©2010 International Journal of Computer Applications (0975 - 8887)
Volume 1 – No. 26

A New Collaborative Trust Enhanced Security Model for Distributed System

Aruna Kumari
CSED, MNNIT Allahabad
(INDIA)

Shakti Mishra
CSED, MNNIT Allahabad
(INDIA)

D.S. Kushwaha
CSED, MNNIT Allahabad
(INDIA)

ABSTRACT
Designing a distributed system with the characteristics of reliability and trustworthiness is an important issue. Yet another important issue in the distributed system is the access to remote system which can be achieved on the basis of certain access rights, policies or authorization semantics. The aim of this paper is to establish a collaborative trust enhanced security model for distributed system in which a node either local or remote is trustworthy. This paper also provides a promising solution with trust policies as authorization semantics. While designing a new secure distributed system, it has been observed that mostly the new nodes joining the system are insecure. If these perfidious nodes are provided full authorization, they can perform malicious activities in the system. In the proposed solution, node registry and service level agreements are used to ensure the trust for a new client node. Kerberos, a network authentication protocol is also used to ensure the security aspect when a client requests for certain services. In the proposed solution, we have also considered the issue of performance bottlenecks. A Reactive agent system is proposed to balance the load of service providers with the aim of enhancing the performance of the distributed system.

General Terms
Security

Keywords
Kerberos, Service level agreement, Capability list, Agent System

1. INTRODUCTION AND SURVEY OF RELATED WORK
One advantage of using distributed system includes the ability to connect remote users with remote resources in an open and scalable way. However, this scalability and openness leads to insecurity. Various organizations considerably depend upon effective use of distributed systems and the level of their protection and security. Many issues such as data storage, data transfer, automation of information processing and complex problems solving demand confidentiality and integrity in trusted environment [15]. Our primary focus is on the identity of the users in the system at each level such as at the entry level, at requesting a resource or at providing resources.

A distributed computing refers to the use of distributed systems to solve many problems where a problem is divided into many tasks, each of which may be solved by different user or different system. It also refers to autonomous processes that run on the same physical computer and interact with each other by message passing.

[7], has identified some fundamental challenges of security-aware distributed computing.

- The first challenge is to specify security requirements for a distributed activity.
- The second challenge is to design and implement distributed security-aware system, which can meet specific requirements of applications executing in it.
- The third challenge is to introduce a new set of performance metrics including the level of security. The level of security of a system can be presented by a formal measurement model based on the level of security of individual tasks.

Our paper conforms the authorization and authentication aspect of secure distributed system. Authorization is based on the concept of specifying access rights to a resource. Authorization collectively works with the concept of authentication which addresses how to determine the identity of nodes with a high degree of confidence. Trusted users that have been authenticated are often authorized to unrestricted access to resource. According to [18], this authentication and authorization approach uses password based methods to identify the identity of users. However, assertion based authentication schemes hardly qualifies as authentication at all since such authentication is easily thwarted by modifying the application. To solve this problem, stronger authentication methods based on cryptography are required. Kerberos is the most commonly used technique for this type of authentication methodology.

Kerberos is a time-tested and widely used lightweight protocol based on inexpensive symmetric key cryptography. Kerberos [10, 13] allows a user to authenticate once and then connect to application servers within the Kerberos realm without authenticating again for a period of time. It is a distributed authentication service that allows a client running on behalf of a user to prove its identity to a verifier (an application server, or just server) without sending data across the network that might allow an attacker or the verifier to subsequently impersonate the principal. It is a third party based authorization system. The predominance of Kerberos involves independent development platform, high speed communication of authentication, mutual authentication between entities and transferable relationship of trust, and a relatively strong compatibility with heterogeneous domains which may adopt various trust policies [4, 15].

The Kerberos server consists of an Authentication Server (AS) and a Ticket-Granting Server (TGS). The AS and TGS are responsible for creating and issuing tickets to the clients upon request. The AS and TGS usually run on the same computer, and are collectively known as the Key Distribution Center (KDC).

Security model for authentication and authorization based on PKI (Public Key Infrastructure) and Kerberos under a cross domain condition in distributed system is proposed in [4]. It focuses on the issue to establish trust relationship across heterogeneous domains due to different trust mechanisms and security policy. The author [4] also discusses the concept of "information isolate islands".

Trust is required everywhere in the distributed system but where to implement the trust policies is a main issue of any distributed system. Trust focuses on the security of utility [10]. It may be applied when a node joins the system as a client, or when it requests for the services, etc. If every access request is authorized, illegal information flow might occur as the well known confinement problem [5].

To understand the requirement of trust let us consider an example of a group of friends. D, a friend of A but not part of that group in which A is, wants to join the group. D requests A to convince others to join him. A introduces D as a new group member but the existing members are not ready to share their confidential matters with D. They permit D to join the group but decided not to share their matters until they found D as reliable.

Several researchers have worked in the area of trust for distributed system. Various economical models to characterize malicious behaviors in the security contact with the aim to mitigate the risk introduced by malicious behavior are discussed in [10]. Jaeger et al [11] emphasizes on the need of trust in the distributed system and the problem associated with it. The Author [11] also discusses various issues like misconfiguration or vulnerable software limits, enforcement of flexible distributed system security goals, and scalable solution for a single machine, while implementing trust in large scale distributed system. Trustworthiness is a holistic property, encompassing security (conventionally including confidentiality, integrity and availability), correctness, reliability, privacy, safety and survivability [15].

Trust Management, introduced by Blaze et al [3] is a unified approach to specifying and interpreting security policies, credentials, and relationships that allows direct authorization of security-critical actions. In particular, a trust management system combines the notion of specifying security policy with the mechanism for specifying security credentials [14].

At the heart of trust management systems is the authorization procedure, which determines whether resource access should be granted or not based on a number of conditions. The semantics of authorization provide meaning to the features supported by trust management systems, for both the policy maker and the resource requester. In a security setting, entities should be able to specify policies precisely, to have an absolute clear idea of the meaning of their policies, and to have confidence that they are correctly enforced by authorization mechanisms [3]. To ensure trustworthiness in the proposed trust model, we have considered different criteria like the history of the new client and server before making them a part of the distributed system.

The trusted intermediaries [16] are the systems which authenticate clients and servers such as the Certificate Authorities in public key based systems and KDC in Kerberos. With an increase in number of users that require authentication, trust intermediaries faces a challenge of scalability. If the challenge is not met, users can experience significant delays in authentication, or be forced to accept an increase risk or fraudulent credentials. The author in [16] describes a method for fully distributed authentication using public key cryptography within the Kerberos ticket framework and to enhance security and scalability by distributing most of the authentication workload away from the trusted intermediary and to the communicating parties is discussed in [16].

The author's approach in [10] shows that the decisions taken by security mechanisms, such as the authorization decisions in a distributed system can have a direct impact on the security of the underlying system. The trust-based theoretic model [10] possesses a unique feature- the ability to use trust evaluation to not only weed out malicious entities, but also allocate appropriate access permissions to the benevolent entities according to the risk levels. It integrates risk into security via trust and allocates a particular risk level for a given transaction, in order to maximize the utility gain obtained from honest and competent transactions from the trusted entities of the same system.

The interaction between nodes and resources allocation with trustworthiness dimensions is a central challenge in building a trustworthy distributed system. Since a new node may not only request network resources, it may also be interested in using a third party service; there services also need to be secured. The service is an exposed piece of functionality which can be dynamically located and invoked. The service provider creates a service and publishes its interface and access information to the service registry maintained by a system which is known to clients.

It has been observed that the violation of information security is based on the activities of the users of an organization. A solution to this problem is proposed in the Agent approach [9]. The activities are monitored and any abnormal activity is regarded as potential intrusion. An Agent based approach [9] envisages on-line and off-line monitoring in order to analyze users' activity. It relies on information confidentiality, integrity and accessibility. The traditional methods such as identification and authentication, access restriction, etc., approaches are basically rigorous and deterministic. There are many drawbacks associated with traditional models like low agility of internal malicious users' detection, inability to process large amounts of information, low productivity, etc. The Agent approach [9] uses neural networks to carry on-line monitoring and the analysis of statistical information obtained during user's work to carry off-line monitoring.

Although various issues have been resolved by many of the work done by previous researches, issues like the life time of the client and server in the distributed system, new semantics for trust based on identity, load balancing feature of controller systems for resource allocation still needs attention. In our work, we have made an attempt to propose a centralized approach to build the trust in the distributed system considering the workload of the systems.

The rest of the paper is organized as follows. Section 2 gives an overview and features of the proposed approach. Section 3 includes the components and design of proposed model. Conclusions and future work are given in section 4.

2. PROPOSED APPROACH

2.1 Introduction to Proposed Approach

To implement trust policies, is a major concern while designing trusted environment. In the proposed approach, the trust is maintained at the initial phase when a new node joins the distributed network. We have embodied the traditional security model with services as client has to authenticate before making a request. We have assumed that in every organization, there are some trustworthy nodes which perform coordination and management. The entire system is controlled by such trustworthy central coordinator which keeps the information about all nodes including number of nodes, number of resources and services present in the system.

The proposed approach enhances the performance of complete system by sharing the authentication workload among all other nodes. To accomplish this goal, we have devised an agent based system with various functionalities as monitoring the servers, balancing the load of service providers and service request in case of unavailability of servers.

The agent system in the proposed approach has reliability, autonomy, reactivity, proactiveness, social ability, and rationality features. It receives the client's request, checks its database for the requested service and the load status of the service providers and responds according to the status. It interacts with the clients and servers and according to their requirements it behaves. It has a goal directed behavior that is servicing other nodes in the system. The agent system is also designed to achieve the goal of improving the performance over time. For example, if agent notices the unavailability of service providers, it can itself serve the client's request.

To maintain the trust, each node is required to register itself to SuperHost which is a trustworthy system with highest trust level. Registering of a node does not mean that it is authorize to obtain all the services and resources of the distributed system. A client has to authenticate itself with KDC to get the reference of Agent system. The agent system is another trustworthy system which is authorized by the SuperHost. The security of the distributed system is enhanced by observing the activities and location of the client by SuperHost. The client may be a perfidious client who has joined the distributed system and wants to occupy more and more resources and services by behaving as a group of multiple users. A SuperHost system keeps the track of such clients and regulates their uncontrolled behavior.

2.1.1 Features of Proposed Approach

2.1.1.1 Assumptions

Some constraints are used for joining the client in the distributed system like client must have to share some resources such as processor, some basic services, etc., in order to access the existing resources and services of the distributed system. Client has to inform about duration of its availability in the distributed system to SuperHost. The load index of server is assumed in terms of number of clients associated with it.

2.1.1.2 Load Balancing

In the proposed model, an agent system is used to balance the load of service providers. Agent records the service registries and maximum load that a server can handle in terms of clients. Agent allocates the list of below loaded servers to client. Whenever a client requests for a service, agent provides the references of servers according to their load status. In the case of unavailability of server for a requested service, the agent provides service by itself, if it has any. Similarly, if all the servers are overloaded for the requested service, the agent system can play the role of server by itself.

2.1.1.3 Life time of Client and Server

The life time of client and server is controlled by SuperHost. However the client and server can also make an explicit leaving request to SuperHost. In the former case, at the time of joining the system, SuperHost informs client about its life time. If client remains idle for a predefined period of time, the SuperHost can retrench the client from the system implicitly assuming that client does not want to continue with this system. In the same way, if server does not make any attempt to modify or update its services for a fixed period of time, it will no longer be allowed to remain server.

3. PROPOSED COLLABORATED TRUST ENHANCED SECURITY (CTES) MODEL

3.1 Components of CTES Model

The components of proposed CTES model are defined as follow.

3.1.1 SuperHost

SuperHost is the trustworthy node which acts as central controller of systems. This system is also responsible to investigate the client about its trustworthiness, to monitor the life time of clients and servers. It provides various to users related to its operation such as record update, reference of Authentication Server (AS) to client and reference of agent to server respectively.

3.1.2 KDC

KDC (Key Distribution Center) is a collective name for AS (Authentication Server) and TGS (Ticket Granting Server) under Kerberos realm. Each client contacts AS to receive TGT (Ticket-Granting Ticket) and then TGS to receive a session key and a SGT (Service-Granting Ticket) before making a request for services.

3.1.3 Agent

The proposed agent system is assumed to be a trustworthy system. An agent system registers the services offered by servers, keeps load status of service providers, provides the list of servers on request of client, and serves the request of client in case of unavailability of service providers.

3.1.4 Client

Any new node registered by the SuperHost is termed as client node. Each node is required to send a request to SuperHost at the time of joining the system. A unique client_id has been assigned to each client by SuperHost. A client can make requests for services and resources in the distributed system for which it has to authenticate through the KDC.

3.1.5 Server

Server is a service provider which is authorized by the SuperHost to provide services in the distributed system.

3.2 CTES Model for Client

The following steps are used to maintain the trust and to access services of system by a client. Fig. 1 shows CTES model for clients.

3.2.1 Joining of a client

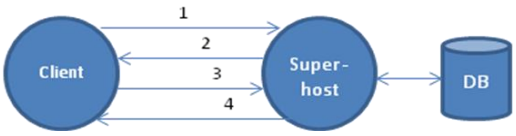

Fig 2: Joining of a client

If a new client has to register itself to SuperHost to be a part of the distributed system it has to pass certain criteria. Only registered client can request for the services in the distributed system as a trustworthy system. Registration process has certain steps (fig. 2).

1. Client → SuperHost

Client requests the SuperHost for registration by giving its identity and capability list. Capability list includes the list of resources and basic services owned by the node. SuperHost recognizes him as a client.

2. SuperHost → Client

SuperHost displays an agreement for the registration to the client which means that each client has to follow the rules and policies mentioned in the agreement. For this SuperHost demands the capability list of the client.

3. Client → SuperHost

Client provides capability list to SuperHost. The capability list contains a list of objects to which the process has right to access. In the proposed model, it consists of available resources and basic services that client holds.

4. SuperHost → Client

At the end of this process, SuperHost stores the reference and details of new client for the use by itself and other nodes of the system administration. SuperHost also provides a client_id to the client for its authentication. This client_id is the result of a crypto-based function (generally user defined).

3.2.2 Service request by a Client

Client can access the services of the distributed system in two steps to authenticate it and then to get the reference of service providers.

Step I: Authentication by KDC

1. Client → AS

The Client requests AS with its Client_id for TGT (Ticket Granting Ticket).

2. AS → Client

The AS checks to existence of the client from SuperHost. If it is, the AS returns a session key (for the communication from client to TGS) in encrypted form using the password of the client and Ticket-Granting Ticket which includes the Client_id, client network address, ticket validity period and client/TGS session key in encrypted form using the secret key of the TGS.

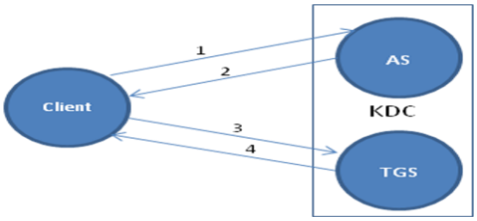

Fig 3: Service request by a client using Kerberos

3. Client → TGS

After receiving the reply from the AS, client decrypts the session key with its password. Here client sends the TGT to TGS.

4. TGS → Client

TGS decrypts the message with its own secret key and returns the reference of Agent system with a session key to access the service.

Step II: Request for references of servers to Agent

Case 1: (Service provided by server)

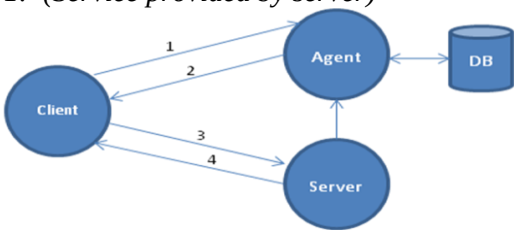

Fig 4(a): Request serviced by Server

1. Client → Agent

In the first step of this process, client makes a request to agent for some service with client_id and a valid session-id for authentication.

2. Agent → Client

Agent searches the requested service in its database and if it finds, it checks the load status of the server. Agent provides the reference according to the load status of servers, that is, under-loaded to client.

3. Client → Server

After receiving a list of servers for the requested service, client is free to make a new request to any of the server. Client sends client_id and service name in request message to destined server.

4. Server → Client

As server receives a request from any of the clients, it updates its load and also sends a load update message to Agent. Server provides service to the client and charges for the service.

Case 2: (Service provided by Agent)

1. Client → Agent
Client makes a request to agent for service with client_id and a valid session-id for authentication.

2. Agent → Client
Agent searches the requested service in its database and if it finds, it checks the load status of the service provider (which can service the requested service). In the case when agent does not find a single server (due to overload or unavailability of service), agent itself serves the service to the client, if it has any. In this case, agent itself will charge for the service.

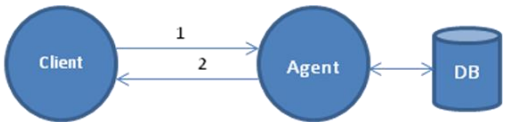

Fig 4(b): Request serviced by Agent system

3.2.3 Leaving of Client
In the proposed CTES model, a client can make an explicit request to SuperHost to unregister it or SuperHost can itself destroy the membership of client if it finds that the client has not been in touch for a long period of time.

3.3 CTES Model for Server

3.3.1 Joining of a Server
Joining of a server involves various steps.

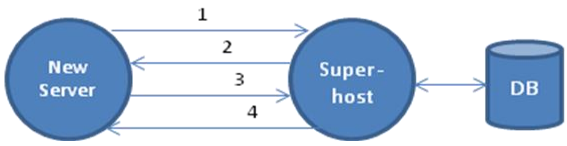

Fig 6: Joining of a new Server

1. New Server → SuperHost
When a node wants to be a server in the distributed system it makes a request by sending its client_id to SuperHost. SuperHost authenticates it by verifying valid client_id and accepts the request.

2. Superhost → New Server
In the next step, superhost shows a service level agreement (SLA). SLA, here, includes basic service and resources that server can provide, the duration for which services would be available, the number of users (clients) that can be served simultaneously by server, specific performance benchmarks to which actual performance will be periodically compared and response time. It specifies the level of availability, serviceability, performance, operation and other attributes.

3. New Server → Superhost
New server provides values to all these metrics of SLA.

4. Superhost → New server
As the result of this process, superhost maintains the record of the server for further reference and provides a server id to new server. New server can use this server_id to authenticate itself at the time of registering services.

3.3.2 Registering Services with Agent by New Server

Phase I (Fig. 7(a)):
In this phase a new server registers its first service with agent.

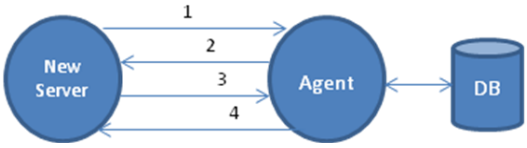

Fig 7(a): Service registration by new Server

1. New server → Agent
A server publishes it services with Agent by making a request that includes server_id.

2. Agent → New server
Agent demands details about the services from the new server.

3. New server → Agent
New server provides all the required details to agent to register a new service.

4. Agent → New Server
Agent maintains a record of the service registry provides service id to new server.

Phase II (Fig. 7(b)):
This phase is considered to enhance the performance of the agent.

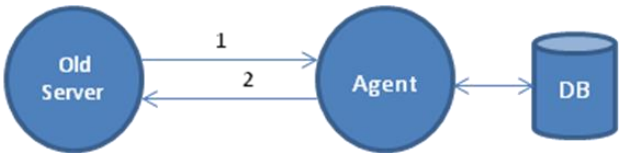

Fig 7(b): Service registration by old server

1. Server→ Agent
If a server which has registered at least one service with agent wants more service to register, it can directly call addService procedure of agent using existing service_id as a proof that it has already registered at least one service with the agent. Details of new services are also sent to agent.

2. Agent → Server
The called procedure will return a sequenced service id to the server as a result of new services registration phase. Server also stores this service id with service details in its record. This procedure can be used by an existing server 'n' number of times.

3.3.3 Request for service by Server
A server can itself require some services. In order to access the services, it can behave like a client in distributed system and make request to agent with its client_id only.

3.3.4 Leaving of Server
In the proposed CTES model, server can make an explicit request to superhost to unregister it or superhost can itself distroy the

membership of server if it finds that server has not been updated or modified its services for a long period of time.

4. CONCLUSION AND FUTURE WORK

This paper makes an attempt to build a trustworthy system using Kerberos extended by trust and its requirement in the distributed system. For this both authorization and authentication aspects are considered. The proposed approach provides a promising solution with trust policies as authorization semantics. A Reactive agent system is proposed in order to balance the load of systems and to maintain the list of services and servers. Distributing the authentication workload in an efficient manner enhances the overall performance of the system. In order to prevent the system from malicious nodes and their activities, the life time of client and server nodes is monitored by a central coordinator. Our model is scalable enough, means that system can easily be altered to accommodate the changes in the number of users, resources and computing entities. The proposed CTES model provides strong expandability and capability of mutual communication in a secure fashion. Our proposed CTES model can also be implemented for heterogeneous systems.

We are now implementing CTES model so as to evaluate the performance of overall system. In future, we shall deploy CTES model with web services on large scale.

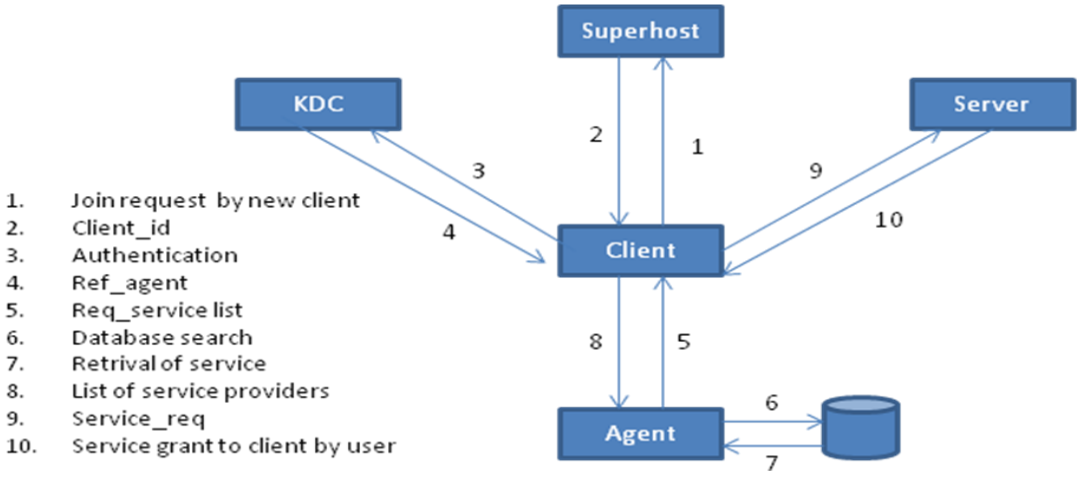

Fig 1: CTES model for Client

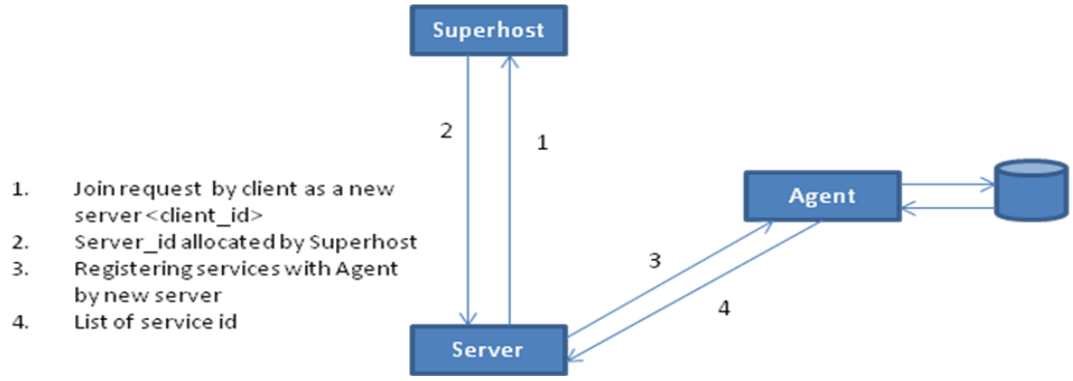

Fig 5: CTES model for Server

5. REFERENCES

[1] Ming He, Aiqun Hu and Hangping Qiu, " Research on secure key techniques of trustworthy distributed systems", in International Conference on Computer Engineering and Technology, 2009.

[2] Ming He, Aiqun Hu and Hangping Qiu, "Research on behaviour trust based on trustworthy distributed system", in International Conference on Networks Security, Wireless Communications and Trusted Computing, 2009.

[3] Peter C. Chapin, Christian Skalka and X. Sean Wang, "Authorization in trust management: features and foundations", in ACM Computing Surveys, August 2008.

[4] Ping Liu, Rui Zong and Sizuo Liu, "A new model for Authentication and Authorization across Heterogeneous

Trust-Domain" in International Conference on Computer and Software Engineering, 2008.

[5] Tomoya Enokido and Makoto Takizawa, "Role based access control in Distributed Object Systems", in International Conference on Distributed Computing Systems Workshps, 2008.

[6] Phillip L. Hellewell, Timothy W. van der Horst and Kent E. Seamons, "Extensible Pre-authentication in Kerberos", in Annual Computer Security Applications Conference, 2007.

[7] Hedi Hamdi, Mohamed Mosbah and Adel Bouhoula, "Domain specific language for securing distributed systems", in Second International Conference on Systems and Networks Communications, 2007.

[8] Huaizhi Li, Mukesh Singhal, "Trust Management in Distributed Systems,", in Computer, vol. 40, no. 2, pp. 45-53, 2007.

[9] Serhiy Skakun and Nataliya Kussul, "An agent approach for providing security in distributed systems", TCSET' 2006.

[10] Ching Lin and Vijay Varadharajan, "Trust based risk management for distributed system security- a new approach", in Proceedings of the First International Conference on Availability, Reliability and Security, 2006.

[11] Jaeger, T., McDaniel, P., St. Clair, L., Cáceres, R., and Sailer, "Shame on trust in distributed systems", in Proceedings of the 1st USENIX Workshop on Hot Topics in Security (Vancouver, B.C., Canada). USENIX Association, Berkeley, CA, 4-4, 2006.

[12] C.Neuman, T. Yu, S.Hartman, and K. Raeburn. RFC 4120: The Kerberos Network Authentication Service (V5), Jul 2005.

[13] Wen Tei-hua, Gu Shi-wem, "An improved method of enhancing Kerberos protocol security", Journal of China Institute of Communications, Vol 25 No. 6. June 2004, pp. 76-79.

[14] Matt Blaze, Joan Feigenbaum, John Ioannidis and Angelos D. Keromytis, " The role of trust management in distributed systems security", in Secure internet Programming: Security Issues For Mobile and Distributed Objects, J. Vitek and C. D. Jensen, Eds. Lecture Notes In Computer Science. Springer-Verlag, London, 185-21, 2001.

[15] Fred B. Schneider, Steven M. Bellovin and Alan S. Inouye, "Building trustworthy systems: Lessons from the PTN and Internet", IEEE Internet Computing, November- December 1999.

[16] Marvin A.Sirbu and John Chung-I Chuang, " Distributed authentication in Kerberos using public key cryptography", in sndss, pp.134, 1997 Symposium on Network and Distributed System Security, 1997.

[17] Nicholas Yialelis, Emil Lupu and Morris Sloman, "Role-based security for distributed Object Systems", in Proceedings of WET ICE, IEEE, 1996.

[18] B.Clifford Neuman and Theodore Ts'o, "Kerberos: an authentication service for computer networks", in IEEE Communications Magazine, 1994.

[19] Neuman C. RFC 1510, "The Kerberos network authentication service (V5)", [S]. 1993.