International Journal of Engineering Research and Development
ISSN: 2278-067X, Volume 1, Issue 11 (July 2012), PP. 30-35
www.ijerd.com

A Novel Distributed Network for Ensuring Highly Secure Proposed Enterprise Network Integrated Firewall

Santosh Kumar Mishra
(Research Scholar, NIMS University Rajasthan, India) & Prof. (Dr.) V. S. Rathore, *Jaipur, Rajasthan, India*.

Abstract— *A study on the current network enabled vulnerability checking software was made. It was found that most of the available solutions had a some defect or the other in implementation of the applications as there was a large delay in the assessment cycle. This was due to the inherent fact that the complete application had to scanned bit by bit and checked for byte length and code compared with the byte address of the sender and a check is made at the firewall for the word by word comparability. In this paper we discuss an active analysis of the system, for any weakness or technical flaws in the operating system or the applications which as running on them. The security solution followed the CORBA architecture.*

Keywords—*Enterprises LAN, vulnerability, security, operating system, CORBA with ORB.*

I. INTRODUCTION

Today's networks must be secure. This can only be achieved by constant effort and vigilance over time. Because security is a process, not a product, it begins and ends with people. The creation of a secure network affects many facets of an organization's business practice. The process includes sound security policies & issues, thorough implementation, diligent monitoring and extensive education to the user.

As networks are built in layers, security must exist between each of these layers, as shown in Figure 1. A layered approach to network security allows organizations to create multiple levels of defense around key assets. The result is security-in-depth in which a breech at one layer does not compromise the entire network. This is the motivation behind the Alcatel information security framework.

.[1]

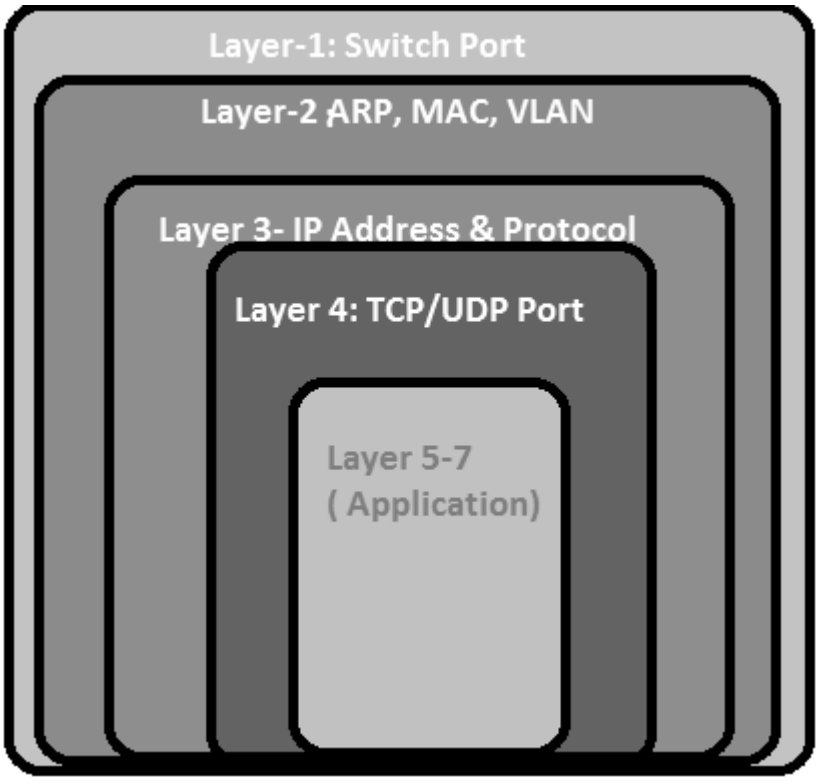

Figure: 1 **Layer of Vulnerabilities Assessment**

To make things worse, while the system user needs to patch all possible vulnerabilities in his/her system, a hacker only needs to locate any one to break in.

Formal verification approach can only provide validation of software against vulnerabilities at abstract level. With increase in number of systems over the network, this approach becomes almost ineffective. As the number of system

vulnerabilities multiply in recent years, Hence V*ulnerability Assessment tools* that can identify vulnerabilities in existing systems be for e actual exploitation takes place have become immensely important.

Vulnerability assessment is a method of evaluating the security of a computer system or the Enterprise network. The process involves an active analysis of the system for any weaknesses or technical flaws in the OS or the applications which are running on them. These are known as vulnerabilities. The vulnerability assessment cycle as shown in figure 2 comprise of the following stages: [2]

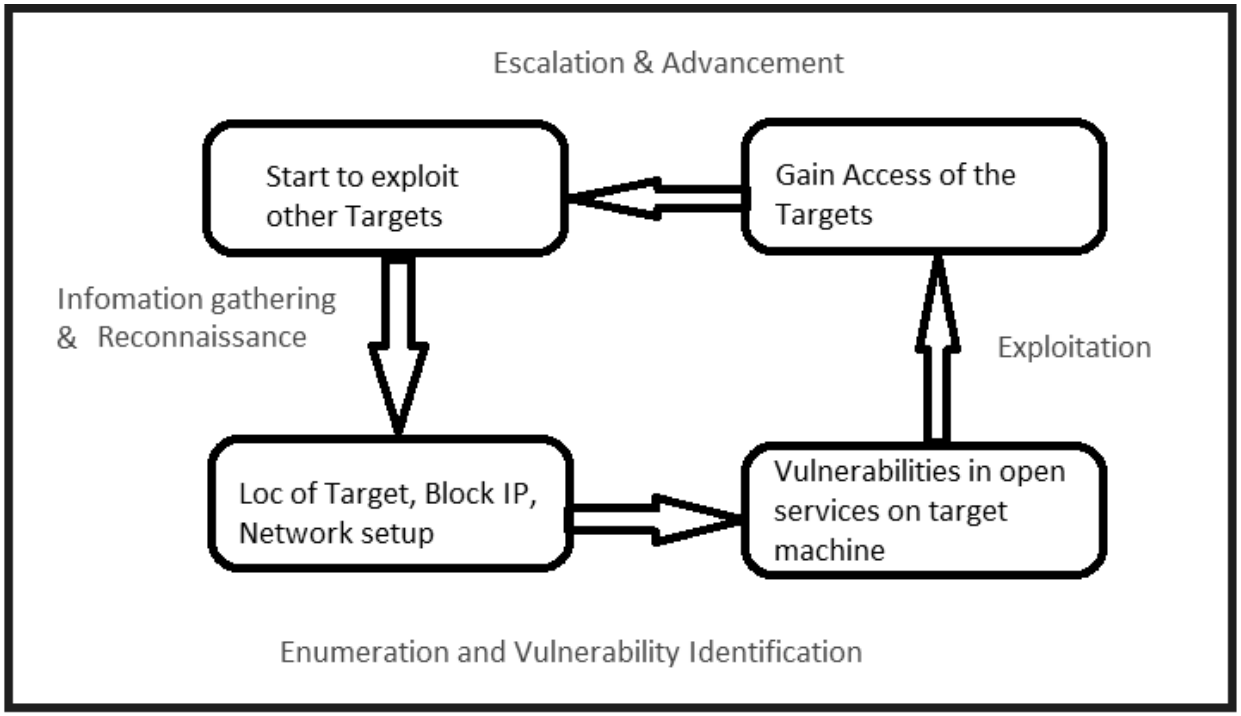

***Figure: 2* A typical vulnerability assessment cycle**

II. LITERATURE SURVEY

Lot of work has been done on vulnerability assessment, particularly relating to improving the accuracy of vulnerability identification methods and remote system[3] identification Vulnerability Assessment software is designed to assist network and system administrators in identifying networked hosts vulnerable to compromise. This makes identifying risk and prioritizing software updates much easier for administrators. Typically, enterprise-class vulnerability assessment falls into one of two categories:-

(1) Host-Based (2) Network-based Systems.

While host-based systems require software on each system that will be analyzed (referred to as *targets* in this paper), network-based vulnerability assessment software analyzes remotely systems.

This paper will focus exclusively on network-based vulnerability assessment software.
Key challenges to network-based vulnerability assessment software can be broken down into the following categories:-

- *scanning*
- *storage*
- *reliability*
- *Dissemination.*

CORBA as a standard protocol supersedes ORB and other peer network protocols likes java based JSB, BOSS, SPRING, SSP and RARP (Reversed Access Router Protocol) and DHCP (Dynamic Host Control Protocol). The larger environment in which a vulnerability assessment solution is deployed, the more difficult meeting these challenges becomes. [5]More networked hosts means a more heterogeneous pool of systems to scan and more time to scan all systems, more data to store in the database, and more individuals to whom data should be disseminated. With all of these increased complications comes a higher likelihood of problems.

III. PRESENT SOLUTIONS

Modern vulnerability assessment systems address challenges with a three-tier approach: a variable number of *scanners* that perform assessments,[6] a *database* at the back-end, and *application server* in the middle to coordinate scans, interface with the database, and provide an interface through which users can administer and control the system. More robust

solutions allow the addition of any number of scanners, enabling parallel scanning of targets in larger environments. Figure 3 illustrates the relationship between these components.[5,7]

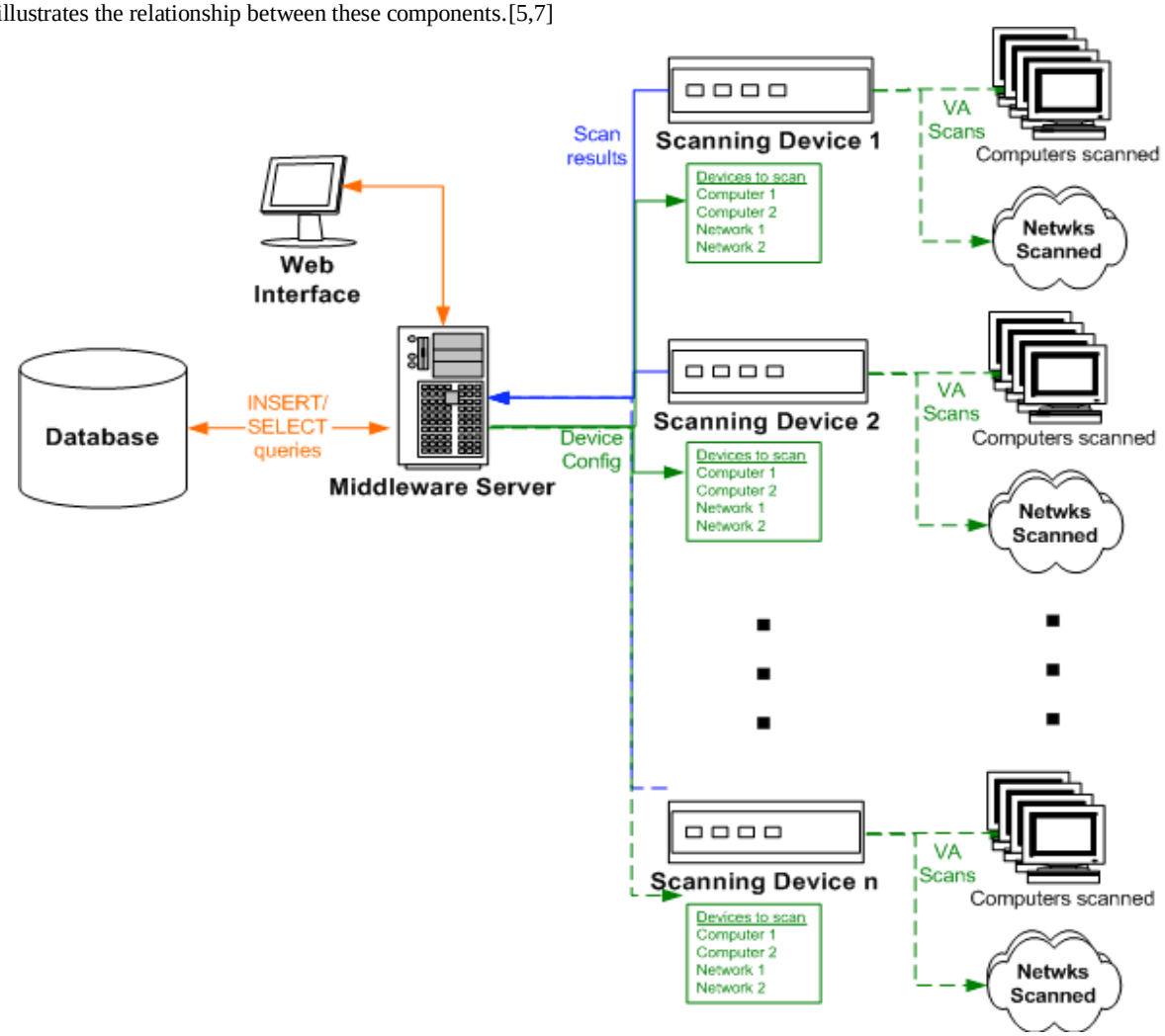

Figure 3: The Three Tiers of Current Vulnerability Assessment Solutions

Database construction and maintenance are problems that have been researched and solved. For this reason, vulnerability assessment solutions should (and nearly always do) use a database provided by another vendor. This permits designers to focus on the application specific problems presented by the design and maintenance of the scanners and middleware server.

IV. UNRESOLVED PROBLEMS

Because databases are essentially a solved problem, the most significant outstanding issues in current vulnerability assessment solutions affect the two remaining tiers. Specifically, problems in the accuracy & efficiency of scan results and reliability of the overall system remain outstanding. The focus of this paper will be on the area in which proper implementation of component architecture such as CORBA can assist: improving the efficiency of scan results and reliability of the system. Current industry-leading products divide the pool of hosts to be scanned amongst available scanners by IP address. That means the smallest atomic task handed off to a scanner includes all vulnerability scans available for a single IP address. These tasks are then assigned to scanners manually by administrators of the system. This approach leads to an inflexible system that cannot adapt when problems occur. In large, distributed networks, anomalies like latency and packet loss are a very real problem.[9.10]

Also, older or very busy systems can respond slowly to queries by vulnerability assessment scanners. Larger problems relating to the system's inability to move scan tasks between scanners can occur if one of the scanners experiences a problem or becomes unavailable. In this case, *none* of the scan tasks assigned to the problematic scanner will be executed, resulting in a gap in data. Furthermore, adding to or removing from the pool of available scanners requires significant maintenance, as IP's need to be manually re-distributed.[11]

V. DESIGN OF A NOVEL ASSESMENT ALGORITHM

Re-designing the VA system to dynamically assign scan tasks to scanners based on availability would fix the problems introduced by latent scans and unavailable scanners. Decreasing the size of each scan task would enhance these improvements by increasing the speed and evenness with which the system could reassign tasks.

CORBA (Common Object Request Broker Architecture) applications fall into one of two types: *clients* invoking operations on objects, and *servers* processing operations and returning the results. In order for a CORBA application to function, all objects on the server that provide operations for clients to invoke must have an IDL defined[5]. The same IDL is used by the client to invoke operations on objects, as well as the server to receive requests and return the results of the invoked operations. However, the details of how the operation is executed are hidden from the client; all it sees is the interface. Figure 3 (labeled in the image as "Figure 1") illustrates how these components work together. *IDL Stub* and *IDL Skeleton* are terminology used for the client and server sides of the IDL, respectively. The Object Request Broker, or ORB, handles not only the delivery of the request from client to server, but also uniquely identifies the request so that the server invokes an operation on the proper object, not an object that has been previously requested on by another client (or another thread on the same client).

The process of creating an IDL and invoking operations on an object with CORBA
The object that will be worked with will be a simple integer. The client will be allowed to set the integer value of the object (which resides on the server), and get the integer value of the object. This is not a very useful example, but it illustrates all of the key functions necessary to implement and use CORBA.3

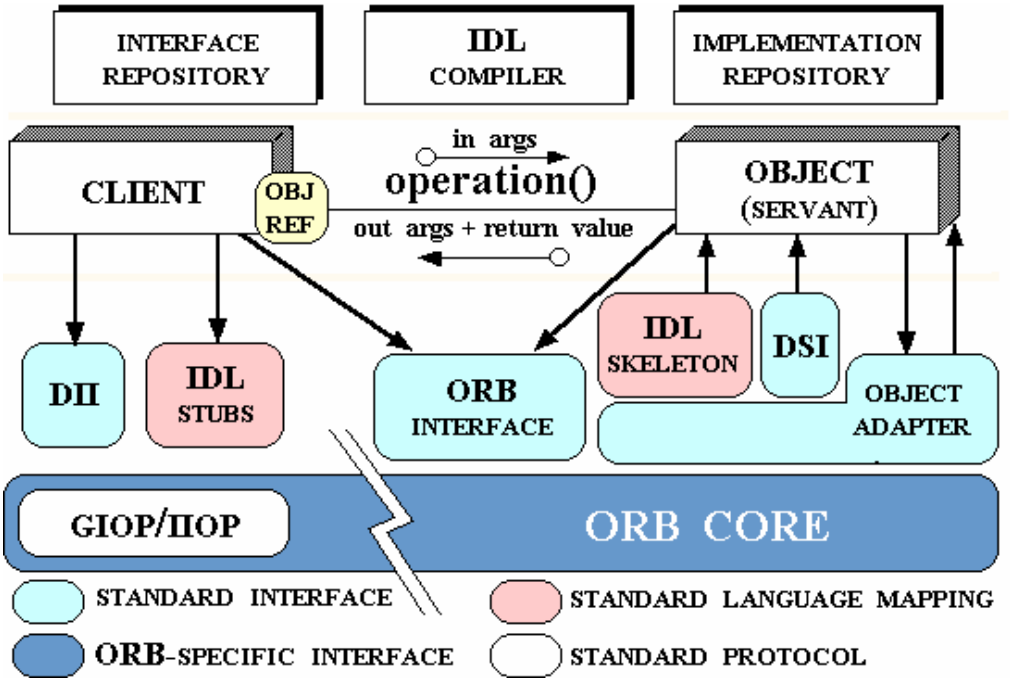

Figure. 4: illustration of a Generic Object Request [5]

Algorithm 1
1: Input: directed graph G = (V;E), constraint set M = fc1; _ _ _ ; cmg, credit vector (_1; _2; _ _ _ ; _m), satisfactory score threshold _;
2: Output: solution set of edge of QoSCE.
3: S all the minimal combinations ss of M with
P
ci2ss _i > _;
4: for each edge (i; j) 2 E do
5: Set f(i; j) = f(j; i) = 0;
6: Set cf (i; j) = 1 and cf (j; i) = 0.
7: end for
8: while S 6= ; do
9: ss extracted from S;
10: while 9q the shortest path satisfying all the constraints in ss do
11: for each edge (u; v) 2 q do
12: cf (q) = minfcf (u; v) : (u; v) 2 qg;
13: f(u; v) = f(u; v) + cf (q); f(v; u) = □f(u; v);
14: cf (u; v) = c(u; v) □ f(u; v); cf (v; u) = c(v; u) □ f(v; u);

15: end for
16: end while
17: end while
18: all the vertices reachable from s on the residual network induces a cut T .
19: Return T .

Algorithm 2 statistical Accessibility and translation test
1: Input: directed graph G = (V;E), constant _;
2: Output: YES if a satisfactory path probably exists, NO otherwise.
3: for every edge e 2 E do
4: '(e)
Pm
i=1
we
i
ci
_i;
5: end for
6: p shortest s-t path on metric ';
7: if '(p) > _ then
8: Return NO;
9: else
10: Return YES;
11: end if

VI. A TEST FOR VULNERABILITY ASSESSMENT

It has been shown that one of the key components is the CORBA interface definition, and properly using the definition in the IDL Stub and IDL Skeleton. The focus of this section will be on the interface definition and a simplified version of the IDL Stub and IDL Skeleton, building on the previous example. The details of the software on the scanner are intentionally omitted; it is assumed that this is a previously-solved problem. The scanIDL listed below takes into account the new atomic scan task, defining requests as a single vulnerability check against a single IP address.

The two interfaces that need to be provided to the server's scan object (not mentioned previously) are *get_os* and *get_services*. These two tasks are necessary for the client to determine which scans to run against a particular host, and are functionally quite different from probing a single vulnerability. As can be inferred from the IDL, each of these is treated as a single scan task. Also note that the list of operating system types in the IDL is abbreviated. An accurate listing of all OS types and subtypes would be quite lengthy, and specific to what the scanner has the capability to detect. The return value for *get_services* is an array of integers. Each integer will represent a TCP or UDP port that has been identified by the scanner as providing a listening "service" (TCP port 80 on an web server, for example). Note that the vulnerability status is returned by *scan_vuln*, along with the host's response that caused the scanner to set or unset the *isVulnerable* booelan. Often, this information is needed by users to identify incorrectly-reported vulnerabilities, or other troubleshooting purposes. The *vuln_index_number* parameter assumes that the client and server both have identical, indexed lists of all vulnerability checks available. The server will only discussed insofar as its interface with the CORBA ORB. As a result, the server implementation has been significantly over-simplified. In the code below, is assumed that the *scan_functions.h* class contains C++-equivalent definitions of the types *os_t*, *ip_prot_t*, and *scanResult_t* used in the IDL, as well as the functions *get_os*, *get_services*, and *scan_vuln*. It is also assumed that the scan object type is *scan_t*.
//server.cpp

VII. A NEW CORBA BASED MODEL FOR SECURE NETWORKING SOLUTIONS

With the scan task redesigned for automatic, dynamic delegation to a scanner by the middleware server, middleware and scanner software re-designed to use CORBA for requesting scan objects, and multithreading implemented so that scans run concurrently on scanners upon request from the middleware, the vulnerability assessment system effectively uses all of its available resources to process queues of scan tasks.

Scanners are unavailable; the client requesting scans isn't affected other than receiving thrown exceptions after fewer scan objects have been requested by the POA. The ORB handles the communication from client to the available servers, so when one goes offline; requests are simply not routed to it. Similarly, when scanners are added, the only difference seen by the client is the capability to request more scan objects before an exception is thrown. The solution is elegant and adaptable.

CORBA automates many common network programming tasks such as object registration, location, and activation; request demultiplexing; framing and error-handling; parameter marshalling and demarshalling; and operation dispatching. The ORB provides a mechanism for transparently communicating client requests to target object implementations. The ORB simplifies distributed programming by decoupling the client from the details of the method invocations. This makes client requests appear to be local procedure calls. When a client invokes an operation, the ORB is

responsible for finding the object implementation, transparently activating it if necessary, delivering the request to the object, and returning any response to the caller.

Now CORBA with ORB would yield a strong base and better performance.

VIII. CONCLUSION

A study on the current network enabled vulnerability checking software was made. It was found that most of the available solutions had a some defect or the other in implementation of the applications as there was a large delay in the assessment cycle. This was due to the inherent fact that the complete application had to scanned bit by bit and checked for byte length and code compared with the byte address of the sender and a check is made at the firewall for the word by word comparability (using byte counting algorithms techniques Example byte check metrics methods). This could be corrected by using a multithreaded CORBA code which could scanned multiple lines of code of at a time integrated check metrics and byte count methods built into it.

The number of bytes sent by the sender was converted into a JAVA thread (CORBA enabled). This thread compared the reminder byte of the divisor algorithm with a gray bit of code in received message. A change in the above thread length are value was counted and such value computed for the whole message length and the division from a average value was determined. To ensure that the error was with the value of the standard division which was fixed by statically error limit computation using allowable error of five percent. Next a bit normalized bit count and word length count taken for conforming the accuracy of the received message. This reduced the scanned time and the load on the checking algorithm. Which decreased and minimized time delay without compromised on the security of the solution are application. Hence multi thread CORBA (equivalent to JAVA network thread and JNS) was significantly faster and completely secured. We conclude that multithreaded CORBA algorithms development and trail implementation done in our work was much better and occupied lesser network memory. The simulation of these algorithms was done using NS2 and CISCO simulation software.

REFERENCES

[1]. B. Howard, O. Paridaens, B. Gamm: "Information Security: Threats and Protection", Alcatel Telecommunications Review, 2nd Quarter 2001, pp 117–121.
[2]. Automated Security Checking and Patching Using TestTalk by Chang Liu Debra J. Richardson. Information and Computer Science, University of California, Irvine 2000 IEEE.
[3]. Fyodor, Nmap Network Mapper ,
[4]. *The Three Tiers of Current Vulnerability Assessment Solutions*
[5]. Krishna, Arvind S., Schmidt, Douglas C., Klefstad, Raymond, Enhan zdycing Real-time CORBA via Real-time Java Features,
[6]. Object Management Group, CORBA FAQ,
[7]. University of Alabama Department of Computer Science, Integrating Component-Based So ware Development into the Computing Curriculum,
[8]. Henning, Michi, and Vinoski, Steve, Advanced CORBA Programming with C++ , Addison-Wesley, 1999.
[9]. Object Management Group, CORBA Language Mapping Specifi cations,
[10]. Common Object Request Broker Archictecture, OMG, July, 1995.
[11]. Common Object aServices Specification, OMG 95-3-31, 1995.