PROJECT

DISTRIBUTED DENIAL OF SERVICE USING MYSQL RELATIONAL DATABASE STRUCTURE BASED ON NETWORK SECURITY

Prepared By:
Pratik Bagade

Guided By:
Zakir Hussain

TABLE OF CONTENTS

1) Distributed Denial of Service (DDoS):

Distributed Denial-of-Service (DDoS) attack is a type of cyberattack where an attacker attempts to make a computer or network resource unavailable by overwhelming it with traffic from multiple sources. This is typically done by using a network of compromised devices (bots) to flood the targeted system with traffic, causing it to become overwhelmed and unable to handle legitimate requests.

Here's how a DDoS attack works:

1) **Botnet Creation:** Attackers makes use of vulnerabilities in devices (like computers, IoT devices, etc.) to install malware, forming a botnet—a network of compromised machines controlled remotely.

2) **Traffic Generation:** Using this botnet, the attacker commands each bot to start sending requests, usually in a coordinated manner, to a target system (such as a server or network).

3) **Traffic Flood:** The compromised devices generate a massive amount of fake traffic or requests, flooding the target with far more data than it can handle. This traffic may consume bandwidth, processing power, or both.

4) **System Overload:** The target system, unable to distinguish legitimate traffic from the flood of malicious requests, becomes overwhelmed, leading to slowdowns or complete service interruptions.

Types of DDoS attacks:

1) **Volume-based attacks:**

Goal: Flood the target with a massive amount of traffic to saturate its bandwidth.

Effect: The target's internet connection gets overwhelmed, making it impossible for legitimate traffic to get through.

2) **Protocol attacks:**

Goal: Exploit weaknesses in network protocols to deplete system resources (e.g., CPU, memory).

Effect: The target system's resources get exhausted as it tries to handle malformed or excessive protocol-level requests.

3) **Application-layer attacks:**

Goal: Target specific applications or services running on the server, often mimicking legitimate user behavior.

Effect: Consumes the resources of the targeted application, making it unresponsive or slow for legitimate users.

DDoS attacks can be launched using various techniques, including:

1) **Botnets:**

Attackers build or rent botnets made up of compromised devices (like computers, IoT devices, routers) that are controlled remotely. These bots are commanded to send large volumes of traffic to a target, overwhelming it.

2) **Malware:**

Malware is used to infect devices, turning them into bots that can be controlled by the attacker. Common malware types include Trojans and worms, which are often used to gain unauthorized control over devices.

3) **Scripting:**

Attackers can use scripting languages (like Python, Perl, or Bash) to automate attack processes. These scripts can send a high number of requests to the target in an automated fashion, making the attack more efficient and scalable.

4) **Amplification Attacks:**

In an amplification attack, the attacker sends small requests to open services like DNS or NTP, which then reply with large responses to the target, amplifying the amount of traffic the victim receives. Examples include DNS amplification and NTP reflection attacks.

To protect against DDoS attacks, organizations can use:

1) **Firewalls:**

Firewalls act as a barrier between the internal network and the internet. They filter traffic by enforcing security rules, allowing only legitimate requests through while blocking suspicious or malicious traffic.

2) **Intrusion Detection/Prevention Systems (IDS/IPS):**

IDS monitors traffic for signs of an attack and alerts administrators when suspicious activity is detected.

IPS takes it a step further by actively blocking or mitigating malicious traffic in real-time, helping to stop attacks before they cause harm.

3) **Load Balancing:**

Load balancers distribute incoming traffic across multiple servers, helping to prevent any single server from becoming overwhelmed. This approach can also reroute traffic in the event of an attack, ensuring availability.

4) **Content Delivery Networks (CDNs):**

CDNs store cached copies of website content in multiple geographical locations. By distributing requests across their network, they reduce the load on the main server, absorb attack traffic, and ensure continuous service availability.

2.1 Software Requirements

Following are the software requirements necessary of the Project.

- **DBMS:** SQL
- **Php :-** Php is used is backend for database connection and fetching content in webpage **.**
- **HTML/CSS**

2.2Hardware Requirements

Following are the hardware requirements that are most important for this project.

- **Processor** : Pentium IV–2.0 GHz
- **RAM** **:** Min 4 GB
- **Hard Disk/SD Card** : Min 250 GB

3) Databases used in this Project:

- Create five database using the below syntax:

create database [name of database];

```
mysql> create database Attack_Detection;
Query OK, 1 row affected (0.01 sec)

mysql> create database Network_Traffic;
Query OK, 1 row affected (0.01 sec)

mysql> create database System_Logging;
Query OK, 1 row affected (0.01 sec)

mysql> create database Botnet_Information;
Query OK, 1 row affected (0.01 sec)

mysql> create database Mitigation_Strategies;
Query OK, 1 row affected (0.01 sec)
```

- To display the names of created databases:

show databases;

```
mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| attack_detection     |
| botnet_information   |
| demo                 |
| food_delivery        |
| information_schema   |
| mitigation_strategies |
| mysql                |
| network_traffic      |
| performance_schema   |
| sys                  |
| system_logging       |
| techmahindra         |
+----------------------+
12 rows in set (0.00 sec)
```

4)Tables used in each of the Databases:

1. Attack_Detection Database

- **attacks**: Stores records of detected attacks.
  - **Columns:** `id`, `attack_type`, `attack_date`, `source_ip`
- **attack_types**: Manages different types of attacks.
  - **Columns:** `id`, `type_name`, `description`
- **sources**: Tracks information about the sources of detected attacks.
  - **Columns:** `id`, `source_ip`, `source_country`
- **detection_rules**: Defines rules used to detect various types of attacks.
  - **Columns:** `id`, `rule_name`, `rule_description`
- **alerts**: Logs alerts generated when an attack is detected.
  - **Columns:** `id`, `attack_id`, `alert_date`, `alert_level`

2. Network_Traffic Database

- **traffic**: Records network traffic data.
  - **Columns:** `id`, `timestamp`, `source_ip`, `destination_ip`, `protocol`
- **protocols**: Stores information about the protocols used in network communication.
  - **Columns:** `id`, `protocol_name`, `description`
- **ip_addresses**: Tracks details of IP addresses involved in the network traffic.
  - **Columns:** `id`, `ip_address`, `ip_type`, `country`, `description`
- **network_devices**: Manages information about network devices.
  - **Columns:** `id`, `device_name`, `device_type`
- **traffic_stats**: Records statistics about the network traffic.
  - **Columns:** `id`, `timestamp`, `traffic_volume`

---

3. System_Logging Database

- **logs**: Stores log entries for system activities.
    - **Columns:** `id`, `log_timestamp`, `log_level`, `message`, `source_ip`
- **error_codes**: Stores information about error codes.
    - **Columns:** `id`, `code`, `description`
- **event_types**: Tracks different types of events occurring in the system.
    - **Columns:** `id`, `event_name`, `event_description`
- **audit_trails**: Logs actions taken by users in the system.
    - **Columns:** `id`, `event_id`, `event_timestamp`, `user_id`, `action`
- **user_sessions**: Records user session information.
    - **Columns:** `id`, `user_id`, `session_start`, `session_end`, `status`

---

4. Botnet_Information Database

- **botnets**: Stores information about identified botnets.
    - **Columns:** `id`, `name`, `description`, `creation_date`
- **botnet_devices**: Tracks devices compromised by botnets.
    - **Columns:** `id`, `botnet_id`, `device_ip`, `infection_date`
- **command_and_control_servers**: Stores information about command-and-control servers used by botnets.
    - **Columns:** `id`, `botnet_id`, `server_ip`, `location`
- **malware_samples**: Keeps records of malware samples used by botnets.
    - **Columns:** `id`, `botnet_id`, `sample_hash`, `date_collected`
- **attack_patterns**: Manages information about known attack patterns used by botnets.
    - **Columns:** `id`, `botnet_id`, `pattern_name`, `description`

5. Mitigation_Strategies Database

- **mitigation_methods**: Stores different methods used to mitigate DDoS attacks.
    - **Columns:** `id`, `method_name`, `description`
- **applied_strategies**: Logs strategies applied to mitigate specific attacks.
    - **Columns:** `id`, `mitigation_id`, `attack_id`, `implementation_date`, `effectiveness`
- **response_teams**: Stores information about teams responsible for responding to DDoS attacks.
    - **Columns:** `id`, `team_name`, `contact_info`
- **incident_reports**: Records detailed reports of incidents, including the attack and response.
    - **Columns:** `id`, `attack_id`, `report_date`, `summary`
- **training_sessions**: Logs training sessions for response teams.
    - **Columns:** `id`, `team_id`, `session_date`, `topic`

5) Queries identified by the Network Infra security team:

Retrieve all attacks with corresponding attack type and source information:

```
mysql> use Attack_Detection;
Database changed
mysql> select a.*, at.Type_Name, s.Source_Country from Attacks a join Attack_Types at on a.Attack_Type=at.Id join Sou
rces s on a.Source_IP = s.Source_IP;
+------+------------+---------------------+--------------+---------------------+----------------+
| Id   | Attack_Type | Attack_Date         | Source_IP    | Type_Name           | Source_Country |
+------+------------+---------------------+--------------+---------------------+----------------+
|    1 |          1 | 2022-01-01 12:00:00 | 192.168.1.100 | DDoS                | USA            |
|    2 |          2 | 2022-01-02 13:00:00 | 192.168.1.101 | SQL Injection       | China          |
|    3 |          3 | 2022-01-03 14:00:00 | 192.168.1.102 | Cross-Site Scripting | Russia         |
|    4 |          1 | 2022-01-04 15:00:00 | 192.168.1.103 | DDoS                | India          |
|    5 |          2 | 2022-01-05 16:00:00 | 192.168.1.104 | SQL Injection       | Brazil         |
+------+------------+---------------------+--------------+---------------------+----------------+
5 rows in set (0.02 sec)
```

Retrieve all detection rules with corresponding attack type:

```
mysql> select dr.*, at.Type_Name from Detection_Rules dr join Attack_Types at on dr.Rule_Description like concat('%',
at.Type_Name,'%');
+------+-----------+---------------------+---------------+
| Id   | Rule_Name | Rule_Description    | Type_Name     |
+------+-----------+---------------------+---------------+
|    1 | Rule 1    | Detect DDoS attacks | DDoS          |
|    2 | Rule 2    | Detect SQL Injection | SQL Injection |
|    4 | Rule 4    | Detect Brute Force  | Brute Force   |
|    5 | Rule 5    | Detect Phishing     | Phishing      |
+------+-----------+---------------------+---------------+
4 rows in set (0.01 sec)
```

Retrieve all alerts with corresponding attack information and alert level:

```
mysql> select al.*, a.Attack_Type, a.Attack_Date, at.Type_Name from Alerts al join Attacks a on al.Attack_Id=a.Id joi
n Attack_Types at on a.Attack_Type = at.Id;
+------+-----------+---------------------+-------------+-------------+---------------------+---------------------+
| Id   | Attack_Id | Alert_Date          | Alert_Level | Attack_Type | Attack_Date         | Type_Name           |
+------+-----------+---------------------+-------------+-------------+---------------------+---------------------+
|    4 |         4 | 2022-01-04 15:00:00 | High        |           1 | 2022-01-04 15:00:00 | DDoS                |
|    1 |         1 | 2022-01-01 12:00:00 | High        |           1 | 2022-01-01 12:00:00 | DDoS                |
|    5 |         5 | 2022-01-05 16:00:00 | Medium      |           2 | 2022-01-05 16:00:00 | SQL Injection       |
|    2 |         2 | 2022-01-02 13:00:00 | Medium      |           2 | 2022-01-02 13:00:00 | SQL Injection       |
|    3 |         3 | 2022-01-03 14:00:00 | Low         |           3 | 2022-01-03 14:00:00 | Cross-Site Scripting |
+------+-----------+---------------------+-------------+-------------+---------------------+---------------------+
5 rows in set (0.00 sec)
```

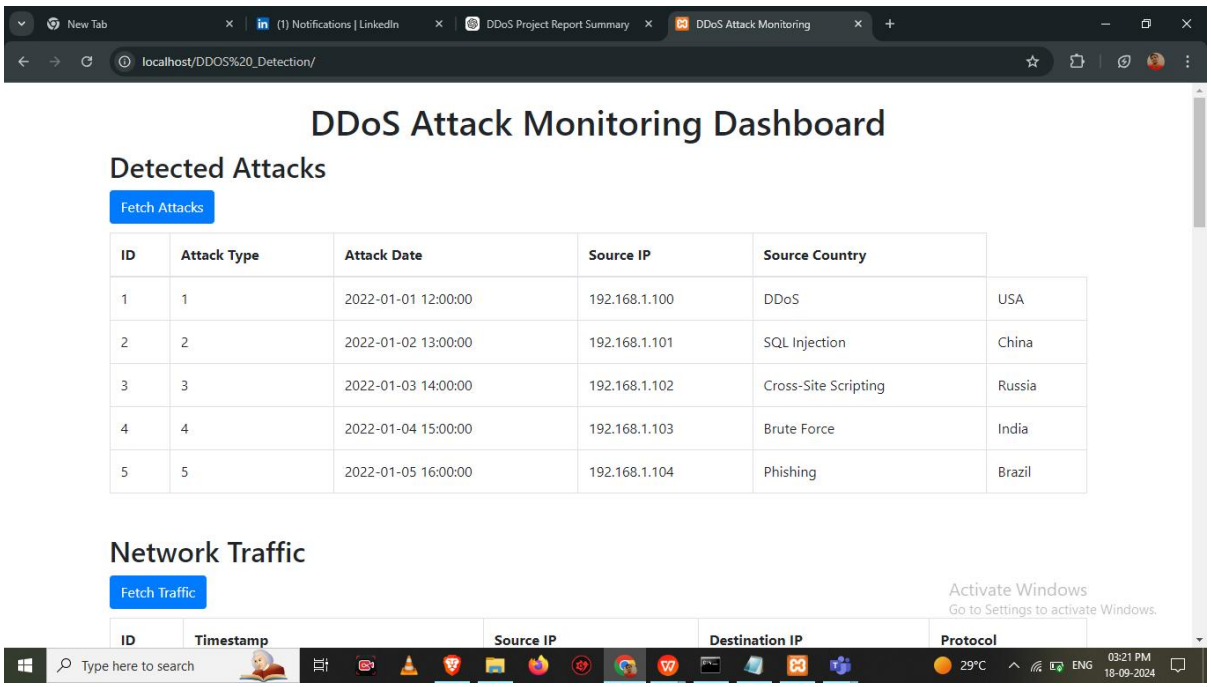
Retrieve all sources with corresponding attack and alert information:

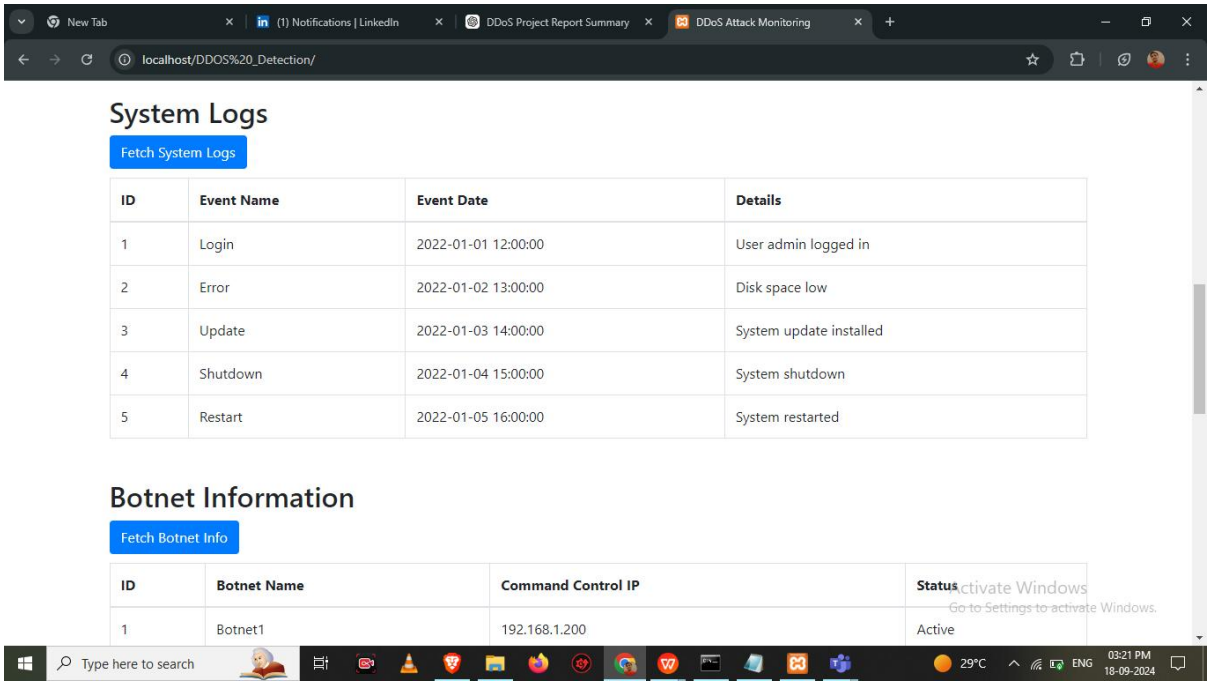
```
mysql> select s.*, a.Attack_Date, al.Alert_Date, al.Alert_Level from Sources s join Attacks a on s.Source_IP=a.Source
_IP join Alerts al on a.Id = al.Attack_Id;
+------+--------------+----------------+---------------------+---------------------+-------------+
| Id   | Source_IP    | Source_Country | Attack_Date         | Alert_Date          | Alert_Level |
+------+--------------+----------------+---------------------+---------------------+-------------+
|    1 | 192.168.1.100 | USA           | 2022-01-01 12:00:00 | 2022-01-01 12:00:00 | High        |
|    2 | 192.168.1.101 | China         | 2022-01-02 13:00:00 | 2022-01-02 13:00:00 | Medium      |
|    3 | 192.168.1.102 | Russia        | 2022-01-03 14:00:00 | 2022-01-03 14:00:00 | Low         |
|    4 | 192.168.1.103 | India         | 2022-01-04 15:00:00 | 2022-01-04 15:00:00 | High        |
|    5 | 192.168.1.104 | Brazil        | 2022-01-05 16:00:00 | 2022-01-05 16:00:00 | Medium      |
+------+--------------+----------------+---------------------+---------------------+-------------+
5 rows in set (0.00 sec)
```

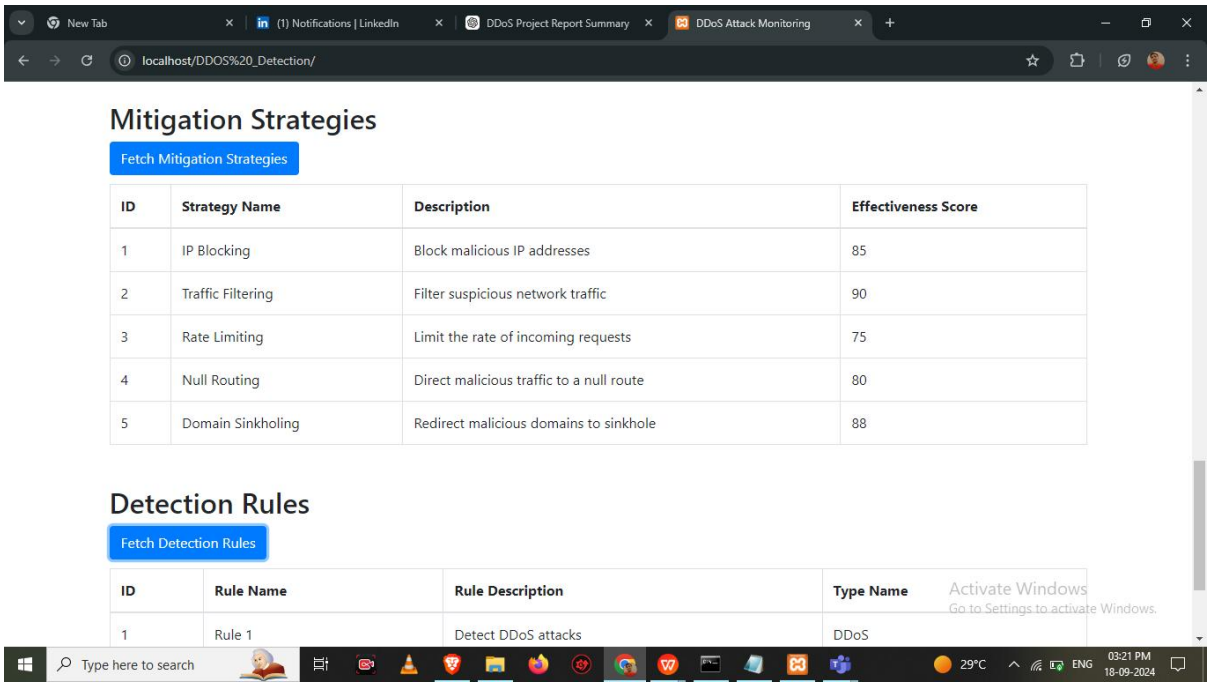
Retrieve all attack types with corresponding detection rules and attacks:

```
mysql> select at.*,dr.Rule_Name,a.Attack_Date from Attack_Types at join Detection_Rules dr on dr.Rule_Description lik
e concat('%',at.Type_Name,'%') join Attacks a on a.Attack_Type=at.Id;
+------+-------------+-----------------------------------+-----------+---------------------+
| Id   | Type_Name   | Description                       | Rule_Name | Attack_Date         |
+------+-------------+-----------------------------------+-----------+---------------------+
|    1 | DDoS        | Distributed Denial of Service     | Rule 1    | 2022-01-01 12:00:00 |
|    2 | SQL Injection | Structured Query Language Injection | Rule 2 | 2022-01-02 13:00:00 |
|    1 | DDoS        | Distributed Denial of Service     | Rule 1    | 2022-01-04 15:00:00 |
|    2 | SQL Injection | Structured Query Language Injection | Rule 2 | 2022-01-05 16:00:00 |
+------+-------------+-----------------------------------+-----------+---------------------+
4 rows in set (0.00 sec)
```

6)Project Output:-

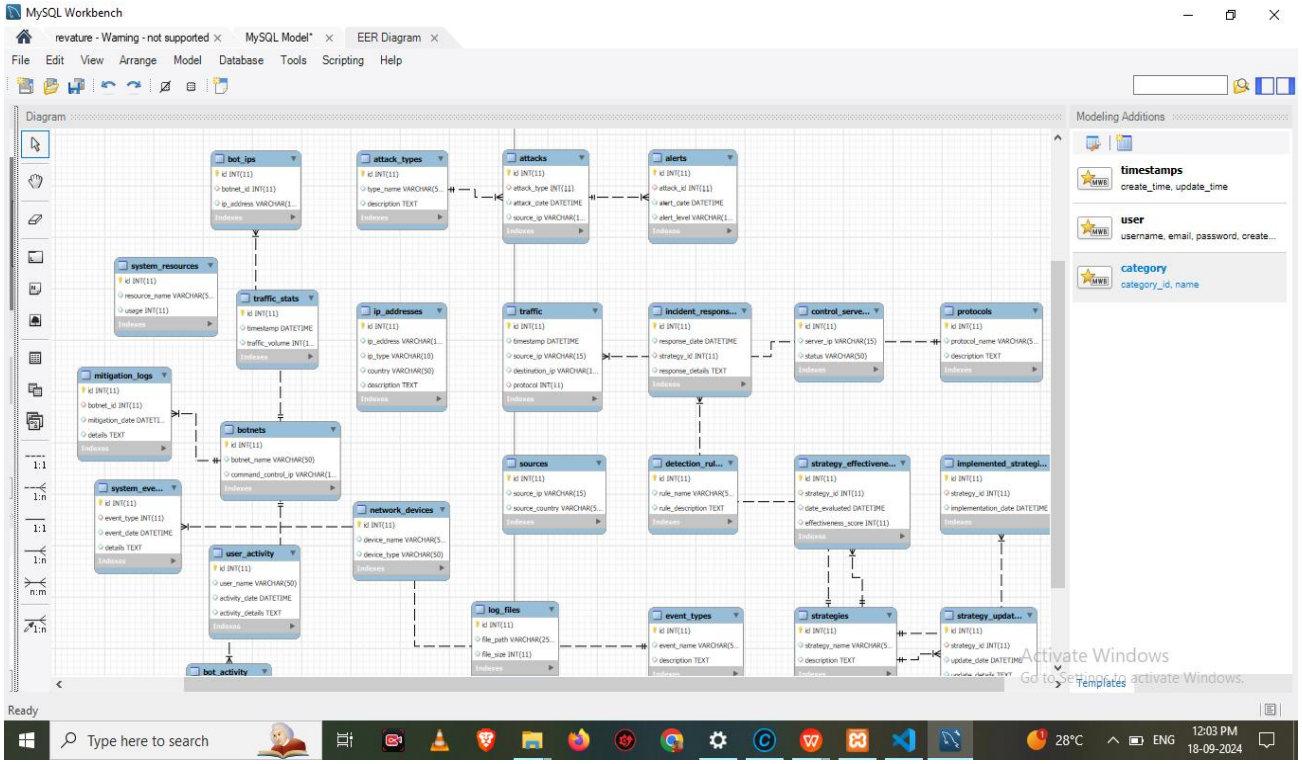



New Tab
(1) Notifications | LinkedIn
DDoS Project Report Summary
DDoS Attack Monitoring
localhost/DDOS%20_Detection/
System Logs
Fetch System Logs
ID
Event Name
Event Date
Details
1
Login
2022-01-01 12:00:00
User admin logged in
2
Error
2022-01-02 13:00:00
Disk space low
3
Update
2022-01-03 14:00:00
System update installed
4
Shutdown
2022-01-04 15:00:00
System shutdown
5
Restart
2022-01-05 16:00:00
System restarted
Botnet Information
Fetch Botnet Info
ID
Botnet Name
Command Control IP
Status
1
Botnet1
192.168.1.200
Active
Activate Windows
Go to Settings to activate Windows.
Type here to search
29°C
ENG
03:21 PM
18-09-2024


New Tab
(1) Notifications | LinkedIn
DDoS Project Report Summary
DDoS Attack Monitoring
localhost/DDOS%20_Detection/
Mitigation Strategies
Fetch Mitigation Strategies
ID
Strategy Name
Description
Effectiveness Score
1
IP Blocking
Block malicious IP addresses
85
2
Traffic Filtering
Filter suspicious network traffic
90
3
Rate Limiting
Limit the rate of incoming requests
75
4
Null Routing
Direct malicious traffic to a null route
80
5
Domain Sinkholing
Redirect malicious domains to sinkhole
88
Detection Rules
Fetch Detection Rules
ID
Rule Name
Rule Description
Type Name
1
Rule 1
Detect DDoS attacks
DDoS
Activate Windows
Go to Settings to activate Windows.
Type here to search
29°C
ENG
03:21 PM
18-09-2024

7)ER Diagram :-

An Entity-Relationship Diagram (ER Diagram) is a visual representation of the relationships between entities in a database. It helps in designing and modeling a database by illustrating entities, attributes, and the connections between them.



8) Final Goal of the Project:

The final goal of the project is to develop a robust, scalable, and secure system that detects, monitors, and mitigates various types of cyber-attacks, like DDoS. By integrating real-time alerts, detection rules, and advanced analytics, the system aims to enhance overall cybersecurity and protect critical assets from threats.