Cloud Computing, is the fastest developing field of IT industries. Cloud Computing is an Internet based computing where implicitly shared servers provides all needed resources such as software, hardware, platform, Infrastructure, devices to the Cloud user.

1. Detecting DDoS Attacks in Cloud Computing Environment
2. Controlling high bandwidth flows at the congested router

* This method to keep high band width flows with the help of a state .This mechanism is named RED-PD, a mechanism that combinessimplicity andprotectionbykeeping state forjust the high-bandwidthflows.
* Preferential dropping is done using a pre-filter in front of the output queue. Packets from the monitored flows are dropped in the pre-filter with a probability dependent on the excess sending rate of the flow. Unmonitored traffic is put in the output queue directly.
* The light-weight mechanism shown above: a) not only protects unmonitored traffic from the monitored flows, but also provides relative fairness among the monitored flows; b) does not starve the monitored flows like "leftover band- width" approaches; c) does not protect the monitored flows

1. Spoofer Project

The Spoofer project originated in 2005 as the result of discussions over the general ability to successfully send spoofed-source IP packets across the Internet. At the time, a common misconception was that “most networks perform source address filtering, and, even if they don’t, botnets remove any of the anonymity advantage afforded by spoofing.

Our approach was to develop a measurement software client that volunteers across the Internet could download and run from their networks, testing their own ability to send various types of spoofed packets from their network to our server, which collects and aggregates test results.

1. A Collaborative Peer-to-Peer Architecture to Defend Against DDoS Attacks

This paper present method to witnessing an important increase in attacks amongwhich distributed denial-of-service (DDoS) that easily flood the victims using multiple paths

Current DDoS solutions are many, ranging from host-based solutions to network and infrastructure solutions. Our architecture is basically proposed for DoS detection and IP trace back solutions.

* + **Anomaly based detection schemes** that detect anomalies caused by DDoS attacks. In this case a model must be established according to standard protocol normal system activities. In general the intrusion detection entities are deployed on hosts or routers and the agent is deployed at a single point or network-based where the agents cooperate either in a centralized or a decentralized manner.
  + **Backtracking techniques** that work in a hop by hop manner to construct a summary of routed flow. In this class we have the proactive measures category where the flow is generated independently from the presence of the attacks and the reactive measures where the summary is generated on demand.
  + **Flow extension techniques** bring additional information to flows during their travel. We have the in-band messaging (packet marking), that can be probabilistic or deterministic, and the out-of-band messaging that ends the trace back data in separated packets.

1. Intrusion Detection in the Cloud

Intrusion Detection Systems (IDS) have been used widely to detect malicious behaviors in network communication and hosts. IDS management is an important capability for distributed IDS solutions, which makes it possible to integrate and handle different types of sensors or collect and synthesize alerts generated from multiple hosts located in the distributed environment.

1. **SYSTEM ANALYSIS**
   1. **Existing System**

This proposes an innovative implementation of DDoS defense technique named Network Egress and Ingress Filtering (NEIF), which can be implemented at ISPs’ edge routers, to avoid a DDoS attacks in Cloud.

* 1. **Proposed system**

In recent years there are significant number of attacks have been launched against a Cloud environment, so it is necessary to take steps against defending DDoS attack in Cloud environment. The purpose Denial of Service attack is to make the network resources such as internet, web services and applications unavailable to the genuine users for a certain period of time.

1. **DESIGN AND DEVELPOMENT**
   1. **MODULE DESCRIPTION**

Project contains five modules

1. Client
2. Attacker
3. Egress filtering
4. Ingress filtering
5. Detection.
6. **Client**

Client is the normal or authorized person who requests/browse data from server. Client firstly register and then login to browse/upload a particular data

1. **Attacker**

Attacker is the unauthorized person who tries to deny services provided to the normal user by hanging or overloading the server by continually sending requests to server

1. **Egress filtering**

egress filtering is the practice of monitoring and potentially restricting the flow of information outbound from one network to another. Typically it is information from a private computer network to the Internet that is controlled.Egress filtering helps ensure that unauthorized or malicious traffic never leaves the internal network

1. **Ingress filtering**

Ingress filtering is a technique used to ensure that incoming packets are actually from the networks from which they claim to originate. Networks receive packets from other networks. Normally a packet will contain the IP address of the computer thatoriginally sent it. This allows devices in the receiving network

1. **Detection**

This module detects the authorization of datapacketes from users.It is mainly done in two stages

1. After data send from the system
2. Before the data is send to server

When client sends a request it is directed to ingress filtering stage where it checks whether the IP address of that user is from the allocated network, if so the data is redirected to server. Before reaching the server the data packets are again checked using egress filtering. There two rules are applied

1. Data packets from the IP address are allocated with particular limit. If packets from user are exceeding that particular limit then it wont directed to server
2. Checks whether the IP address of each packets is registered within the server, if not wont redirected to server.

**Development**

We have designed the client module for register and login

* 1. **UML DIAGRAM**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* *User Model View*

This view represents the system from the users perspective. The analysis representation describes a usage scenario from the end-users perspective.

* *Structural model view*

In this model the data and functionality are arrived from inside the system. This model view models the static structures.

* *Behavioral Model View*

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* *Implementation Model View*

In this the structural and behavioral as parts of the system are represented as they are to be built.

* *Environmental Model View*

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented. UML is specifically constructed through two different domains they are:

UML Analysis modeling, this focuses on the user model and structural model views of the system.

UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

* + 1. **Use Case Diagram**

Use case diagram consists of use cases and actors and shows the interaction between the use cases and actors. The purpose is to show the interactions between use cases and actors to represent the system requirement from user’s perspective. It must be remembered that the use cases are the function that are to be performed in the module. An actor could be the end user of the system or the external system.

* + 1. **Sequence Diagram**

The purpose is to show the flow of the functionality through a use case. In other words we can call it mapping processes in terms of data transfers from the actor through corresponding objects. To represent the logical flow of data with respective a process it must be remembered that the sequence diagram display objects and not the classes.

* + 1. **Data Flow Diagram**

Data Flow Diagram is used to define the flow of system and its resources such as information. Data Flow Diagrams represent one of the most ingenious tools used for structured analysis. A Data Flow Diagram or DFD as it is shortly called is also known as bubble chart

Six rules for considering the Data Flow Diagram:-

1. Arrows should not cross each other.
2. Rectangle, circle and files must bear names.
3. Decomposed data flow square and circles can have the same names.
4. Choose meaningful name for data flow.
5. Draw all data flows around the outside of the diagram.
6. Control information such as record count, password and validation requirements is
7. not pertinent to Data Flow Diagram.

**Advantages of DFD:-**

1. Graphic technique is superb and simple.
2. System boundaries are well described.
3. Each part of data can be represented by different level of details.
4. **DATABASE DESIGN**

Data Base design is the logical form of design of data storage in the form of records in a particular structure in the form of tables with fields which is not transparent to the normal user but it actually act as the backbone of the system. As we know database is a collection of which helps the system to manage and store data is called database management system. Data base management system builds some form of constraints like integrity constraints, i.e., the primary key or unique key and referential integrity which help to keep data structure storage and access of data from tables efficiently and accurately and take necessary steps to concurrent access of data and avoid redundancy of data in tables by normalization criterions.

Normalization is the method of breaking down complex table structures into simple table structures by using certain rules thus reduce redundancy and inconsistency and disk space usage and thus increase the performance of the system or application which is directly linked to the database design and also solve the problems of anomalies.

**There are different forms of normalization, some are:**

* First normal form (1NF)
* Second normal form (2NF)
* Third normal form (3NF)

**Boyce codde normal form :**

* Forth normal form (4NF)
* Fifth normal form (5NF)

The data base design of the new system is in second normal form and every non key attribute is functionally depend only on the primary key. The master and transaction tables and their structure are shown below.

**3.1 User Details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Datatype** | **Constraint** | **Description** |
| **id** | **Int** | **Primary key** | **User ID Auto increment** |
| **Name** | **Varchar** | **Not null** | **Name of use** |
| **User Name** | **Varchar** | **Not null** | **Username** |
| **Varchar** | **Varchar** | **Not null** | **Password** |

1. **TECHNOLOGY FEATURES**
   1. **Microsoft.NET**

To build C# applications that meet your performance objectives, you need to understand the places where bottlenecks typically occur, the causes of the bottlenecks, and the steps to take to prevent the bottlenecks from occurring in your application. A combination of sound architecture and design, best practice coding techniques, and optimized platform and Microsoft® .NET Framework configuration is required.

The .NET initiative offers a complete suite for developing and deploying applications. This suite consists of .NET products, .NET services and .NET frame work.

Advantages of the .NET frame work

1. Consistent programming model
2. Multi-platform applications
3. Multi language integration
4. Automatic resource management
5. Ease of deployment