INTRUSION DETECTION SYSTEM FOR GRID AND CLOUD COMPUTING

Explanation on it's working and construction

4/7/2011

L0rd CrusAd3r
Lord.v5111@gmail.com

Dedicated to My Friends

P.Sivasankaran

T.R.Venkatesh

R.Harikrishnan

INTRODUCTION

SYSTEM ANALYSIS

In these days a single server handles the multiple requests from the user. Here the server has to process the all the requests from the users simultaneously, so the processing time will be high. This may leads to loss of data and packets may be delayed and corrupted. On doing this the server cannot process the query from the user in a proper manner.

So the processing time gets increased. It may leads to traffic and congestion. To overcome these problems we are going for the concept called “cloud computing”. In this cloud computing we are going to implement the Proxy server to avoid these problems.

But in this system Data Efficiency is improved but not the data security. Whenever we speak all about data efficiency we should speak about data security also, because in the cloud computing we don’t know from which cloud the data is coming, so in the existing system there is no system to find the data security.

The system based on the new architecture has better scalability and fault tolerance. A cluster consists of a single server and multiple proxy servers and is accessed by multiple clients. Proxy servers stores data on local disks and read or write data specified by a server. The server maintains the index for all file stored in different proxies. When a client wants to download some data, it will first send a request to the Server and the Server then redirect the request to a corresponding proxy that have the required data and hence the data will be sent to the client. With the combination of Cloud and Grid computing concepts, the data request can be efficiently serviced in a timely manner.

The major part of the Project is Security, so above mentioned phase speaks all about Cloud & Grid Technology, but not about security. The Security implementation in this project is achieved by two phase, namely

- Behavioral
- Knowledge

BEHAVIOR ANALYSIS

Using this method, we need to recognize expected behavior (legitimate use) or a severe behavior deviation. The network must be correctly trained to efficiently detect intrusions. For a given intrusion sample set, the network learns to identify the intrusions. However, we focus on identifying user behavioral patterns and deviations from such patterns. With this strategy, we can cover a wider range of unknown attacks.

KNOWLEDGE ANALYSIS

Using an expert system, we can describe a malicious behavior with a rule. One advantage of using this kind of intrusion detection is that we can add new rules without modifying existing ones.

IMPLEMENTATION STEPS

1. User sends the query to the Main Cloud Server, to which all the Cloud Servers (proxy servers) are connected.
2. The main Cloud server will have the index information of the data information by all the Cloud Servers (proxy servers).
3. After receiving the request given by the client, the main Cloud server will verify the exact proxy server which can carry the work more efficiently. By this process we are implementing both Cloud and Grid Computing together.
4. But in the cloud computing, plenty of Hackers may intrude the data, so we are working on Intrusion Detection system.

5. Intrusion Detection system is carried by two process,
   - Knowledge
   - Behavior
6. Knowledge is done by violating the set of rules.
7. Behavior is done by comparing the previous Behavior with the present Behavior.
8. If the system finds the there is Intrusion attack has occurred, it should alert other Cloud Server, so that data access is prevented.

FEASIBILITY STUDY

Feasibility studies aim to objectively and rationally uncover the weaknesses of the existing system and the strengths of proposed system, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success

OPERATIONAL FEASIBILITY

Since we use Java as the base platform to develop the project, interoperability is high and hence the operational feasibility is achievable. It can be implemented in all kind of environments as stated above so the constrains are found to be less and we can implement it in all LAN and WAN environments. The proposed system also tends to solve the data pipeline effect which is most commonly found in network

TECHNICAL FEASIBILITY

All the technical aspects of the proposed system is achievable under any circumstances, the minimum requirements of hardware and open source software makes the user to achieve the goals exactly. When it is developed as an application to implement in WAN environment it provides high technical support to the users regardless of the operating system, The interoperability is high and the demonstrated goals are achievable with the technical details chosen

The cache data , cache path and hybrid cache are the three main aspects which makes the optimal cache in a organized way even when there is a congestion over the network . Java supports both the developer and user giving a good support with database and front end

SOFTWARE DESCRIPTION

TECHNOLOGY DESCRIPTION

- JAVA JDK1.5
- MS SQL SERVER
- Apache tomcat

FEATURES

a .Java

It is a Platform Independent. Java is an object-oriented programming language developed initially by James G+osling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

b. Working of java

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it looks for a particular method by the name of main, which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv[] of C), and is declared as a static method.

To output text from the program, we execute the println method of System.out , which is java's output stream. Unix users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user's program.

Java consists of two things:

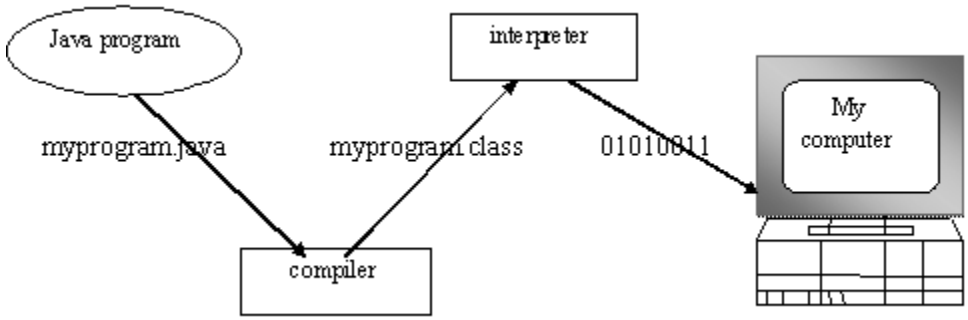
- Programming language
- Platform

c. The Java Programming Language

Java is a high-level programming language that is all of the following:

- Simple
- Object-oriented
- Distributed
- Interpreted
- Robust
- Secure
- Portable
- Multithreaded
- Dynamic

Java is unusual in that each Java program is both complied and interpreted. With a compiler, you translate a Java program into an intermediate language called Java byte codes – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works,



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (JVM)**.** Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of JVM. That JVM can also be implemented in hardware. Java byte codes help make "write once, run anywhere" possible.

You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM. For example, that same Java program can e run on Windows NT, Solaris and Macintosh.

Java is one of the first programming languages to consider security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind. The compiler, interpreter, and Java-compatible browsers all contain several levels of security measures that are designed to reduce the risk of security compromise, loss of data and program integrity, and damage to system users.

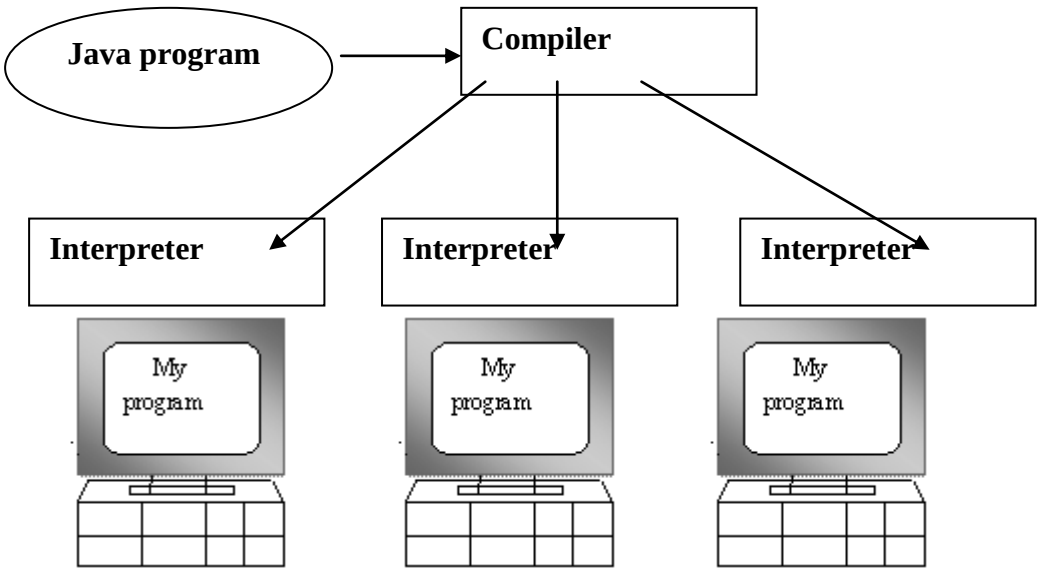
d. The Java Platform

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components:

- The Java Virtual Machine (JVM)
- The Java Application Programming Interface (Java API)

You've already been introduced to the JVM. It's the base for the Java platform and is ported onto various hardware-based platforms



You've already been introduced to the JVM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (packages) of related components. The following figure depicts a Java program, such as an application or applet, that's running on the Java platform.

As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

As a platform-independent environment, Java can be a bit slower than native code. However, smart compliers, wheel-tuned interpreters, and just-in-time byte compilers can bring Java's performance close to that of native code without threatening portability.

e. Structured Query Language

About SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL is an ANSI (American National Standards Institute) standard

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

RDBMS

- RDBMS stands for Relational Database Management System.
- RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- The data in RDBMS is stored in database objects called tables.

- A table is a collection of related data entries and it consists of columns and rows.

f. DML and DDL

SQL can be divided into two parts: The Data Manipulation Language (DML) and the Data Definition Language (DDL).

The query and update commands form the DML part of SQL:

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database

The DDL part of SQL permits database tables to be created or deleted. It also define indexes (keys), specify links between tables, and impose constraints between tables. The most important DDL statements in SQL are:

- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

ARCHITECTURE

a. Protocol Layer

Protocol layer implements the external interface to SQL Server. All operations that can be invoked on SQL Server are communicated to it via a Microsoft-dsequently, access to SQL Server is available over these protocols. In addition, the SQL Server API is also exposed over web services

b. Data Storage

The main unit of data storage is a database, which is a collection of tables with typed columns. SQL Server supports different data types, including primary types suchas *Integer*, *Float*, *Decimal*, *Char* (includingcharacterstrings), *Varchar* (vari able length character strings), binary (for unstructured blobs of data), *Text* (for textual data) among others. The rounding of floats to integers uses either Symmetric Arithmetic Rounding or Symmetric Round Down (*Fix*) depending on arguments: SELECT Round(2.5, 0) gives 3.

Microsoft SQL Server also allows user-defined composite types (UDTs) to be defined and used. It also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). In addition to tables, a database can also contain other objects including views, stored procedures, indexes and constraints, along with a transaction log. A SQL Server database can contain a maximum of objects, and can span multiple OS-level files with a maximum file size of The data in the database are stored in primary data files with an extension .mdf. Secondary data files, identified with a .ndf extension, are used to store optional metadata. Log files are identified with the .ldf extension.

Storage space allocated to a database is divided into sequentially numbered *pages*, each 8 KB in size. A *page* is the basic unit of I/O for SQL Server operations. A page is marked with a 96-byte header which stores metadata about the page including the page number, page type, free space on the page and the ID of the object that owns it. Page type defines the data contained in the page - data stored in the database, index, allocation map which holds information about how pages are allocated to tables and indexes, change map which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text. While page is the basic unit of an I/O operation, space is actually managed in terms of an *extent* which consists of 8 pages. A database object can either span all 8 pages in an extent ("uniform extent") or share an extent with up to 7 more objects ("mixed extent"). A row in a database table cannot span more than one page, so is limited to 8 KB in size.

However, if the data exceeds 8 KB and the row contains *Varchar* or *Varbinary* data, the data in those columns are moved to a new page (or possibly a sequence of pages, called an *Allocation unit*) and replaced with a pointer to the data.

c. Buffer Management

SQL Server buffers pages in RAM to minimize disc I/O. Any 8 KB page can be buffered in-memory, and the set of all pages currently buffered is called the buffer cache. The amount of memory available to SQL Server decides how many pages will be cached in memory. The buffer cache is managed by the *Buffer Manager*. Either reading from or writing to any page copies it to the buffer cache. Subsequent reads or writes are redirected to the in-memory copy, rather than the on-disc version. The page is updated on the disc by the Buffer Manager only if the in-memory cache has not been referenced for some time. While writing pages back to disc, asynchronous I/O is used whereby the I/O operation is done in a background thread so that other operations do not have to wait for the I/O operation to complete. Each page is written along with its checksum when it is written.

When reading the page back, its checksum is computed again and matched with the stored version to ensure the page has not been damaged or tampered with in the meantime

d. Logging and Transaction

SQL Server ensures that any change to the data is ACID-compliant, i.e. it uses transactions to ensure that the database will always revert to a known consistent state on failure. Each transaction may consist of multiple SQL statements all of which will only make a permanent change to the database if the last statement in the transaction (a COMMIT statement) completes successfully. If the COMMIT successfully completes the transaction is safely on disk.

SQL Server implements transactions using a write-ahead log.

Any changes made to any page will update the in-memory cache of the page, simultaneously all the operations performed will be written to a log, along with the transaction ID which the operation was a part of. Each log entry is identified by an increasing *Log Sequence Number*(LSN) which is used to ensure that all changes are written to the data files. Also during a log restore it is used to check that no logs are duplicated or skipped. SQL Server requires that the log is written onto the disc before the data page is written back. It must also ensure that all operations in a transaction are written to the log before any COMMIT operation is reported as completed.

At a later point the server will *checkpoint* the database and ensure that all pages in the data files have the state of their contents synchronized to a point at or after the LSN that the checkpoint started. When completed the checkpoint marks that portion of the

log file as complete and may free it (see Simple transaction logging vs Full transaction logging). This enables SQL Server to ensure integrity of the data, even if the system fails.

On failure the database log has to be replayed to ensure the data files are in a consistent state. All pages stored in the roll forward part of the log (not marked as completed) are rewritten to the database, when the end of the log is reached all open transactions are rolled back using the roll back portion of the log file.

e. Concurrency and Locking

SQL Server allows multiple clients to use the same database concurrently. As such, it needs to control concurrent access to shared data, to ensure data integrity - when multiple clients update the same data, or clients attempt to read data that is in the process of being changed by another client. SQL Server provides two modes of concurrency control: pessimistic concurrency and optimistic concurrency. When pessimistic concurrency control is being used, SQL Server controls concurrent access by using locks. Locks can be either shared or exclusive. Exclusive lock grants the user exclusive access to the data - no other user can access the data as long as the lock is held. Shared locks are used when some data is being read - multiple users can read from data locked with a shared lock, but not acquire an exclusive lock. The latter would have to wait for all shared locks to be released. Locks can be applied on different levels of granularity - on entire tables, pages, or even on a per-row basis on tables. For indexes, it can either be on the entire index or on index leaves.

f. Data Retrieval

The main mode of retrieving data from an SQL Server database is querying for it. The query is expressed using a variant of SQL called T-SQL, a dialect Microsoft SQL Server shares with Sybase SQL Server due to its legacy. The query declaratively specifies what is to be retrieved. It is processed by the query processor, which figures out the sequence of steps that will be necessary to retrieve the requested data. The sequence of actions necessary to execute a query is called a query plan. There might be multiple ways to process the same query. For example, for a query that contains a join statement and a select statement, executing join on both the tables and then executing select on the results would give the same result as selecting from each table and then executing the join, but result in different execution plans. In such case, SQL Server chooses the plan that is expected to yield the results in the shortest possible time. This is called query optimization and is performed by the query processor itself.

SQL Server includes a cost-based query optimizer which tries to optimize on the cost, in terms of the resources it will take to execute the query. Given a query,

then the query optimizer looks at the database schema, the database statistics and the system load at that time. It then decides which sequence to access the tables referred in the query, which sequence to execute the operations and what access method to be used to access the tables. Once a query plan is generated for a query, it is temporarily cached. For further invocations of the same query, the cached plan is used. Unused plans are discarded after some time

g. Service Broker

Used inside an instance, it is used to provide an asynchronous programming environment. For cross instance applications, Service Broker communicates The Service Broker, which runs as a part of the database engine, provides a reliable messaging and message queuing platform for SQL Server applications. over TCP/IP and allows the different components to be synchronized together, via exchange of messages.

h. Replication Service

SQL Server Replication Services are used by SQL Server to replicate and synchronize database objects, either in entirety or a subset of the objects present, across replication agents, which might be other database servers across the network, or database caches on the client side. Replication follows a publisher/subscriber model, i.e., the changes are sent out by one database server ("publisher") and are received by others ("subscribers").

PROJECT DESCRIPTION

PROBLEM DEFINITION

Providing security in a distributed system requires more than user authentication with passwords or digital certificates and confidentiality in data transmission. The Grid and Cloud Computing Intrusion Detection System integrates knowledge and behaviour analysis to detect intrusions.

OVERVIEW OF THE PROJECT

In this paper we are going to introduce concept using Cloud Computing and Grid Computing that efficiently look up and satisfies the user request. In order to make the concept of Cloud computing more efficient and reliable, we are going to include Grid computing for providing much more performance for the systems which access the data in server. This is done since more users login at the same time and the Server might not be able to provide the equal performance to all the systems, so when we add Grid Computing by combining other systems to the server we can achieve the performance by getting the performance from the systems that are connected to the server and providing them to the Systems that are accessing the Server.

MODULES

- Client
- Proxy server
- IDS service
  - ✓ Analyzer system
  - ✓ Alert system
- Storage service
  - ✓ Knowledge based service
  - ✓ Behavior based
- Event auditor

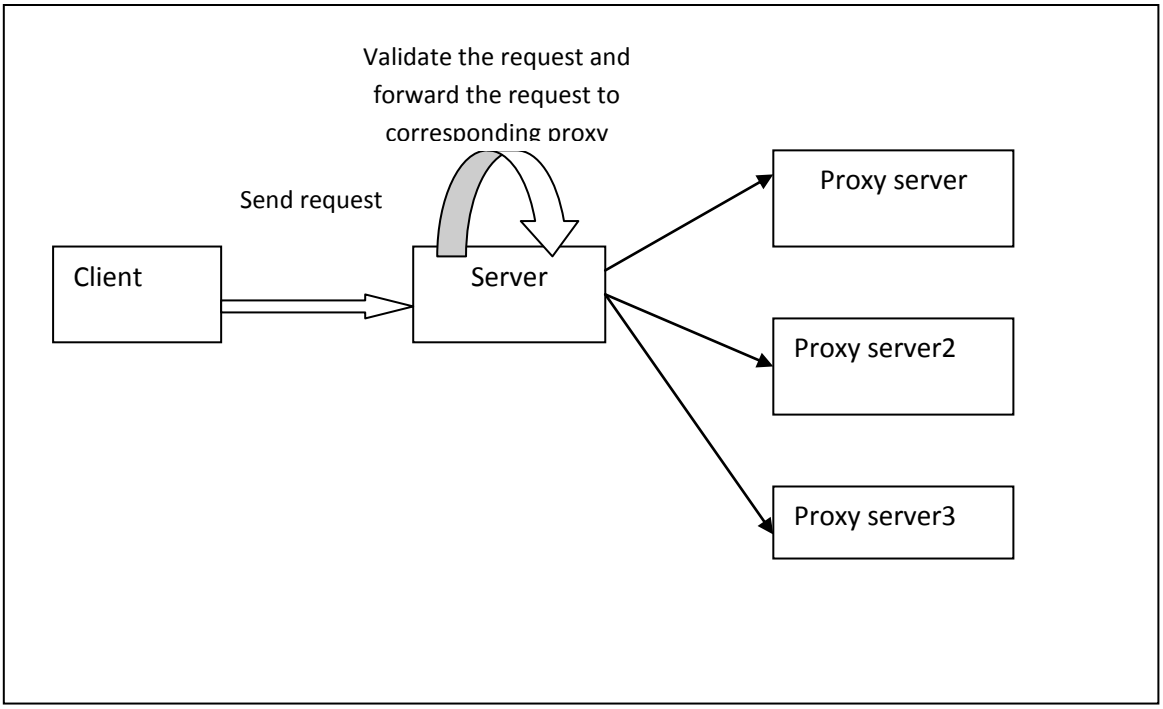
MODULE DESCRIPTION

A. CLIENT

The client system is the system which wants to get service or response from a server by forwarding request to the server.

B. PROXY SERVER

In the existing client server/distributed architecture all requests from all clients are accessed by a single server only. Due to high stress the system may hang or the data may be lost .to overcome this concept we moved for new cloud and Grid computing which provides a special system known as proxy server which extends the functionality of the server system. An anonymous proxy serves as a middleman between your web browser and an end server. Instead of contacting the end server directly to get a Web page, the browser contacts the proxy, which forwards the request on to the end server. When the end server replies to the proxy, the proxy sends the reply on to the browser. No direct communication occurs between the client and the destination server; therefore it appears as if the HTTP request originated from the intermediate proxy server.



C. IDS SERVICE

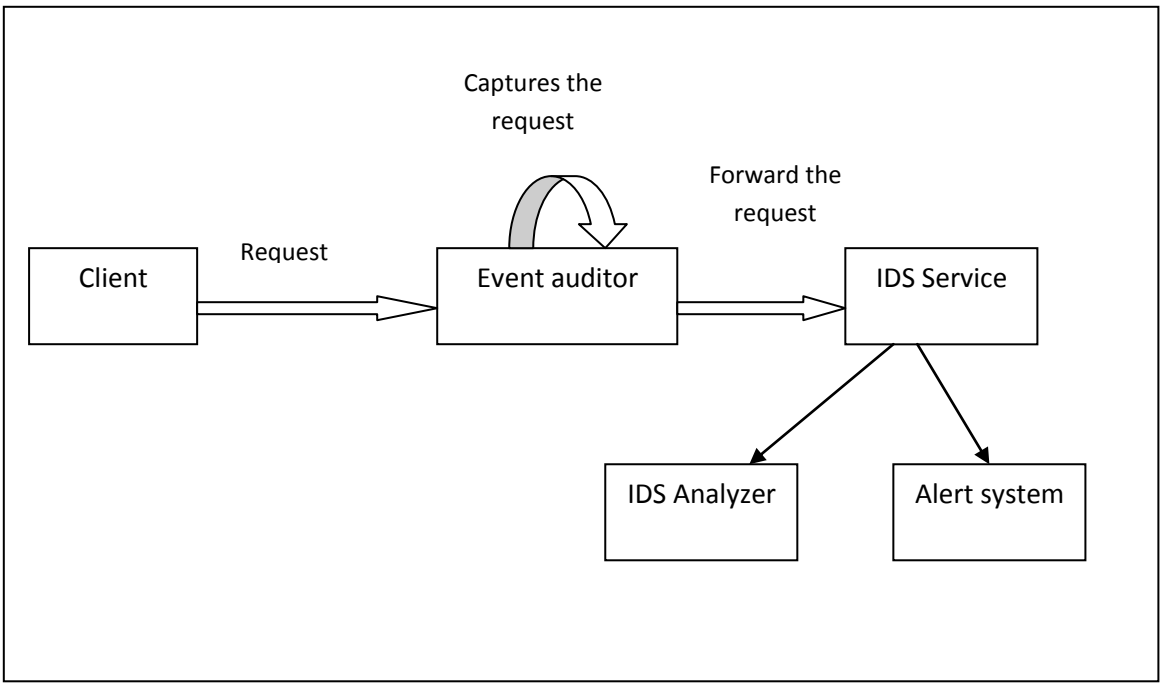
The Intrusion Detection Service (IDS) service increases a cloud's security level by providing two methods of intrusion detection. First method is behavior-based method which dictates how to compare recent user actions to the usual behavior. The second approach is knowledge-based method that detects known trails left by attacks or certain sequences of actions from a user who might represent an attack. The audited data is sent to the IDS service core, which analyzes the behavior using artificial intelligence to detect deviations. This has two subsystems namely analyzer system and alert system.

D. ANALYZER SYSTEM

The analyzer uses a profile history database to determine the distance between a typical user behavior and the suspect behavior and communicates this to the IDS service. The rules analyzer receives audit packages and determines whether a rule in the database is being broken. It returns the result to the IDS service core. With these responses, the IDS calculate the probability that the action represents an attack and alerts the other nodes if the probability is sufficiently high.

E. ALERT SYSTEM

This subsystem will work when intrusion is detected. If any node among the cloud system is affected by intrusion then this alert system will alert the remaining nodes about the intrusion.



F. STORAGE SERVICE

The storage service is a database system which contains two types of services namely knowledge based service and behaviour based service. Whenever a node gets requests or responses, the analyzer system compares the node information in the storage service.

G. KNOWLEDGE BASED SERVICE

We used audit data from both a log system and the communication system to evaluate the knowledge based system. We created a series of rules to illustrate security policies that the IDS should monitor.

The knowledge service is nothing but set of rules which is formed from previous attacks. Following things comes under this category

- Password cracking and access violation,
- Trojan horses,
- Interceptions most frequently associated with TCP/IP stealing and interceptions that often employ additional mechanisms to compromise operation of attacked systems (for example by flooding) man in the middle attacks.

- If any packets come with .exe extension
- Packets containing worms

H. BEHAVIOR BASED SERVICE

Here we have to classify the behaviour in two types as follows

USER BEHAVIOUR

For user behavior we have to analyze the users' behaviour. Using this method, we need to recognize expected behavior (legitimate use) or a severe behavior deviation.

Examples

Eg 1 -IP spoofing

There is a range of attacks that take advantage of the ability to forge (or 'spoof') your IP address. While a source address is sent along with every IP packet, it isn't actually used for routing. This means an intruder can pretend to be you when talking to a server. The intruder never sees the response packets (although your machine does, but throws them away because they don't match any requests you've sent). The intruder won't get data back this way, but can still send commands to the server pretending to be you

Eg 2- DNS poisoning through sequence prediction

DNS servers will "recursively" resolve DNS names. Thus, the DNS server that satisfies a client request will become itself a client to the next server in the recursive chain. The sequence numbers it uses are predictable. Thus, an intruder can send a request to the DNS server and a response to the server forged to be from the next server in the chain. It will then believe the forged response, and use that to satisfy other clients.

NODE BEHAVIOR

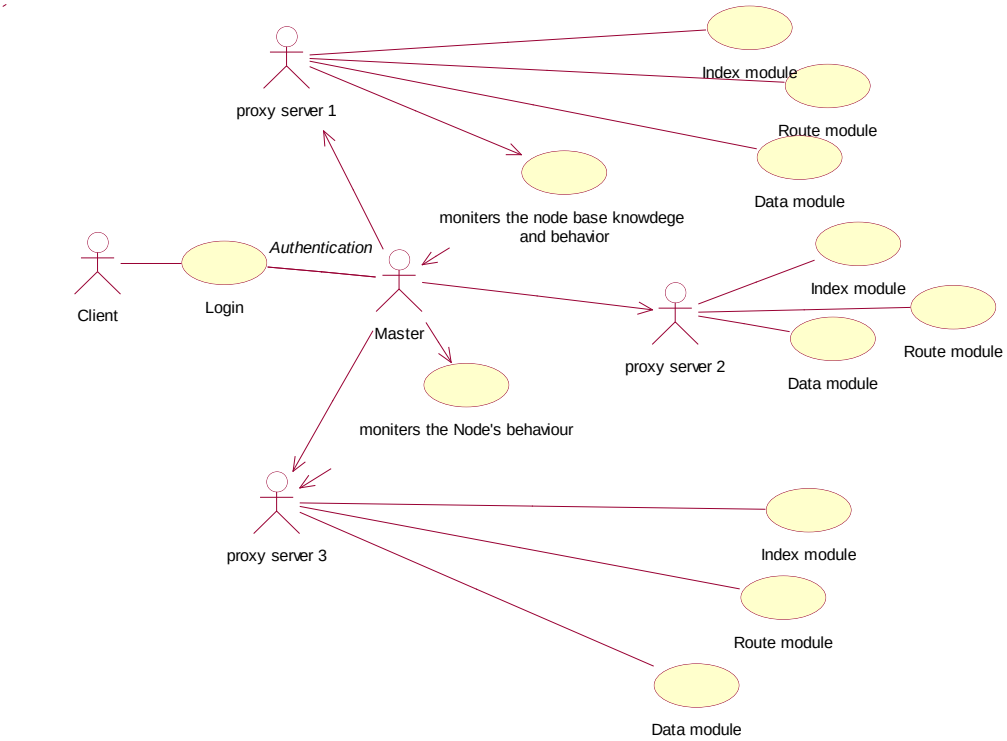
The most common way people approach network intrusion detection is to detect statistical anomalies. The idea behind this approach is to measure a "baseline" of such stats as CPU utilization, disk activity, user logins, file activity, and so forth. Then, the system can trigger when there is a deviation from this baseline.

The benefit of this approach is that it can detect the anomalies without having to understand the underlying cause behind the anomalies.

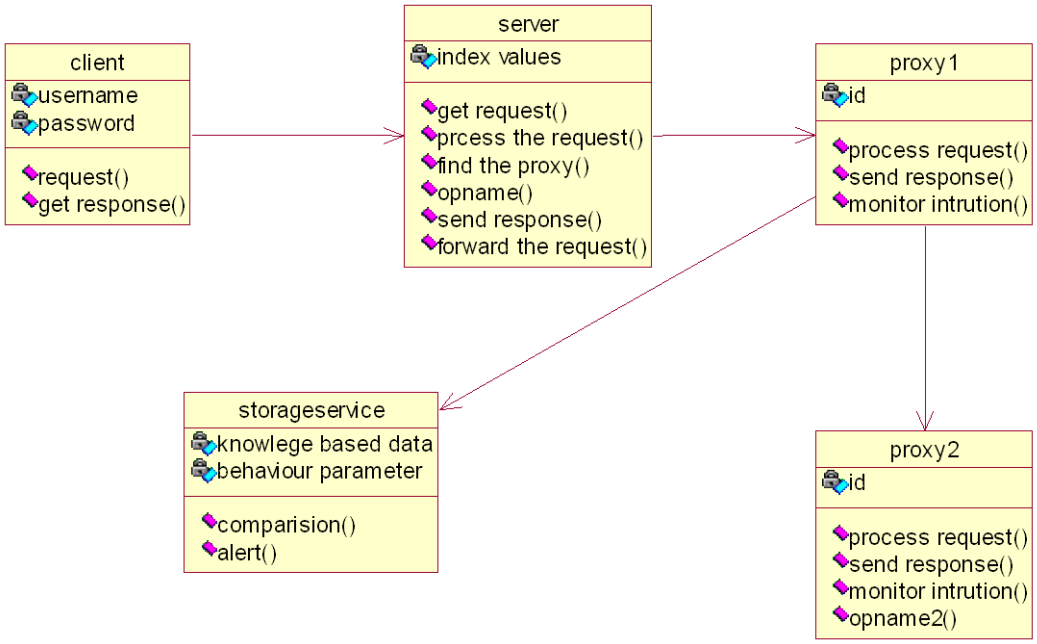
EVENT AUDITOR

To detect an intrusion, we need audit data describing the environment's state and the messages being exchanged. The event auditor can monitor the data that the analyzers are accessing. The first component monitors message exchange between nodes. Although audit information about the communication between nodes is being captured, no network data is taken into account—only node information. The second component monitors the middleware logging system. For each action occurring in a node, a log entry is created containing the action's type (such as error, alert, or warning), the event that generated it, and the message. With this kind of data, it's possible to identify an ongoing intrusion.

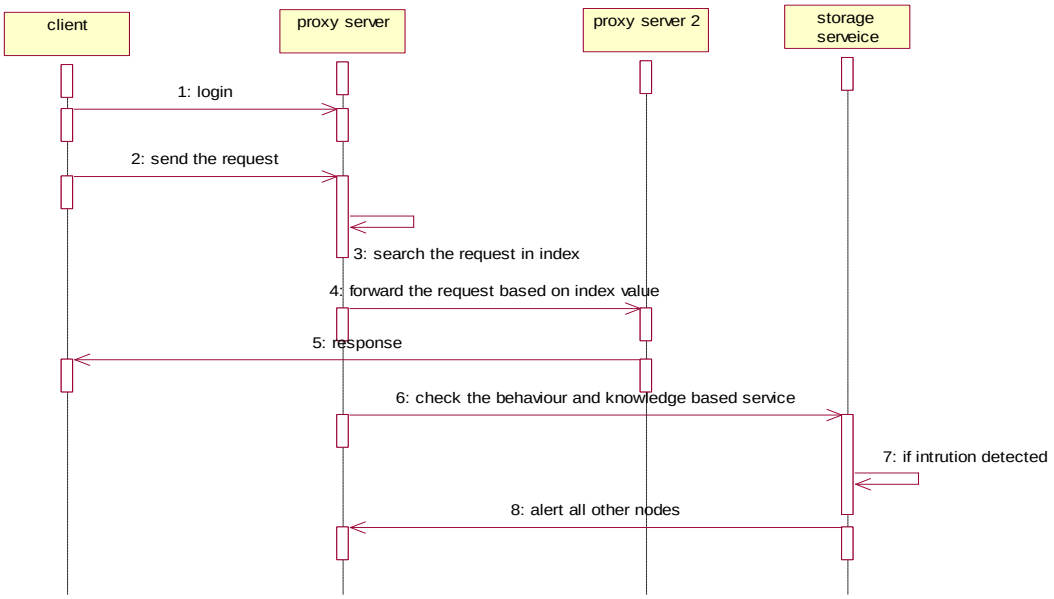
USE CASE DIAGRAM



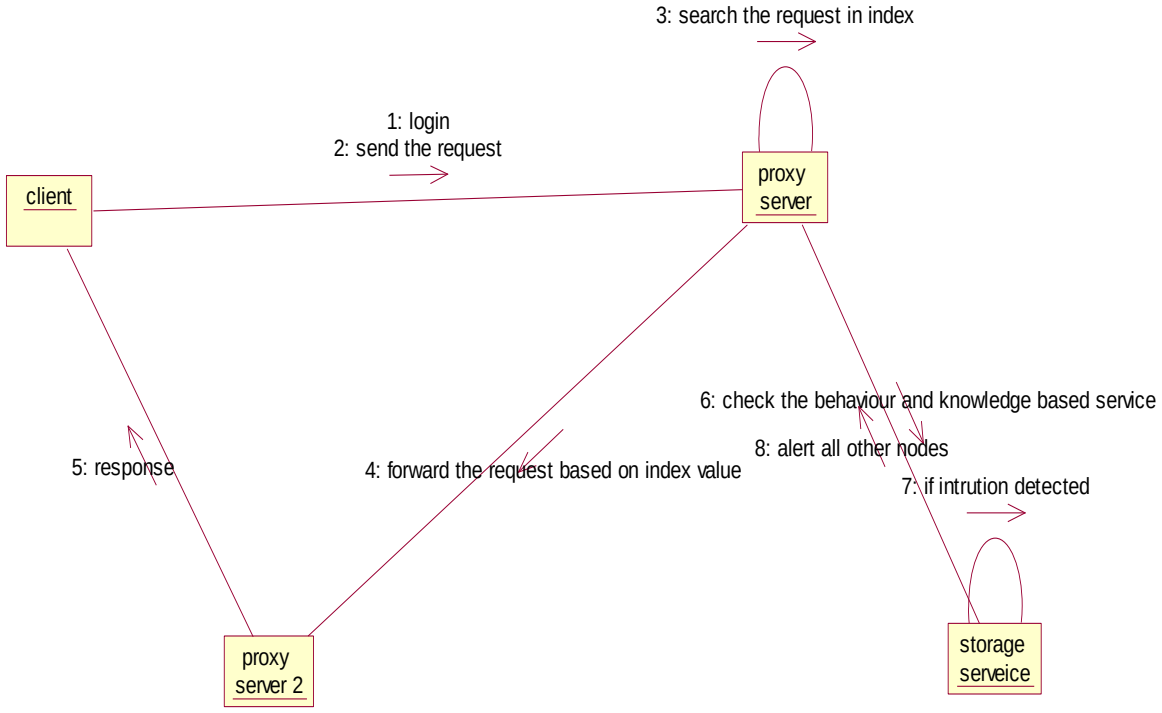
CLASS DIAGRAM



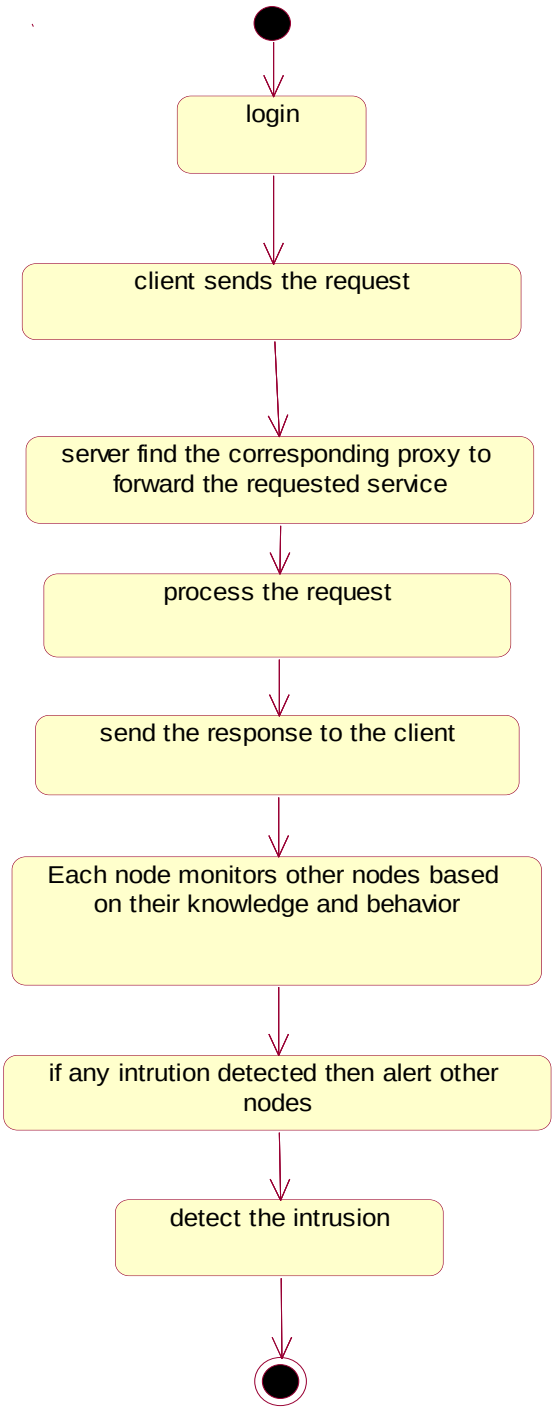
SEQUENCE DIAGRAM



COLLABRATION DIAGRAM



ACTIVITY DIAGRAM



SYSTEM TESTING

TESTING TYPES

UNIT TESTING:

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

INTEGRATION TESTING:

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

FUNCTIONAL TESTS:

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

PERFORMANCE TEST:

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

STRUCTURED TEST:

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

TESTING TECHNIQUES

Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is

the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

A. *Basis Path Testing*:

- Flow graph notation
- Cyclometric_complexity
- Deriving test cases
- Graph matrices Control

BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

TESTING STRATEGIES

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works "outward" toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

a. *Integration Testing*

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is "putting them together"- interfacing. There may be the chances of data lost across on another's sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

b. *Validation Testing*

Software validation is achieved through a series of tests that demonstrates conformity with requirements. A test plan outlines the classes of test to be conducted and a test procedure defines specific test cases that will be used to demonstrate conformity with requirements. Thus the proposed system under consideration has been tested by validation and found to be working satisfactorily.

c. Program Testing

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

d. *Security Testing*

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

e. Validation Testing:

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer.After validation test has been conducted, one of two conditions exists.

* The function or performance characteristics confirm to specifications and are accepted.
* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has

been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic.

f. *User Acceptance Testing*

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.
- Menu driven system.

SYSTEM IMPLEMENTATION

a. Implementation

Implementation is the most crucial stage in achieving a successful system and giving the user's confidence that the new system is workable and effective. Implementation of a modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user. And so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions clearly and quickly.

Initially as a first step the executable form of the application is to be created and loaded in the common server machine which is accessible to the entire user and the server is to be connected to a network. The final stage is to document the entire system which provides components and the operating procedures of the system.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

b. *Result Analysis*

In testing our prototype, we learned that it has a low processing cost while still providing a satisfactory performance for real-time implementation. Sending data to other nodes for processing didn't seem necessary. The individual analysis performed in each node reduces the complexity and the volume of data in comparison to previous solutions, where the audit data is concentrated in single points. In the future, we'll implement our IDS, helping to improve green (energy-efficient), white (using wireless networks), and cognitive (using cognitive networks) cloud computing environments. We also intend to research and improve cloud computing security.

CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION

This project is providing a satisfactory performance for real-time implementation. In this system we implement a best remedial technique to overcome the drawbacks in the existing cloud and grid system. The individual analysis performed in each node reduces the complexity and the volume of data in comparison to previous solutions, where the audit data is concentrated in single points.

FUTURE ENCHANCEMENT

In the future, we'll implement our IDS, helping to improve green (energy-efficient), white (using wireless networks), and cognitive (using cognitive networks) cloud computing environments. We also intend to research and improve cloud computing security.

APPENDIX

SAMPLE CODING

Main Server

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import java.util.regex.Pattern;
import javax.swing.*;

class MainServer implements Runnable
{

        static ServerSocket ss;
        static Socket soc,soc1,soc2,soc3;
        Connection con;
        Statement stmt;
        ResultSet rs=null;
        DataInputStream din=null,din1=null,din2=null,din3=null;
        DataOutputStream dout=null,dout1=null,dout2=null,dout3=null;
        String result="";
        public MainServer(){
                                try{
                                        ss = new ServerSocket(555);      }
                                catch(Exception ex){
                                        System.out.println(ex);
                                  }
                                }
        public void run(){
                try{
                        din=new DataInputStream(soc.getInputStream());
                        dout=new DataOutputStream(soc.getOutputStream());
                        String str=din.readUTF();
                         soc1 =new Socket("127.0.0.1",111); // Proxy 1
                        dout1 = new DataOutputStream(soc1.getOutputStream());
```

```
                soc2 =new Socket("127.0.0.1",222);// Proxy 2
                dout2 = new DataOutputStream(soc2.getOutputStream());
                soc3 =new Socket("127.0.0.1",333);// Proxy 3
                dout3=new DataOutputStream(soc3.getOutputStream());
            if(str.intern()=="Attacker")
            {
                        dout1.writeUTF("Attacker");
                        dout2.writeUTF("Attacker");
                        dout3.writeUTF("Attacker");
                        //din=new DataInputStream(soc.getInputStream());
                        //if((str=din.readUTF()).equals("OK"))
                        //{
                        //System.out.println("text");
                        dout.writeUTF("OK");
                        //}
                }
                else if(str.intern()== "UPLOAD"){
                        String filename = din.readUTF();
                        String filecontent = din.readUTF();
                        //boolean flag = EventAuditor(filename,filecontent);
                        //if(flag==true){


//JOptionPane.showMessageDialog(null,"Intruder");
                                dout1.writeUTF("UPLOAD");
                                dout2.writeUTF("UPLOAD");
                                dout3.writeUTF("UPLOAD");

        //   filename = din.readUTF();
                        //String filecontent = din.readUTF();
                        String filedata=filename+"@"+filecontent;
                        dout1.writeUTF(filedata);
                        dout2.writeUTF(filedata);
                        dout3.writeUTF(filedata);
                        }
                    else
                     {
                        StringTokenizer st = new StringTokenizer(str,"&");
                        String filename = st.nextToken();
                        String add = st.nextToken();

                        //Proxy1 Server
                        dout1.writeUTF(str);
```

```
            System.out.println("Data sent");
            din1 = new DataInputStream(soc1.getInputStream());

            String str1=din1.readUTF();
            System.out.println("File Availabele in System No:"+str1);
            if(str1.startsWith("Sorry No Files")){
                  System.out.println("Sorry No Files");
                  dout.writeUTF("Sorry No Files");
            }else{
                  dout.writeUTF(str1);
            }
            //Proxy2 Server

            dout2.writeUTF((str));
            System.out.println("Data sent");
            din2 = new DataInputStream(soc2.getInputStream());
            str1=din2.readUTF();
System.out.println("File Availabele in System No:"+str1);
            if(str1.startsWith("Sorry No Files")){
                  System.out.println("Sorry No Files");
                  dout.writeUTF("Sorry No Files");
            }else{
                  dout.writeUTF(str1);
            }

     // Proxy3 Server

dout3.writeUTF((str));
            System.out.println("Data sent");
            din3=new DataInputStream(soc3.getInputStream());
            str1=din3.readUTF();
            System.out.println("Read the data from Proxy3"+str1);
System.out.println("File Availabele in System No:"+str1);
            if(str1.startsWith("Sorry No Files")){
                  System.out.println("Sorry No Files");
                  dout.writeUTF("Sorry No Files");
            }else{
                  dout.writeUTF(str1);
            }
}

}catch(Exception ex){
            ex.printStackTrace();
```

```
                System.out.println(ex);
            }
        }


        private boolean EventAuditor(String filename,String filecontent){
        System.out.println("EventAuditor Start Here ... ");

            boolean flag = storageService(filename,filecontent);
            if(flag){
                return true;
                }

            return false;
        }
private boolean storageService(String filename , String fileContent){

            String fname = filename;
            String fcontent = fileContent;
            boolean flag = false;
            try{
                con = getConnection();
                String regex = "";
                if(con == null){
                    System.out.println("No Database Connection Found");
                }else{
                    stmt = con.createStatement();
                    rs = stmt.executeQuery("select * from knowledge");
                    while(rs.next()){
                        regex = rs.getString("knowledge");
                        System.out.println("DB Content"+regex.trim());
                        System.out.println("kkkkk"+fcontent.trim());

                        boolean isMatch =
Pattern.matches(regex.trim(),fcontent.trim());
            // System.out.println(isMatch);
                        if(isMatch){
                            System.out.println("Data is matched");

                            flag = true;
                            break;
                        }else{
                            System.out.println("Sorry");
```

```
                                        }

                        }
                        }


                }catch(Exception ex){
                        ex.printStackTrace();
                        System.out.println(ex);
                }

                return flag;
        }

        public static void main(String[] args){
                try{
                MainServer gw = new MainServer();
                while(true){
                soc=ss.accept();
                Thread t=new Thread(gw);
                        t.start();
                }
                }catch(Exception ex){
                        System.out.println(ex);
                }
        }

        private Connection getConnection(){
          //Server IP
                try{
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        Connection con =
DriverManager.getConnection("jdbc:odbc:Driver={SQL
Server};Server=.;Database=ids;UID=sa");
                        return con ;
                }catch(Exception ex){
                        ex.printStackTrace();
                        System.out.println(ex);
                }return null; } }
```

Client Login

```
import javax.swing.*;
import java.io.*;
import java.net.*;
```

```
import java.sql.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import com.birosoft.liquid.LiquidLookAndFeel;
class UserLogin extends JFrame
 {
        private JLabel jLabel1;
        private JLabel jLabel2;

        private JTextField jTextField1;
        private JPasswordField jTextField2;
        private JButton jButton1;
        private JButton jButton2;
        private JPanel contentPane;
        Toolkit tk;
        Dimension d;
        String n="",tno,uname,pin;
        Connection con,conn;
        Statement stmt;
        ResultSet rs;

        public UserLogin()
        {
                super();
                initializeComponent();
                //Server IP
                try{
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:DRIVER={SQL
Server};Server=127.0.0.1;Database=IDS;UID=sa");
                stmt = con.createStatement();
                }catch(SQLException sql){}
                catch(ClassNotFoundException cnf){}

        }
        private void initializeComponent(){
                try{
                  //
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndF
eel");

UIManager.setLookAndFeel("com.birosoft.liquid.LiquidLookAndFeel");
```

```
                }
                catch (Exception e) {System.out.println("UL-GE");}
                JLabel bgr;
                bgr = new JLabel();
                tk = Toolkit.getDefaultToolkit();
                d = tk.getScreenSize();

                jLabel1 = new JLabel();
                jLabel2 = new JLabel();

                jTextField1 = new JTextField();
                jTextField2 = new JPasswordField();
                jButton1 = new JButton();
                jButton2 = new JButton();
                bgr.setText("User Login");
                bgr.setFont(new Font("monotype corsiva",2,20));
                contentPane = (JPanel)this.getContentPane();
                jLabel1.setText("UserName : ");
                jLabel2.setText("  Password : ");

                jButton1.setText("SignIn");
                jButton1.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e)
                        {
                                try
                                {
                                        String s1=jTextField1.getText();
                                        String s2=jTextField2.getText();
                                        rs= stmt.executeQuery("select * from login
where uname='"+s1+"' and pass='"+s2+"'");
                                  if(rs.next()){
                                          dispose();
                                  new MainMenu(s1);
                                    //System.out.println("Karthik");
                                        }else{

        JOptionPane.showMessageDialog(null,"InValid Login");
                                        }
                                }
                                catch (Exception ex)
                                {ex.printStackTrace();System.out.println("ERROR :
"+ex);}
                        }
```

```
                });

                jButton2.setText("Cancel");
                jButton2.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e){

                                jTextField1.setText("");
                                jTextField2.setText("");
                        }
                });
                addWindowListener(new WindowAdapter(){
                                public void windowClosing(WindowEvent e){
                                                System.exit(0);
                                }
                        }
                );

                contentPane.setLayout(null);
                addComponent(contentPane, bgr,105,0,100,50);
                addComponent(contentPane, jLabel1, 45,50,80,18);
                addComponent(contentPane, jLabel2, 42,80,80,18);
                addComponent(contentPane, jTextField1, 140,50,100,21);
                addComponent(contentPane, jTextField2, 140,80,100,22);
                addComponent(contentPane, jButton1,60,120,83,28);
                addComponent(contentPane, jButton2, 160,120,83,28);

                this.setTitle("IDS : UserLogin ...");
                int x = (d.height/2) - 100;
                int y = (d.width/2) - 100;
                this.setLocation(new Point(y,x));
                this.setSize(new Dimension(300,200));
                this.setVisible(true);
                this.setResizable(false);
        }
        private void addComponent(Container container,Component c,int x,int y,int
width,int height){
                c.setBounds(x,y,width,height);
                container.add(c);
        }
        public static void main(String[] args){
                        new UserLogin();
        }
 }
```

Proxy

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.swing.*;
import java.util.regex.Pattern;

class Proxy1 implements Runnable
{
	DataInputStream din=null;
	DataOutputStream dout=null;
	static Socket soc=null;
	static ServerSocket ss=null;
	String FileList[]=null,dirname="C://Download//",ip=null;
	File f=null;
	FileInputStream fin=null;
	InetAddress add=null;
	boolean flag=false;
	String filename="",fname="",fcontent="";
	Connection con;
	ResultSet rs;
	Statement stmt;

	public Proxy1()
	{
		try
		{
			ss=new ServerSocket(111);
			add=InetAddress.getLocalHost();
			ip=add.getHostAddress();

		}
		catch(Exception ex)
		{
			ex.printStackTrace();
			System.out.println(ex);
		}
	}
	public void run()
	{
		try
```

```
                {

                        din=new DataInputStream(soc.getInputStream());
                        dout=new DataOutputStream(soc.getOutputStream());
                        filename=din.readUTF();

                         if(filename.intern()=="Attacker")
                        {
                                JOptionPane.showMessageDialog(null,"Attacker");
                                ///dout=new
DataOutputStream(soc.getOutputStream());
                                //System.out.println("test");
                                //dout.writeUTF("OK");
                        }
                        else if(filename.equals("UPLOAD")){
                                din=new DataInputStream(soc.getInputStream());
                                String files=din.readUTF();
                                StringTokenizer userreq=new
StringTokenizer(files,"@");
                        //String req=userreq.nextToken();
                         fname=userreq.nextToken();
                         fcontent=userreq.nextToken();
                                boolean flag = EventAuditor(fname,fcontent);
                                if(flag==true){
                                JOptionPane.showMessageDialog(null,"INTRUDER");
                                }else{
                                        FileOutputStream fout = new
FileOutputStream(".//Download//"+fname);
                                        fout.write(fcontent.getBytes());
                                        fout.close();
                                }
                        }else{
                        System.out.println("File Name"+filename);
                        StringTokenizer st = new StringTokenizer(filename,"&");
                        fname="";
                        filename = st.nextToken();
                        System.out.println(filename);
                        f=new File(dirname);
                        FileList=f.list();
                        for(int i=0;i<FileList.length;i++){

                                if(filename.equalsIgnoreCase(FileList[i])){
                                        System.out.println("Karthik");
```

```
                                        //dout.writeUTF(ip+"&"+filename);
                                        fname = filename;
                                        flag=true;
                                        System.out.println("KarthikDatasent");
                                        //break;

                                }else
                                        System.out.println("Sorry No Files");
                                  //  dout.writeUTF(ip+"&&"+"Sorry No Files");
                        }
if(flag==true){
                                dout.writeUTF(ip+"&"+fname);
                                System.out.println(fname);
                                flag=false;

                        }else{
                          dout.writeUTF("Sorry No Files");
                        }
                        }
                }catch(Exception ex){
                        ex.printStackTrace();
                        System.out.println(ex);
                }
        }
                        private boolean EventAuditor(String filename,String
filecontent){
                System.out.println("EventAuditor Start Here ... ");

                 boolean flag = storageService(filename,filecontent);
                 if(flag){
                         return true;
                 }


                return false;
        }

        private boolean storageService(String filename , String fileContent){

                String fname = filename;
                String fcontent = fileContent;
                boolean flag = false;
                try{
```

```
                        con = getConnection();
                        String regex = "";
                        if(con == null){
                                System.out.println("No Database Connection Found");
                        }else{
                                stmt = con.createStatement();
                                rs = stmt.executeQuery("select * from knowledge");
                                while(rs.next()){
                                        regex = rs.getString("knowledge");

        System.out.println(".....");
                                        System.out.println("DB Content"+regex.trim());
                                        System.out.println("kkkkk"+fcontent.trim());

                                         boolean isMatch =
Pattern.matches(regex.trim(),fcontent.trim());
                // System.out.println(isMatch);
                                         if(isMatch){
                                                System.out.println("Data is matched");

                                                flag = true;
                                                break;
                                        }else{
                                                System.out.println("Sorry");
                                        }

                        }
                        }


                }catch(Exception ex){
                        ex.printStackTrace();
                        System.out.println(ex);
                }
                return flag;
        }
        private Connection getConnection(){
                try{
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        Connection con =
DriverManager.getConnection("jdbc:odbc:Driver={SQL
Server};Server=.;Database=ids;UID=sa");
                        return con ;
```

```
        }catch(Exception ex){
                ex.printStackTrace();
                System.out.println(ex);
        }
        return null;
}
public static void main(String[] args){
        try{
        Proxy1 p=new Proxy1();

        while(true){
        soc=ss.accept();
        Thread t=new Thread(p);
        t.start();
        }
        }catch(Exception ex){


        System.out.println(ex); } }
```

REFERENCE

[1] H. Debar, M. Dacier, and A. Wespi, "Towards a Taxonomy of Intrusion Detection Systems," *Int'l J. Computer and Telecommunications Networking*, vol. 31, no. 9,1999, pp. 805–822.

[2] I. Foster et al., "A Security Architecture for Computational Grids," *Proc. 5th ACM Conf. Computer and Communications Security*, ACM Press, 1998, pp. 83–92.

[3] S. Axelsson, *Research in Intrusion-Detection Systems: A Survey*, tech. report TR-98-17, Dept. Computer Eng., Chalmers Univ. of Technology, 1999. 4. A. Schulter et al., "Intrusion Detection for Computational Grids," *Proc. 2nd Int'l Conf. New Technologies, Mobility, and Security*, IEEE Press, 2008, pp. 1–5.

[5] H. Franke et al., "Grid-M: Middleware to Integrate Mobile Devices, Sensors and Grid Computing," *Proc. 3rd Int'l Conf. Wireless and Mobile Comm.* (ICWMC 07),IEEE CS Press, 2007, p. 19. 6. N.B. Idris and B. Shanmugam, "Artificial IntelligenceTechniques Applied to Intrusion Detection," *Proc. 2005 IEEE India Conf. (Indicon) 2005 Conf.*, IEEE Press, 2005, pp. 52–55.

[7] P.F. da Silva and C.B. Westphall, "Improvements in the Model for Interoperability of Intrusion Detection Responses Compatible with the IDWG Model," *Int'l J. Network Management*, vol. 17, no. 4, 2007, pp. 287–294.

[8] P. Luo, F. Lin, Y. Xiong, Y. Zhao, and Z. Shi, *Towards combining web classification and web information extraction: a case study*, In KDD'09:Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009.

[9] J. Ma, L. Saul, S. Savage, and G. Voelker, *Beyond blacklists: Learning to detect malicious web sites from suspicious urls*, In The 15th ACM SIGKDD Conference On Knowledge Discovery and Date Mining, 2009.

[10] 1Paulo Fernando da Silva, 2Carlos Becker Westphall*Network and Management Laboratory, Post-Graduate Program in Computer Science, Federal Universityof Santa Catarina, Florianópolis, Brazil*

[11] Herve Debar, Marc Dacier and Andreas Wespi IBM Research Division Zurich Research Laboratory, Saumerstrasse 4 CH-8803 Ruschlikon Switzerland zurich@ibm.com

[12] Hans A. Franke, Fernando L. Koch, Carlos O. Rolim, Carlos B. Westphall, Douglas O. BalenNetworks and Management Laboratory Federal University of Santa Catarina, Florianópolis, Brazil{koiote,koch,ober,westphal,douglasb}@inf.ufsc.br

[13] Matti Manninen, Helsinki University of Technology, mimannin@niksula.hut.fi

[14] Netcraft Ltd, http://toolbar.netcraft.com/.

[15] Phishtank, www.phishtank.com.

[16] J. M. Pierre, *Practical issues for*