Distributed Mechanism for Protecting Resources in a Newly Emerged Digital Ecosystem Technology

Ilung Pranata, Geoff Skinner, and Rukshan Athauda

University of Newcastle, School of Design, Communication and IT,
University Drive, Callaghan NSW, Australia
{Ilung.Pranata,Geoff.Skinner,Rukshan.Athauda}@newcastle.edu.au

Abstract. Digital Ecosystem (DE) is characterized as an open and dynamic environment where the interaction and collaboration between its entities are highly promoted. A major requirement to promote such intensive interaction and collaboration in a DE environment is the ability to secure and uphold the confidentiality, integrity and non-repudiation of shared resources and information. However, current developments of such security mechanisms for protecting the shared resources are still in their infancy. Most of the proposed protection frameworks do not provide a scalable and effective mechanism for engaging multiple interacting entities to protect their resources. This is even a greater issue when multiple resources are exchanged and shared in an open and dynamic environment. Therefore, we proposes a distributed mechanism for individual enterprises to manage their own authorization processes and resource access permissions with an aim to provide a rigorous protection of entities' resources.

Keywords: Authentication, authorization, digital ecosystem.

1 Introduction

Since its first introduction in 2002, a new emerging concept of Digital Ecosystem (DE) has grasped numerous attentions from researchers, businesses, ICT professionals and communities around the world. This concept aimed at achieving a set of predetermined goals of Lisbon summit in March 2000 which primarily focuses on dynamic formation of knowledge based economy [1]. Further, the 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 environment [3], distributed system [4], and grid technology [5]. The combination of concepts from these models provides the ability for a DE environment to deliver an open, flexible and loosely coupled resource sharing environment. On the other hand, this combination also develops several complicated security issues which need to be addressed before the full implementation of a DE concept. Unfortunately, the evaluation on DE security dimensions from the current literature signifies a number of deficiencies in its security architecture particularly in protecting the enterprise resources and information. There is a need for a comprehensive resource protection solution that is able to provide a

strong and rigorous mechanism to safeguard the critical resources and further to reduce the possibility of information leakage to the unauthorized parties.

A key challenge for enterprises who involved in a DE environment is to determine the right users who are able to access the services, resources, knowledge and information hosted by these enterprises. This challenge is occurred due to several reasons. First, the occurrences of multiple resources published and shared by each enterprise in a DE environment and second, the situation where various clients are able to access each individual resource. Due to these reasons, enterprises urgently need a mechanism that effectively manages their clients' access control and authorization permissions with an aim to protect their resources. In this paper, we attempt to deliver a comprehensive framework allowing enterprises to protect their resources and information from any unauthorized use.

2 Related Work

In a DE environment where multiple interacting entities exist, the required efforts to enforce a strong authentication and authorisation mechanism are extensive. We identify three core issues that appear to be the challenging tasks to enforce such mechanisms. First, as the DE community expands its size to incorporate more entities, the resource providers face a challenge to identify the legal entities that are able to access their resources. Second, the fact that each entity would have different set of access permissions to access multiple resources further complicates the issue. Third, it is probable that each resource provider would host multiple resources and services in a DE environment. This situation, in turn, creates a great issue to authorize the right entities to the right resources with the right permissions. Failure in assigning right permissions to the entities would compromise the usage of resources and would bring negative impact for resource provider.

The current internet mechanisms are still far from adequate to provide a reliable authentication and authorisation processes for a DE environment. This view is reflected from our literature analysis over a number of internet mechanisms. The most prominent mechanism to manage the client credentials is through the implementation of Identity Provider (IdP) or Credential Provider [6, 7]. IdP mainly focuses at storing and providing the client credential to any resource providers for their client authentication process. On every authentication process, resource provider will request the client credential from its trusted IdP where it receives any access request from a client. In latter development, several technology standards such as SAML [8] and Liberty Alliance [9] are adopted in this mechanism to provide the federation mechanism of multiple entities for Single Sign On (SSO) services. Similarly, the Credential Server (CRES) [10] and the Grid Security Infrastructure (GSI) MyProxy [11] utilizes the IdP concept, and they further leverage its concept for large number of dispersed servers over a wide geographical area. Both mechanisms store the clients' credentials in the local server; however, authentication of a remote client can be facilitated by requesting his credential from the trusted remote server.

In both MyProxy and CRES mechanisms, the resource provider requests the client credential from the local server on every authentication process. The local server then creates a certificate token which contains client information. Subsequently, the

certificate token is sent to the resource provider as the acknowledgement of the authentic client. When the resource provider receives the token, it allows the client to access the resources based on the trust established with the publisher of token. The SSL/TLS technology [12] has been extensively used in e-commerce transactions for secure authentication and communication. This technology is designed with a highly reliance on the Certificate Authority (CA) to ensure the legitimacy of an entity. Therefore, SSL/TLS technology also presents a centralized credential management. Although these approaches could be deployed well in a DE environment, the conspicuous issue of single server failure must be carefully considered. In an event where the credential provider server is down, there possibly a chaos in a DE community due to the unavailability of credential services for client authentication.

Apparently, our literature review identifies that several internet authorisation mechanisms take similar approach as its authentication mechanisms. The most prominent authorisation mechanisms, such as CAS [13], Akenti [14], and PMI [15], utilize a central server to assign the multiple access permissions to the clients individually although their implementation are differ between each other. These mechanisms also inherit several issues pertinent to the central management of authorisation permissions. First, the central management would face real issue with the bottleneck and failure on its servers. Security breach would occur if the central servers fail to perform their authorisation processes over the clients. Although it is possible to replicate the central server, the replication process will bring abundance administrative issues, considering a huge amount of data that needs to be replicated.

Second, challenges occur when the central server attempt to assign the access permissions to the DE member entities. As a large number of resource providers that host one or more resources, the central server needs to register each resource and its access permissions individually. Further, this situation 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, the central management is not practical when there is huge number of entities in a DE environment. Third, serious administration issues would occur as a DE environment grows in size and diversity due to the great benefits that they can achieve. A central server will be experiencing huge burden to manage all client and resource providers' accounts and permissions even with the use of super computers or grid collections of computers.

Several DE literatures clearly reveal that DE is characterized as an open environment on which a centralized structure is minimized. DE must be engineered to provide a high resilience infrastructure while avoiding single point of control and failure [16, 17]. Therefore, a completely distributed control mechanism is required that immune to the super control failure. It is evident that the aforementioned internet mechanisms are inappropriate to be implemented in a DE environment due to Its centralized management. In this paper, we propose a solution to manage the authentication and authorization in a full distributed approach that focuses on each entity to manage the authentication and authorization mechanism with the utilization of a capability token. We termed our solution as Distributed Resource Protection Mechanism (DRPM) [18, 19]. In this paper, we enhance our solution by eliminating its reliance on the central credential server and further secure the mechanism by utilizing the Public Key Infrastructure [15].

3 Overview of DRPM

3.1 Identifying Entity through Client Profile

The present mechanism for service discovery in a DE environment requires a client to search for resources by utilizing a semantic discovery portal through its browsers or rich applications [20]. This discovery portal would search and list all resources which are provided by DE resource providers. Once the client finds the resource, it then contacts the resource provider and requests for that resource. At this stage, resource provider does not know any information about this client and its intended purpose on the resource. This may put the resource at risk as it 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 its client's information before any access to the resource 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 on who is accessing a resource. For example, there may be a situation where a resource provider needs to trace back which client was delegated an access to the resource in case there was an incident involving a dispute or counterfeiting of the resource. In order to fully implement a client profile, it is necessary that a client registration portal is employed in DRPM. A client profile is generated through this registration portal. 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. 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 inside a DE environment.

3.2 Storing Permission in a Capability Token

It is always a challenge to enforce client access permissions on the available resources within a DE environment. This challenge is due to the occurrences of a high amount of entities that actively interact in a DE environment. Further, these entities could also make the same request for a particular resource either at the same or at different time. To solve the issue of managing multiple resource access permissions on a diverse range of DE clients, we utilize and further evolve the concept of capability introduced by CAS server that is 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 that hosts a particular resource. This capability would be used to grant the client access to the resources, and it further facilitates the authorization process for the clients.

Once a client profile is created, a list of client authorization permissions are assigned into the capability token. The client's access permissions and policies are expressed in XML [21] due to its simplicity, wide usability and self-descriptive characteristics. Our basic design of a capability token contains the client profile identifier, resource provider identifier, resource identifier and list of access permissions. A time-stamp can be implemented in the capability token to determine the validity period of a client when accessing the resources. In the event where the trustworthiness of a new client is equivocal, a short-life capability token can be issued. Once the trustworthiness of this client gradually increases, resource provider can replace the short-life token with longer time-stamp validity. Additionally, the Uniform Resource Locator (URL) of resources is embedded in the token to provide an automatic and seamless connection to resource servers. Once a capability token is created, it would be disseminated to the requesting client. Every time a client makes a request to the resource provider, the client sends back its initial configured capability to the resource provider. The resource provider then authenticates the client's capability token and grants the access permissions based on the listed permissions obtained from the client's capability.

4 Developing a Secure DRPM Workflow

In this section, we present a secure DRPM which provides a strong authentication and authorization mechanism while upholding the confidentiality, integrity and non-repudiation of resources. The following notation will be used to mathematically define the secure DRPM:

- Cl: Client that request for the resources.
- RP: Resource Provider that host the resources.
- PK_i: Public Key of i.
- SK_i: Secret Key of i.
- Cl_{CP}: Capability token of client.
- $S_i(x)_i$: Signed object x with private key of i.
- $E[x]_j$: Encrypted object x with public key of j.
- AT_{Cl}: Authentication Token of client.
- SyK_i: Symmetric key passphrase of i
- $i \rightarrow j:\{x_1, \ldots, x_n\}$: A message sent from i to j with content x_i to x_n.

given that:

- $PK_i \sim SK_i$: Public key of i is only related to secret key of i therefore, $S_i(x)_i$ can only be verified with PK_i and $E[x]_i$ can only be decrypted with SK_i.

4.1 Securing Registration Workflow

The DRPM registration portal is used to generate a client profile during the initial resource provisioning. This registration portal also captures the client information and possibly his reasons for accessing the resources. The registration process comprises of three main stages: client registration, public key exchanges, and secure transfer of capability token. The resource provider endorsed certificate is utilized to identify the authentic resource provider based on its community endorsed public key certificate, which will be discussed in the next sub-section. The Public Key Infrastructure (PKI) is used to provide a secure communication between the client and resource provider. Figure 1 shows the principal workflow for securing three stages of registration process.

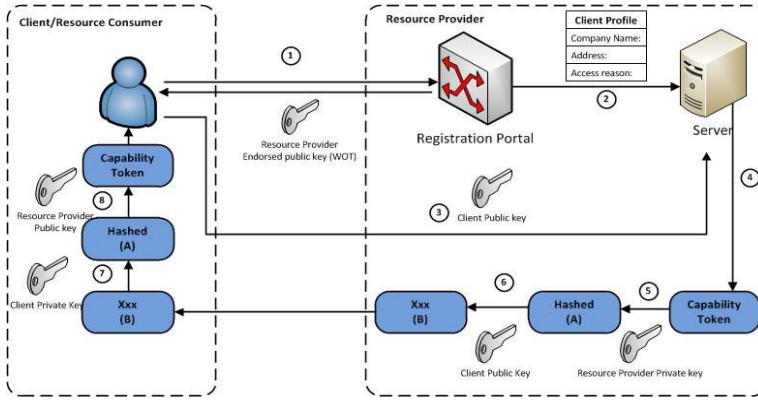

Fig. 1. DRPM secure registration workflow

The registration steps are detailed below:

1. A new client contacts the resource provider for requesting a resource (Cl → RP). Resource provider sends its WoT endorsed public key to the client (Cl ← RP:{PK_{RP}}). Once the client determines and accepts the trustworthiness of the public key, he stores the resource provider trusted public keys and fills his information on the registration portal.
2. After the client information is filled, the registration portal will build a unique client profile which identify the client, and send this client profile to the repository server.
3. Resource provider then requests for client certificate and stores the client public key on its repository *(*Cl:{PK_{Cl}} → RP*)*. If required, WoT verification could be performed on client certificate to ensure the trustworthiness of the client.
4. The resource provider generates a client capability token based on client's allowed permissions.
5. Resource provider uses its own private key to sign the capability token (SK_{RP} + Cl_{CP} = $Si(Cl_{CP})_{RP}$*)*. SHA Algorithm is used to hash the capability token. This process enhances the integrity of capability token over the untrusted network.

6. Resource provider then uses client's public key, received from step 3, to encrypt the signed message *(*PK_{Cl} + $S_i(Cl_{cp})RP$ = $E[S_i(Cl_{cp})RP]Cl$*)* and send it to client end- point ($Cl \leftarrow RP:\{E[S_i(Cl_{cp})RP]Cl\}$).
7. Client uses his own private key to decrypt the encrypted capability token ($E[S_i(Cl_{cp})RP]Cl$ - PK_{Cl} = $S_i(Cl_{cp})RP$*)*. This process further ensures the confidentiality of capability token. A capability token is breached if client cannot decrypt the message.
8. Client then uses resource provider public key to generate the capability token from the signed message ($S_i(Cl_{cp})RP$ - PK_{RP} = Cl_{cp}). This process further ensures that the client receives the capability token from the genuine resource provider unchanged.

Note that at the final step of registration process, client will have his capability token and public key which were retrieved from the resource provider. The capability token and resource provider public key will then be stored in client repository for future communication or resource access. On another end-point, the resource provider stores the client's public key in its own repository. We trust that the combination of both encryption and hashing mechanisms further uphold the confidentiality, integrity and non-repudiation of capability token during its transfer.

4.2 Fine-Grained Resource Access Workflow

Once a client has been successfully registered with the resource provider, client will present his capability token to the resource provider on every access request. The capability token which contains client assertions and authorization permissions is primarily used as a base by the resource provider for granting the resource access. Resource provider utilizes client's capability token to authenticate and authorize client access. Three foremost protection requirements for the resource access are the identification of resource provider, secured transfer of capability token, and authentication of a requesting client. A detailed workflow that ensures security protection on each resource access is provided in figure 2.
The steps are as follows:

1. Client looks at his repository for his intended resource provider capability token. He then retrieves this capability token from client repository. The capability token contains the client access permissions and the resource URL. At this stage, the client also determines a symmetric pass key which will be shared with the resource provider and generate the Authentication Token which consists of symmetric pass key and capability token (Cl_{cp} + SyK_{cl} = $ATCl$).
2. Client uses his private key to sign the capability token ($SKCl$ + $ATCl$ = $S_i(ATCl)Cl$). The signing process is essential to uphold the non-repudiation of capability token.
3. Client then encrypts the signed capability token using resource provider public key (PK_{RP} + $S_i(ATCl)$ = $E[S_i(ATCl)Cl]RP$*)* and he sends the encrypted message to the resource provider ($Cl:\{E[S_i(ATCl)Cl]RP\} \rightarrow RP$).
4. When resource provider received the encrypted message, it uses its own private key to de-crypt the message and retrieve signed capability token ($E[S_i(ATCl)Cl]RP$ - SK_{RP} = $S_i(ATCl)Cl$*)*.

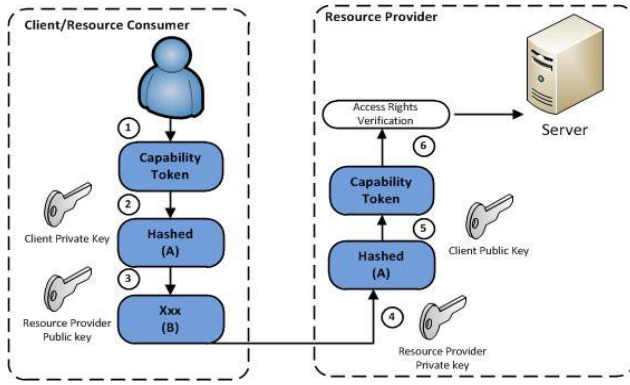

Fig. 2. DRPM resource access protection

5. Resource provider then verifies the signature of capability token using the client public key ($S_i(AT_{Cl})_{Cl} - PK_{Cl} = AT_{Cl}$). It then verifies the integrity of the capability token by generating the hash number from capability token using the SHA Algorithm.
6. Resource provider retrieves the access permissions listed in capability token.

Note that, on the step 1 of the workflow the client determines a symmetric pass key. This pass key will be utilized to generate a symmetric key for further communication after the capability token authentication and authorization processes is valid. In an event where the capability token is stolen due to the man-in-middle attack, the unauthorized entity will still not be able to access the resource due to the symmetric key passphrase that is shared between the legitimate client and resource provider only. If there is a security breach on which resource provider generates a new pair of public-private keys, client would not be able to decrypt using his current resource provider public key. Therefore, a request needs to be made to obtain a new public key.

PKI is extensively utilized during the DRPM resource workflow. The other party public key retained by both client and resource provider during the registration process is re-used to provide the confidentiality and integrity of capability token. PKI is primarily adopted during the initial handshake and capability token transfer. Due to the limitation of PKI which requires higher computation process, we suggest the utilization of symmetric key for transferring the data after the authentication and authorization process. The symmetric key can be incorporated into the capability token message before encrypting with the client's private key. Resource exchange is then encrypted by this symmetric key over the untrusted network.

6 Implementation Strategies and Scalability Testing

Due to the limitation on the length of this paper, this section provides a very brief review on the implementation strategies and the scalability performance of our proposed DRPM mechanism. Our DRPM prototype implementation was divided into two major applications: the resource provider application and the client application. The resource provider application was built of three main system components: listener

module, registration page and resource page. The main tasks of these components were to listen for any incoming connection from the client, to automatically create the client profile and capability token, to securely exchange the information, and to host multiple resources. In contrast, the client application was primarily being used to securely register and access the hosted resources.

We tested our prototype for the scalability on its listener server component to handle multiple *HttpWebRequest* requests. This test was conducted by utilizing the Apache JMeter [21] tool that specializes on the web scalability and performance testing.

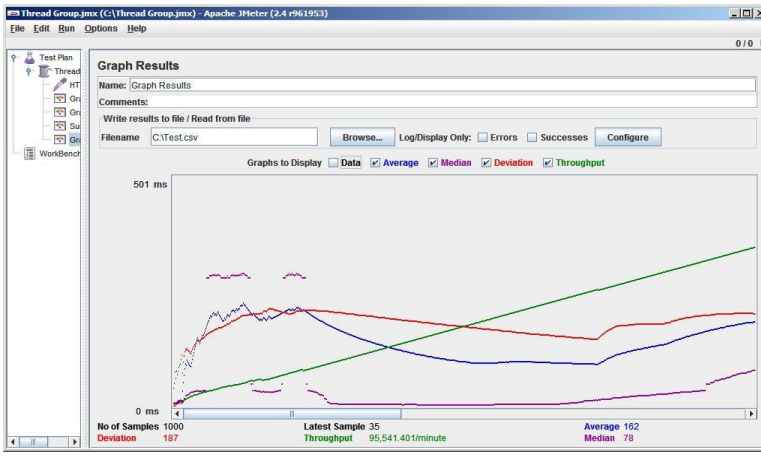

Fig. 3. DRPM listener component scalability testing

In our test bed, 1000 users were generated to access the listener component concurrently. Each user accessed the listener component for either registering or accessing the resources. Our test shows that the average elapsed time was 162 ms with the aggregate highest elapsed time of 327 ms for resource access process and the aggregate lowest elapsed time of 5 ms for registration process. It was shown that the highest elapsed time was primarily due to the encryption/decryption and hash verification of the capability token.

7 Conclusion

In this paper we have highlighted the needs for protecting enterprise resources from any unauthorized use in a Digital Ecosystem (DE) environment. Further, we have also analysed the appropriateness of several existing security mechanisms for DE. After a thorough analysis, we found a number of deficiencies of the current mechanisms to promote a strong community protection. Therefore, we propose the Distributed Resource Protection Mechanism (DRPM) for DE to provide a comprehensive resource protection. DRPM can be classified as a new approach to facilitate the authorization process for enterprises that request for specific resources or information. DRPM emphasizes on the decentralized authorization mechanism that is performed by each resource provider. It is achieved by utilizing the client profile and capability

token for its authentication and authorization permissions. Several future works such as analysis of DRPM and scalability of the prototype are needed to ensure a strong protection for DE member entities. Further, investigation on an effective trust mechanism to improve the overall DRPM security is critically needed. Our proposal incorporates the Web of Trust (WoT) to actively engage the community to protect the resources. As trust is critical in DRPM to build the confidence of the entities to interact and sharing their resources, a close analysis on the applicability of WoT to develop an effective trust management is desired.

References

1. Nachira, F., Dini, P., Nicolai, A.: A network of digital business ecosystems for Europe: roots, processes and perspectives, European Commission, Bruxelles, Introductory Paper (2007)
2. Dini, P., Darking, M., Rathbone, N., Vidal, M., Hernandez, P., Ferronato, P., Briscoe, G., Hendryx, S.: The digital ecosystems research vision: 2010 and beyond, European Commisssion, Bruxelles, Position Paper (2005)
3. Ballesteros, I.L.: 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. van Steen, M., Homburg, P., Tanenbaum, A.S.: Globe: a wide area distributed system. IEEE Concurrency 7, 70–78 (1999)
5. Czajkowski, K., Kesselman, C., Fitzgerald, S., Foster, I.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, HPDC-10 2001 (2001)
6. Koshutanski, H., et al.: Distributed Identity Management Model for Digital Ecosystems. Presented at the International Conference on Emerging Security Information, Systems and Technologies (Securware 2007), Valencia (2007)
7. Seigneur, J.M.: Demonstration of security through collaborative in digital business ecosystem. In: Proceedings of the IEEE SECOVAL Workshop, Athens, Greece (2005)
8. Hughes, J., Maler, E.: Security Assertion Markup Language (SAML) v. 2.0 Technical Overview, OASIS, Working Paper (2005)
9. Alliance, L.: Liberty Aliance Project (2011), http://www.projectliberty.org/
10. Seigneur, J.M.: Demonstration of security through collaborative in digital business ecosystem. In: Proceedings of the IEEE SECOVAL Workshop, Athens, Greece (2005)
11. Novotny, J.: An online credential repository for the Grid: MyProxy. In: Proceedings of the IEEE Tenth International Symposium on High Performance Distributed Computing (HPDC 2010), San Fransisco, USA (2001)
12. Chou, W.: Inside SSL: The Secure Sockets Layer Protocol. IEEE Computer Society: IT Professional (2002)
13. Pearlman, L., et al.: A Community Authorization Service for Group Collaboration. In: Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks, Monterey, USA (2002)
14. Thompson, M., et al.: Certificate-based access control for widely distributed resources. In: Proceedings of the 8th Conference on USENIX Security Symposium, Washington DC (1999)
15. Weise, J.: Public Key Infrastructure Overview, Sun BluePrints Online (2001)

16. Boley, H., Chang, E.: Digital Ecosystem: Principles and Semantics. Presented at the Inaugural IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2007), Cairns, Australia (2007)
17. Briscoe, G., Wilde, P.: Digital Ecosystems: Evolving Service-Oriented Architectures. In: Proceedings of the 1st International Conference on Bio Inspired Models of Network, Information and Computing Systems, New York, USA (2006)
18. Pranata, I., Skinner, G.: Managing enterprise authentication and authorization permissions in digital ecosystem. Presented at the 3th IEEE International Conference on Digital Ecosystems and Technologies (DEST), Istanbul, Turkey (2009)
19. Pranata, I., Skinner, G.: Digital ecosystem access control management. WSEAS Transactions on Information Science and Applications 6, 926–935 (2009)
20. Kennedy, J.: Distributed infrastructural service. In: Nachira, F., Dini, P., Nicolai, A., Le Louarn, M., Leon, L.R. (eds.) Digital Ecosystem Technology. European Commission: Information Society and Media (2007)
21. Apache JMeter (July 2011), http://jakarta.apache.org/jmeter/