows that both of our architectures are scalable.

Fig. 5 shows the time taken by each of the four parts in three architectures. The original Eucalyptus only has the authentication part, and the decentralized architecture combines the authentication and COI checking in one step. We can see from the chart that most of the time taken in the centralized architecture was actually due to other operations like loading images and components of the web page. Neither of our architectures introduced a big overhead.

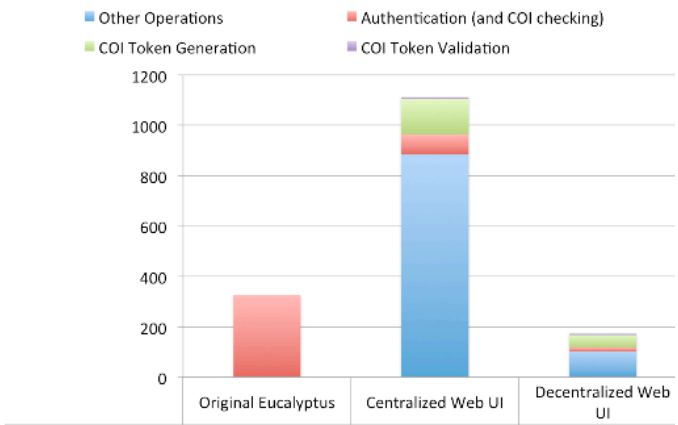
V. DISSCUSION

Due to the limitation of time and hardware, the largest amount of concurrent requests we tested was 1000. In the future we would like to increase the number of concurrent requests to find out the bottleneck of the new architectures, and optimize them based on the results.

VI. CONCLUSION

The increasing popularity of cloud computing has brought new challenges to the existing IaaS models. While the current models are ideal for a small scale of information sharing, they fail to provide the finer-grained access control mechanism that can take care of the issues brought by large-scale information sharing among companies. Conflict of interest is one of the issues that can cause this problem.

In this paper, we proposed two IaaS cloud reference architectures that can address this issue by applying the Chinese Wall Security Policy at the cloud level. The new architectures also feature the notion of domains in the cloud, flexible company-defined access control on the domain level, privacy-preserving information retrieval, Single Sign On, and protection against admin-based insider attacks. We built a proof-of-concept implementation and conducted a small scale performance testing. The testing results showed that our approach is feasible, efficient and scalable.



	Original Eucalyptus	Centralized Web UI	Decentralized Web UI
COI Token Validation		7	6.5
COI Token Generation		142.25	49.25
Authentication (and COI checking)	325.35	78.75	15.25
Other Operations		884.5	102.75

Figure 5. The time taken by each major components in the original Eucalyptus and both architectures that we implemented. The unit is Millisecond.

REFERENCES

[1] NIST, "NIST working definition of cloud computing," unpublished.

[2] Cisco, "Infrustracture as a Service: accelarating time to profitable new revenue streams, " unpublished.

[3] Gartner, "Forecast overview: public cloud services, worldwide, 2011-2016, 4Q12 update," unpublished.

[4] Amazon EC2 and S3. http://aws.amazon.com/

[5] Google Drive. http://drive.google.com

[6] Microsoft Skydrive. http://skydrive.live.com

[7] Eucalyptus Open Source. http://www.eucalyptus.com/

[8] D. Shin, H. Akkan and W. Claycomb, "Towards role-based provisioning and access control for Infrastructure as a Service (IaaS)," In *Proceeding of TrustCol 2010*, 2010.

[9] W. Tsai, "Role-based access-control using reference ontology in clouds," in *Proceedings of ISADS '11*, 2011.

[10] S. Sanka, C. Hota and M. Rajarajan, "Secure data access in cloud computing," In *Proceedings of IMSAA'10*, 2010.

[11] S. Yu, C. Wang, K. Ren and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," In *Proceedings of INFOCOM'10*, 2010.

[12] D.F.C. Brewer and M.J. Nash, "The Chinese Wall security policy," in *Proceedings of 1989 IEEE Symposium on Security and Privacy*, 1989, pp. 206-214.

[13] T.Y. Lin, "Chinese wall security policy-an aggressive model," in *Computer Security Applications Conference*, 1989, pp.282-289.

[14] R.S. Sandhu, "Lattice-based enforcement of chinese walls," *Computers & Security*, 1992, pp.753–763.

[15] R.S. Sandhu. A lattice interpretation of the chinese wall policy," in *Proceedings of the 15th National Computer Security Conference, NISSC*, 1992, pp. 221–235.

[16] V. Kessler, Jr, "On the Chinese Wall model," in *the Second European Symposium on Research in Computer Security (ESORICS 92)*, 1992.

[17] T. Tsai, Y. Chen, H. Huang, P. Huang and K. Chou, "A practical Chinese wall security model in cloud computing," in *Network Operations and Management Symposium (APNOMS)*, 2011, pp. 21-23.

[18] R. Wu, G. Ahn, H. Hu and M. Singhal, "Information flow control in cloud computing," in *6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2010, pp. 9-12.

[19] V. Gupta, "Chinese Wall security policy," M.S. Thesis, San Jose State University, San Jose, CA, 2009.

[20] Y. Hsiao and G. Hwang, "Implementing the Chinese Wall security model in workflow management systems," in *2010 International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2010, pp. 574-581.

[21] R.A. Popa, C.M.S. Redfield, N. Zeldovich and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *Proceedings of SOSP '11*, 2011.

[22] S. Sanka, C. Hota and M. Rajarajan, "Secure data access in cloud computing," in *Proceedings of IMSAA'10*, 2010.

[23] S. Yu, C. Wang, K. Ren and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of INFOCOM'10*, 2010.

[24] V.B. Velpula and D. Gudipudi, "Behavior-anomaly-based system for detecting insider attacks and data mining," in *International Journal of Recent Trends in Engineering,* 2009.

[25] A.H. Phyo and S.M. Furnell, "A detection-oriented classification of insider IT misuse," in *Proceedings of the Third Security Conference,* 2004.

[26] S. Bleikertz, A. Kurmus, Z. Nagy and M. Schunter, "Secure cloud maintenance- protecting workloads against insider attacks," in *Proceedings of ASIACCS 2012*, 2012.

[27] OpenID. http://www.openid.net

[28] Virtual Box. https://www.virtualbox.org/