Digital Ecosystem Access Control Management

ILUNG PRANATA AND GEOFF SKINNER
Faculty of Science and information Technology
University of Newcastle
University Drive, Callaghan, NSW, 2308
AUSTRALIA
Geoff.Skinner@newcastle.edu.au , ilung.pranata@studentmail.newcastle.edu.au

Abstract: - The newly emerging concept of Digital Ecosystem (DE) has played a significant role in today's technology, especially for Small and Medium Enterprises (SMEs) to adopt Information and Communication Technology (ICT) inside their businesses. DE reveals the opportunities to enhance the productivity and efficiency of each business transaction. Therefore, it will further contribute to the success of the enterprise's businesses. Along with the advancement of DE technology, security has emerged as a vital element in protecting the resources and information for the interacting DE member entities in particular. However, current developments of such security mechanisms for protecting these resources are still in their infancy. This paper proposes a distributed mechanism for individual enterprises to manage their own authorization processes and information access permissions with the aim of providing rigorous protection of enterprise resources.

Key-Words: - information management, authorization, authentication, access permissions, distributed resource protection

1 Introduction

Since its first introduction in 2002, the newly emerging concept of Digital Ecosystem (DE) has received increasing attention from researchers, businesses, ICT professionals and communities around the world. The DE concept is aimed at achieving a set of predetermined goals that resulted from a Lisbon summit in March 2000. The derived objectives primarily focus on dynamic formation of knowledge based economies [1]. Further, it was proposed that a knowledge based economy will lead to a creation of more jobs and a greater social inclusion in sustaining the world economic growth [2].

DE is a multi-dimensional concept that encompasses several current technology models such as collaborative environments [3], distributed systems [4], and grid technology [5]. The combination of concepts from these models provides the DE environment the ability to deliver an open, flexible and loosely coupled resource sharing environment. However, this combinational configuration also produces a number of complicated security issues. Such security problems need to be addressed before the full implementation of a DE concept can be realised. Unfortunately, after review and evaluation of the current literature on the DE security dimensions, a number of deficiencies in DE security architecture, particularly in protecting the enterprise resources and information, are apparent. As such, there is a need for a comprehensive resource protection solution. The solution must be able to provide a strong and rigorous mechanism for safeguarding the DE critical resources from unauthenticated entities in addition to reducing the possibility of information leaking to unauthorized DE member entities.

A key challenge for enterprises that are involved and participating in a DE environment is to determine the right entities or users whom are able to access the services, resources, knowledge and information hosted in the DE by member enterprises. There are several key reasons why this challenge is particularly difficult to address. Firstly, the occurrences of multiple resources published and shared by each enterprise in a DE environment and secondly, the situation where various clients are able to access each individual resource. Due to these reasons, enterprises and more importantly DE's, to ensure their increasing uptake and utility, urgently need a mechanism that effectively manages their clients' access control and authorization permissions with the aim to protect their respective resources. In this paper, we attempt to deliver a comprehensive framework allowing enterprises to protect their resources and information from any unauthorized use. In turn, as displayed by previous similarly security enhanced environments behaviours; provision of improved protection will contribute towards sustaining the credibility, integrity and availability of enterprise resources and information in an evolving DE environment.

The reminder of the paper is structured in the following manner: section 2 lists all key definitions used throughout this paper and our work, while section 3 discusses the background and related work. Section 4 presents the details of our proposal for addressing the security challenges identified in DE environments, section 5 discusses results and evaluation of testing the proposed solution, and finally section 6 provides a brief conclusion of the work contained in the paper.

2 Preliminary Definitions

Before entering into a detailed explanation of the proposed framework, a list of key terminologies used throughout this paper is presented below:

1. *Resources*: any services, knowledge, or information which is published, shared, or registered by an enterprise in a DE environment.
2. *Resource provider*: any entity or enterprise who provide resources in a DE environment.
3. *Client*: any user or enterprise who request for a specific resource provided by a resource provider.
4. *Client profile*: the identity of a client which provides the information about the client and the purpose of requesting a specific resource.
5. *Permission*: a special authorization rule which govern how a resource is being accessed by the client.
6. *Capability*: a mechanism that contains resource access permissions which is entitled to each client profile.

.

3 Background and Related Work

Protecting enterprise resources requires the resource provider ability to explicitly know which clients are accessing what resources at all times. Information about a client and about the purpose of a client in accessing a specific resource is critical for a resource provider to assess whether it is safe to allow a client to access the resources. Furthermore, a resource provider uses this information to perform client authorization and authentication analysis before granting and assigning the resource access permissions to each client. The authorization and authentication processes are extremely important, especially for new resource providers who are integrating and sharing their resources, in a DE environment.

Unfortunately, the current state of development for a DE environment is still unable to provide a reliable client authentication and authorization process over the shared resources. The current proposals focus on managing client identities by implementing a distributed identity management framework [6]. In this framework, client's identities are stored in an Identity Provider (IdP) or Credential Provider which is trusted in the DE environment, and the authentication process is enforced by the means of this Identity Provider. Current technology standards such as Liberty Alliance [7] and SAML [8] are utilized inside the framework to provide the identity federation and certificate token transformation for authentication purposes. The other similar mechanism is Credential Server (CRES) repository framework [9] where client credentials can be stored in both local and remote servers, resembling the Grid Security Infrastructure (GSI) MyProxy Credential [10]. Generally, the authentication process of these mechanisms solely takes place in a credential provider/server where the certificate token is produced for the authenticated client, and this certificate is sent to the resource provider. A client will not be authenticated again in the resource provider as a trust relationship is built between the resource provider and the credential provider/server.

While a few mechanisms have been proposed to manage and authenticate the clients' identities, there are limited frameworks that have been proposed for managing the authorization process of client's permissions. A number of currently popular authorization frameworks for collaborative environments (CE's), include the Community Authorization Service (CAS) [11] and Akenti [12]. Both of which could be adapted and employed in a DE environment. This is due to the fact that both collaborative environments and Digital Ecosystems have a similarity in function and purpose: to encourage a collaboration and interaction between their components [1]. Both CAS and Akenti utilize an authorization policy server which is responsible for managing access permissions of all clients. However, a significant difference can be drawn from both frameworks. Akenti uses a pull method where a resource server authenticates the client first and then query the access decision to Akenti server, while CAS utilize the push method where access decisions and rights are being pushed together to the resource server [13].

Although the mechanisms discussed previously work practically well, there are some apparent issues that may limit the ability of resource providers in providing a strong protection for their resources. Firstly, the current mechanisms store all identities and information about the clients in a central server where the authentication or authorization process is performed. This server is prone to a single failure which would greatly affect the whole environment. In a case that the server fails to perform the clients' authentication or authorization process, no other mechanisms are in place to provide the process. This is in turn can contribute to having all environment resources becoming prone to malicious attacks. Although, it is possible to replicate the central server, the replication process will bring increase administrative issues and in addition to higher chances for compromising the resources considering the huge amount of data to be replicated.

Secondly, a centralized authorization server faces various challenges when it comes to assigning the access permissions to each DE client. This is due to the fact that a huge amount of resource providers are actively involved in a DE environment, and each resource provider is able to host and publish more than one resource. This condition becomes even more challenging as a single resource could be associated with multiple different access permissions, and each client may have different access permissions assigned to him. Therefore, in order to correctly assign the access permissions to each client, the authorization server must be able to answer at least the following questions: On which resource provider is the requested resource located?; What access permissions are defined for this resource?; Is the client allowed to access this resource?; and What access permissions are associated to this client for the requested resource?

A single authorization server will most certainly face a heavy burden in assigning multiple various and diverse permissions correctly to each individual client. For a simplistic scenario example, assume that there are only 30 resource providers providing their resources in a DE environment and each resource provider hosted 30 dissimilar resources which are registered and published in a DE environment. Further, each resource has 30 different access permissions associated with it. If there is only a single authorization server responsible to assign the resource access permissions to each DE client, this server must handle at least 900 different resource access permissions in its own repository. Further, the server must also understand the association links between the access permissions and each resource provider as well as the association links between resources and the resource providers. Therefore, it is very laborious and complex for a single server to effectively manage those permissions and assign them correctly to multiple clients. Furthermore, as a DE environment grows in size and diversity, more and more clients and resource providers participate and interact in the environment due to the potential benefits that they can achieve. A central server will be increasingly experiencing serious administrative issues in trying to manage all client and resource providers' accounts and permissions even with the possible use of super computers or grid collections of computers.

Due to the identified issues on the current mechanisms from the literature, we argue that the resource protection mechanism in a DE environment must be a distributed process rather than a centralized process. The distributed process is implemented by allowing each resource provider to perform its own clients' authorization and authentication process, and further facilitating the resource provider to assign the resource right permissions to its clients. To fulfil these requirements, we propose a Distributed Resource Protection Mechanism (DRPM). DPRM could effectively perform the authorization process and manage clients' permissions in a DE environment. In the following section, we present and discuss a detailed explanation of our propose DRPM framework.

4 Distributed Resource Protection Mechanism

Inside the DRPM framework, the concept of client profiles is used to identify and keep clients' information. A new client who makes a request for a specific resource is required to provide their information which will then be stored in a client profile. Once a client profile is created, the resource provider will grant the resource access permissions to the client based on a list of permissions defined inside a capability [14]. This capability will always be used every time a client request for the same resource occurs.

4.1 Client Identification

In the present mechanism for service discovery inside a DE environment, a client searches for resources by utilizing a semantic discovery portal through their browsers or rich applications [15]. The discovery portal lists all client intended resources which are provided by DE resource providers. Once the client finds the resource that they want, they then contact the resource provider and requests for the resource. At this stage, the resource provider does not know any information about the client and their intended purpose on the resource. This may put the resource at risk as the resource provided by a resource provider may contain highly sensitive information which must be protected from any misuse and malicious act. Therefore, it is crucial for a resource provider to understand a requesting client's information before any access to their resources is granted.

Taking this into consideration, we adopted a method of creation for a client profile that aims to capture all required, but voluntarily provided, information about a client. The information which is contained in a client profile provides necessary data about who the client is and about their intentions and purpose for using the requested resources. The aim of implementing a client profile is to ensure the resource provider that resources are not going to the wrong entities and further impose the confidentiality and integrity of the resources. The use of client profiles also facilitates the auditing process of the clients accessing a resource. For example, there may be a situation where a resource provider needs to make a trace back to determine which client was delegated access to the resource in case there was an incident involving a dispute or counterfeiting of the resource in question.

In order to fully implement the client profile, it is necessary that a client registration portal is employed in our framework. A client profile is generated through this registration portal. It is built on the HTML language so that it can be universally accessed through a clients' browser. Further, resource providers are able to customize the registration portal to contain only the information which is important to them. New clients wishing to access a specific resource are initially redirected into this portal. If they wish to access the resource, they must continue to fill in all the necessary information required by the resource provider to produce a client profile. Once it is produced, the client profile is stored in the resource provider repository and can be used for authenticating the client. On future access requests on the same resource, the client will provide their credential and it will be mapped to their client profile. If it is matched, the client will be granted another access to the resources. Utilisation of this functional procedure and process provides an additional and enhanced method for determining who is accessing a particular resource at a particular time within a DE.

4.2. Enforcing Access Permission

It is a challenge to enforce client access permissions on resources available within a DE environment. As discussed previously, a DE environment comprises various clients, and these clients could make the same request for a particular resource either at the same or at different time. Through our own investigations and as the literature indicates it is apparent that resource providers who publishes DE resources are having difficulties in managing the resource access permissions of the multiple and diverse range of DE clients. A further complication to the problem is that a resource provider is able to have multiple resources registered and published in a DE environment. This situation creates a complex and hard to define set of resource access permissions for each client. Therefore, a strong mechanism for managing the access permission of each client and their intended requested resource is crucial. This access permission mechanism must provide the ability for a resource provider to know which clients have the permission to access which resources.

To solve this issue, we utilize and evolve the concept of capability introduced by the CAS server used in a Collaborative Environment. In CAS, capability is used to store all access rights of a user which are determined by a community policy. However, the implementation of the capability in our framework is slightly different to the capability implementation in a CAS server. In our framework, capability contains all the necessary right permissions for each client to perform a set of operations on a particular resource. This capability is produced by the resource provider on which a particular resource is hosted. This capability is used by the resource provider to grant the client access to the resources and further provide the authorization process for the clients. After a new client profile is created, the resource provider generates a capability for this client profile that defines all resource access granted to the client. The capability limits the resource access of a particular client by listing all permissions which are granted to the client. Further, this capability is sent to the requesting client and will be used for future requests on the same resource. Every time a client makes a request to the resource provider for the same resource, the client sends back its initial configured capability to the resource provider. The resource provider then authenticates the provided client's capability and grants the access permissions based on the permissions stored inside the client's capability. Permissions and policies languages in a capability are expressed by using SAML Authorization Assertions [16]. Further, Public Key Infrastructure (PKI) [17] can be implemented in order to protect the capability by using the resource provider private key to sign the capability.

4.3. Client Profile and Capability Token Management

Using a combination of a client profile with a capability seems to solve the problems associated with accurately identifying the clients who make resource requests in addition to be able to grant access permissions on a particular resource to a specific client. However, an administrative problem is faced by a resource provider in managing multiple client profiles and diverse capability tokens. This is again due to the ability of a resource provider to host more than one resource in a DE environment while the clients' access permissions on these resources are distinct.

In order to address this administrative challenge, we propose an association model between the capability tokens and client profiles. As a client profile contains the general information about a client and is going to be used for accessing multiple resources hosted by a single resource provider, we propose that this client profile is only issued once. It is issued when the client request for the resource at occurs for the first time. Once issued, a client profile can be used to access all resources with the condition that these resources are hosted by the same resource provider. Therefore, it promotes a single registration for all resources and reduces the potentially overwhelming client registration process required every time a client requests as different resources. In a further step, a capability token which contains all access permissions of a particular client on a single resource has an association of one too many relationships with each client profile that it represents. This relationship means that each client profile may have more than one capability token in accessing different resources provided by a single resource provider. The implementation is showed in figure 1.

As shown in figure 1, each capability represents a list of access permissions of a particular client profile on a particular resource. Capability that is owned by different client profiles can be applied on the same resources within the same resource provider e.g. Capability 1 and 3 of client profile A and capability 5 and 6 of client profile B can be applied on the same resources respectively. However, those capability tokens that are associated with a particular client profile could not be applied on the same resources e.g. all capability tokens owned by client A/B are not applied on the same resource (capability 1 applied on resource 1, capability 2 applied on resource 2, and so on). For a further identification process of each capability on the resource that it represents, we register a resource manifest ID on each capability. In current DE technology, a service is identified by a service manifest which contains a unique ID representing a service [18]. The registered manifest ID will allow the client to send the correct capability token when requesting for a specific resource. Therefore, the association between client profile and capability allows the resource provider to specifically know who has access permission on a particular resource and what kind of permission is allowed on that resource. This approach, further address the hectic administrative issue associated with managing the client profiles and capability tokens.

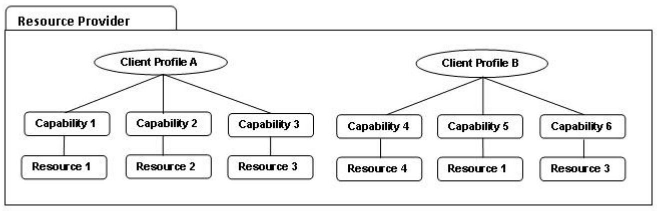

Fig. 1: Classification hierarchies of client profiles and capability tokens

4.4 Registration and Authentication Process

On every authentication process, a client is required to provide his own certificate or credential to the resource provider. Resource provider then retrieves client information from the certificate and maps this information with their own client profile. A certificate or credential can be obtained by a client in two possible methods: 1) a certificate issued by external party or 2) a certificate issued by a proxy credential server. The clients may have already implemented their own certificates issued by their Trusted Third Party (TTP) or Certificate Authority (CA), such as VeriSign [19], TrustAlert [20], etc. This certificate is required to be used for each and every online transaction. Additionally, a proxy credential server [21] could also be implemented in a DE environment to provide the certification services for the clients. In order to cope with those methods, the certificates issued by either external parties or a proxy certificate can be accepted as long as a trust relationship is built between the certificate issuer and the resource provider. A client is responsible for requesting a X.509 certificate [22] from his own CA or X.509 proxy certificate [23] from the proxy credential server. Further, the client must send this certificate to the resource provider for authentication on each time they make a request for a specific resource.

However, as a primary focus of DE is to promote ICT to the Small and Medium Enterprise (SMEs) [24], there is a strong possibility that such SMEs do not have any certificate issuing party. Therefore, a solution for these types of environments needs to take into consideration and facilitate the authentication process of their clients. The combination of username and password is the simplest way to provide the authentication to the clients. Although this method has several disadvantages such as the difficulty for the clients to manage multiple usernames and passwords [25], it is still the cheapest and easiest method to provide the authentication for the clients before accessing the resources. The authors also note that username/password methods of authentication are also deemed the least secure. However, the focus of this paper is the improvement of the DE authentication and authorization process, not the strength of the methods used.

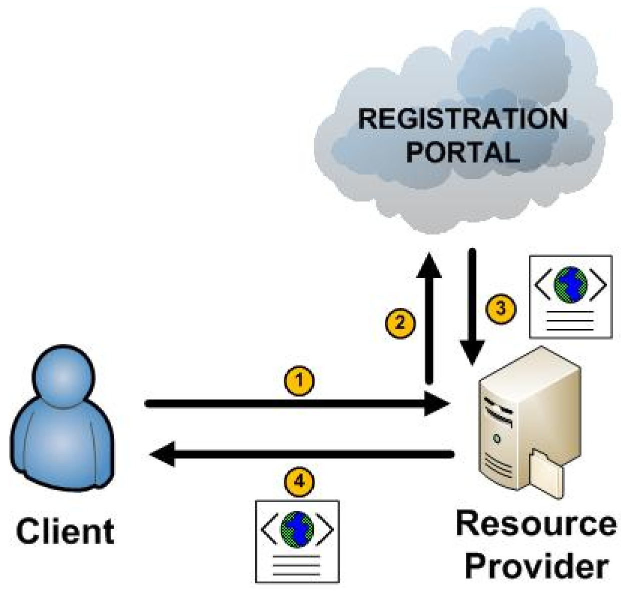

Fig. 2: New client requesting for resource

4.5 The Workflow Process

Having discussed all major components of our DRPM framework, we present two workflow process scenarios for when a resource is being requested. The first scenario shows a request process for those new clients who have never requested and consumed any resources provided by a resource provider. The second scenario presents the request process for those clients who have previously requested or consumed resources from a resource provider.

Scenario 1: A new client request for a specific resource
Figure 2 shows the workflow process model for a new client who has never requested or consumed any resources before.

1. A new client attempts to access a specific resource by making a request to the resource provider where the resource is hosted.
2. Resource provider asks for client certificate and resource capability token. Once client send the certificate, the resource provider will match the certificate with the client profiles stored in its repository. If the client profile is not found, the client will then be redirected to a registration portal. On this portal, the client is required to fill in his information.
3. Registration portal sends back the client information to the resource provider. Resource provider generates a client profile based on the information provided by the portal. Further, resource provider assigns the resource access permissions and policies of a requesting client into a capability token.
4. Resource provider stores the client profile and its capability inside its repository for future resource request. Further, the capability token is signed by the resource provider by using its private key and it is sent back to the client.
5. The client is granted the access to the requested resource based on the newly created capability token.

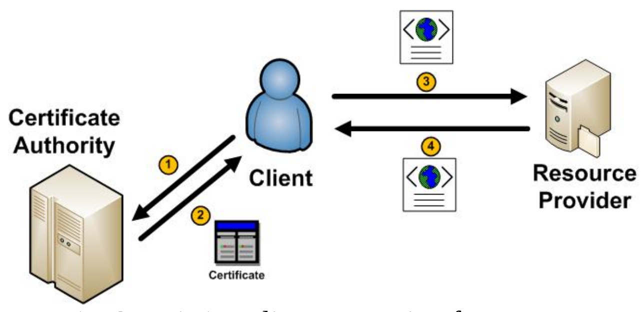

Fig. 3: Existing client requesting for resource

Scenario 2: A registered client request for a specific resource

Figure 3 shows the workflow process model for a client who have ever requested and consumed any resource provided by a resource provider.
1. A client who makes a request for a specific resource will retrieve his certificate from his own Certificate Authority (CA) or a proxy credential server who resides in DE environment.
2. CA or proxy credential server authenticates the client and sends his encrypted certificate back. Then, Client looks at his repository for the associated capability token of the requested resource.
3. Client will then send both certificate and capability token to the resource provider. Thereafter, the following steps are occurred:

3.1 The resource provider verifies and authenticates the client certificate and further matches client information with the client profile stored in its repository. If the certificate information does not match with the client profile, client will not be allowed to access the resource.

3.2 The resource provider decrypts and authenticates the capability token sent by the client.

3.3 Resource provider will then authorizes the client and grants the access control to the resource based on the permissions defined in client's capability token.

4. In a case that a client does not have the capability token for the resource as he has not previously consume this resource, client will only send his certificate to the resource provider. Thereafter, the following steps are occurred:

4.1 The resource provider verifies and authenticates the client certificate and further matches client information with the client profile stored in its repository. If the certificate information does not match with the client profile, client will not be allowed to access the resource.

4.2 The resource provider then assigns all resource access permissions and policies of the client on the requested resource. These access permissions are stored in a capability token. Then, this capability token will be signed using resource provider's private key and sent to the client.

4.3 The resource provider then authorizes the client and grants the access control to the resource based on the access permissions defined on client's capability token.

4.6 Granting Access to Different Entities

As discussed previously, once a capability token is created by a resource provider, this token is sent to the requesting client. Further, this capability token is stored inside the client repository and is presented by the client on future requests for the same resource. This mechanism may be dubious as a question may arise on why there is a need to send the capability token back to the client while this capability token can be stored in the resource provider repository and used when the same client make the request on the same resource. Therefore, we attempt to provide a further explanation and discussion in addressing this question.

In a DE environment, there is a strong possibility that a client, which in this case could be an enterprise, is able to allow a number of its employees to request and consume the published resources using its client's profile. These employees are trusted by their enterprise, and they are representing the enterprise in doing online business negotiations and transactions. Although multiple employees use the same client profile, different access restrictions on the resources must be set accordingly. This is due to the fact that the enterprise as the owner of client profile has different policies specifying who should able to access which resources. Therefore, a number of cases show that an enterprise may want to grant different resource access to different employees based on its own policies. For example, enterprise allows employee A to use its client profile for accessing resources 1 and 2 on resource provider I however, enterprise only allows employee B to use its client profile for accessing resources 2 on resource provider I.

By sending the capability token to the client, it allows the client to allocate specific resource access to each employee. A client can further decide by itself which employee is able to access which resources by allocating the right capability tokens to the right employees instead of letting all employees access all resources using its generic client profile. The workflow process for the employee to access the specific resource using his enterprise client profile is detailed as follows: once an enterprise has granted and allocated the capability tokens to their employees respectively, an employee who wants to access the specific resource from the resource provider must ask for the capability token from his enterprise and send this token together with the enterprise certificate to the resource provider. Then the resource provider grants the resource access for the employee based on the received capability token.

5 Framework Testing and Evaluation

5.1 Enforcing Strong Access Limitation on the Resources to Multiple Resource Consumer

The testing process ensures only the authorized clients that are allowed to access the resources. In a resource sharing environment, it is essential for the resource provider to provide a limitation of access on their resources. Resource provider must only allow the authenticated and authorized clients to access the resources while prohibit the unauthorized clients from gaining access to the resources. Therefore, the aim of this testing process is to ensure that the framework enforces this strong access limitation on the resources. We developed a structure scenario to facilitate the testing process. We developed a resource provider which hosts two resources in its server. Further, we constructed two clients who attempted to access both resources. The first client is authenticated and recognized by the resource provider as it has gone through the registration process. Therefore, the resource provider has the corresponding client profile in its repository. The second client is a new resource consumer who has not been authenticated or registered by the resource provider. Access permission on the first resource was given to first client; however, the client was not given any access permission to access the second resource. Moreover, the second client has not been given any authorization permissions to access any of the resources.

In this testing process, we would verify whether the framework allow the clients to access both resources. We would investigate which of clients is authorized to access the resources. Further, we would provide the validation on which of resources can be accessed by the authorized client. We conducted the testing on our prototype implementation based on the previous scenario. The first test was getting both clients to request for the first resource hosted by the resource provider. The testing process resulted that the first client was able to request for the resource. Further, the access permissions of the first client are showed on the page. However, the second client was directly redirected to the registration page without have any access on the resource. The first screenshot showed that the company or client 1 successfully accessed the resources while the second screenshot showed that the company or client 2 was redirected to the registration page. The next testing was getting both clients to access the second resource. This testing resulted that both clients did not able to access the resource. For the second client, the client was still redirected to the registration page. However, the first client was not redirected to the registration page. It was redirected to the failed authorization page.

The evaluation on the previous testing proved successful for resolving one of our main identified research issues. In the first testing process, the first client was allowed to access the resources due to this client has been authenticated by the resource provider. Further, this client has the validated authorization permissions to access the resources. On the other hand, the second client was not given an access on the resource as it was not recognized by the resource provider. The second client was redirected to the registration page. To conclude, this testing process provides the validation that only to those clients that have been registered and authenticated by the resource providers and hence are able to access the resource.

The result of second testing process shows that only the authorized clients are permitted to access the resources. Although the first client has been registered and authenticated by the resource provider, this client has permission only to access the first resource. He has no access permissions toward the second resource. Therefore, the resource provider denies the access of the client to the second resource. The results of our extensive testing on the prototype have proved that the DRPM framework enforces a strong access limitation on the resources to multiple clients. Further, this framework also successfully limits the access of each client on multiple resources.

5.2 Defining and Enforcing the Access Permissions of Each Resource Consumer

In this testing process, we are investigating the mechanism whereby the resource provider could clearly define the access permissions of each resource consumer on its resources. As discussed in chapter 4, a trivial issue is identified for managing multiple clients' access permissions. In a DE environment, it is common that each client is assigned with unique access permissions. Further, a huge number of clients who are accessing the resource generate complexity for the resource provider to strictly enforce the unique permissions assigned to each client. Thus, the aim of this testing process is to investigate the ability of the framework to define the unique client access permissions, and further force these unique permissions during the resource access.

In order to conduct this testing process, we assigned two clients with different access permissions on the same resource. The assignment of different access permissions were enforced by assigning different roles to the two clients. This role assignment to the client further implements the RBAC concept. The first client was assigned with 'Executive client' roles which gave him the read, update and modify access permissions on the resource. Further, we assigned the second client with 'Common client' role which would only give him read access permission on the resource.

The testing process would provide a verification and validation of managing the unique clients' access permissions. The investigation would be carried out to look at the ability of framework to enforce the correct access permissions to each resource consumer. The testing result presented on the next section will be reviewed and evaluated. The testing process is conducted based on the previous discussed scenario in our prototype implementation. The result of the testing process showed that the first client was able to access the resource. Further, this client has read, update and modify access permission on the resource.

The second client was able to access the resource as well however, the resource access permissions given to this client was only read access permission. Figure 6.4 presents the access permissions granted to the second client. The result has proved that DRPM framework is able to define the unique access permissions to individual resource consumer. The first client was successfully given all three permissions for accessing the resource. Further, the second client was only given a read permission for accessing the resource. The testing process has validated the framework ability in defining individual client access permissions. Unique access permissions are strictly given and enforced to each client.

5.3 Identifying the Accessing Clients and Their Purposes in Accessing the Resources

This testing process ensures that a new resource consumer which has not been authenticated to use the resource is redirected to a registration page. As discussed, once a client certificate is validated from the retrieved HTTP request message, resource provider extracts the client unique username from his certificate and capability token. This username is mapped with the client profiles that are stored inside server database to check whether the client is authorized to access the resource.

We implement a database to store the client profiles. Resource provider creates a query to retrieve all their client profiles and further, it validates the received username from HTTP request message with the client profiles. When resource provider could not find the matched client profiles, resource consumer is redirected automatically to a registration page. Resource consumer must provide his details for further resource access. Further, a client or resource consumer must state his aim of accessing the resource. This information is essential for the resource provider to understand the intended purpose of accessing the resource before granting the permission to access the resource.Once the resource consumer successfully registered with the resource provider, resource provider then creates a client profile based on the information provided. This client profile is stored inside resource provider database.

6 Conclusion

In this paper we have highlighted the needs for protecting enterprise resources from any unauthorized use in a DE environment. Further, we have discussed the issues faced by the current available and proposed security mechanisms for DE's in providing this protection. From our analysis and research work in this area we have proposed the use of the Distributed Resource Protection Mechanism (DRPM). The DRPM can be classified as a new approach to facilitate the authorization process for enterprises who request specific resources or information. DRPM emphasizes decentralised authorization mechanism performed by individual resource providers. It is achieved by utilizing the client profile and capability token.

The utilization of client profile provides the ability for a resource provider to garner a better understanding of its clients before any access to the resources is given. In addition, it provides the means to allow resource provider to trace back to a specific client access transaction in case any malicious attacks may have occurred. Enforcement of the authorization permissions is achieved by implementing the capability token. A capability token list contains all the permissions granted to a particular client on a particular resource and are defined by SAML technology. The capability token further allows an enterprise to delegate the resource access permissions to each of its employees.

As the authors have conducted extensive research into the related fields of collaborative environments [26, 27, 28] potential limitations of the solution have been identified. Going forward, the future work for the research project is to address these limitations. Some identified limitations of our proposed current solution are the following:

1. User ought to login to a DE through their browser or rich client which means that registration is needed to a DE environment and there should be a single server containing user registration. How if this server is down? is the DE registration really needed? DE promotes the flexibility and transparency to use the ecosystem, why the user needs to register first?
2. Current solution offers an identity provider/credential provider who provides the identity of user and authenticates the user to service provider. The trust must be established between the service provider and identity provider. However, it makes the centralized approach of identity management. The question to be answered is: How if the current identity provider is down??
3. From the current solution, the service provider will only get the certificate token mentioning that the user is identified and authenticated to use a particular service. The question is: how if the service provider would like to know the detail information of the user who uses its service?
4. There is no delegation ability to trace which user uses which resources and what kinds of permissions are allowed to use the resources?
5. DENIAL OF SERVICE Attacks

References:

[1] F. Nachira, P. Dini, and A. Nicolai, "A network of digital business ecosystems for Europe: roots, processes and perspectives," European Commission, Bruxelles, Introductory Paper, 2007.

[2] P. Dini, M. Darking, N. Rathbone, M. Vidal, P. Hernandez, P. Ferronato, G. Briscoe, and S. Hendryx, "The digital ecosystems research vision: 2010 and beyond," European Commisssion, Bruxelles, Position Paper, 2005.

[3] I. L. Ballesteros, "New Collaborative Working Environments 2020," European Commission, Report on industry-led FP7 consultations and 3rd Report of the Experts Group on Collaboration@Work 2006.

[4] M. van Steen, P. Homburg, and A. S. Tanenbaum, "Globe: a wide area distributed system," IEEE Concurrency vol. 7, pp. 70-78, 1999.

[5] K. Czajkowski, C. Kesselman, S. Fitzgerald, and I. Foster, "Grid Information Services for Distributed Resource Sharing," in 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01), 2001.

[6] H. Koshutanski, M. Ion, and L. Telesca, "Distributed Identity Management Model for Digital Ecosystems" in International Conference on Emerging Security Information, Systems and Technologies (Securware'07) Valencia, 2007.

[7] L. Alliance, "Liberty Alliance Project." http://www.projectliberty. org/.

[8] OASIS, "Security Assertion Markup Language (SAML)," 2005. http://www.oasis-open.org/committees/security/.

[9] J. M. Seigneur, "Demonstration of security through collaborative in digital business ecosystem," in Proceedings of the IEEE SECOVAL Workshop, IEEE, Athens, Greece, 2005.

[10] J. Novotny, S. Tuecke, and W. V, "An online credential repository for the Grid: MyProxy," in Proceedings of the IEEE Tenth International Symposium on High Performance Distributed Computing (HPDC-10), San Fransisco, California, 2001.

[11] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, "A community authorization service for group collaboration," in Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (POLICY'02) California, USA, 2002.

[12] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari, "Certificate-based access control for widely distributed resources," in Proceedings of the 8th conference on USENIX Security Symposium, Wshington, DC, 1999.

[13] K. Keahey and V. Welch, "Fine-Grain Authorization for Resource Management in the Grid Environment," Lecturer Notes in Computer Science, vol. 2536, pp. 199-206, 2002.
[14] S. J. Mullender and A. S. Tanenbaum, "The design of a capability based distributed operating system," The Computer Journal, vol. 29, pp. 289-299, 1984.
[15] J. Kennedy, "Distributed infrastructural service," in Digital Ecosystem Technology, F. Nachira, P. Dini, A. Nicolai, M. Le Louarn, and L. R. Leon, Eds.: European Commission: Information Society and Media, 2007.
[16] J. Hughes and E. Maler, "Security Assertion Markup Language (SAML) V2.0 Technical Overview," OASIS, Working Paper 2005.
[17] J. Weise, "Public key infrastructure overview," Sun Microsystem, Sun BluePrints Online 2001.
[18] T. Kurz and T. J. Heistracher, "Simulation of a self-optimising digital ecosystem," in Inaugural IEEE-IES Digital EcoSystems and Technologies Conference Cairns, Australia, 2007, pp. 165-170.
[19] VeriSign, "Verisign Security." http://www.verisign.com/.
[20] TrustAlert, "TrustAlert Security." http://www.trustalert.com/.
[21] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A security architecture for computational grids," in Proceedings of the 5th ACM conference on Computer and communications security, San Francisco, California, 1998, pp. 83-92.
[22] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280, IETF 2002.
[23] I. Foster, V. Welch, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist, "X. 509 Proxy Certificates for Dynamic Delegation," in Proceedings of the 3rd Annual PKI R&D Workshop, Gaithersburg MD, USA, 2004.
[24] F. Nachira, E. Chiozza, H. Ihonen, M. Manzoni, and F. Cunningham, "Towards a network of digital business ecosystems fostering the local development," Bruxelles, Discussion Paper 2002.
[25] D. Agarwal, M. Lorch, M. Thompson, and M. Perry, "A New Security Model for Collaborative Environments," in Proceedings of the Workshop on Advanced Collaborative Environments, Edinburgh, Scotland, 2003.
[26] Skinner, G., "Making A CASE for PACE: Components of the Combined Authentication Scheme Encapsulation for a Privacy Augmented Collaborative Environment," WSEAS TRANSACTIONS on COMPUTERS, Issue 1, Volume 7, January 2008.
[27] Skinner, G., "A Privacy Augmented Collaborative Environment (PACE)," PROCEEDINGS of the 7th WSEAS INTERNATIONAL CONFERENCE on APPLIED COMPUTER SCIENCE (ACS'07), Venice, Italy, November 21-23, 2007.
[28] Skinner, G., "Shield Privacy: A conceptual framework for Information Privacy and Data Access Controls," WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Volume 5, June 2006, pp. 1375-1384.