PROJECT

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

Prepared By:
Nandkumar Chavan

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.

5) **DDoS Mitigation Services:**

Specialized services (such as Cloudflare, AWS Shield, or Akamai) are designed to detect and mitigate DDoS attacks. These services filter malicious traffic, absorb the excess load, and ensure that only legitimate requests reach the server.

2) 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   |
| day2                 |
| db                   |
| db1                  |
| hrmszplus            |
| information_schema   |
| mitigation_strategies |
| mysql                |
| nandu                |
| network_traffic      |
| performance_schema   |
| sakila               |
| school_management    |
| sys                  |
| system_logging       |
| tasks_day2           |
| world                |
+----------------------+
18 rows in set (0.05 sec)
```

3) Tables used in each of the Databases:

Using **first database**, named as **‘Attack_Detection’**:

```
mysql> USE Attack_Detection;
Database changed
mysql>
```

- Creating first table, named as ‘Attacks’:

```
mysql> CREATE TABLE attacks (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     attack_type INT,
    ->     attack_date DATETIME,
    ->     source_ip VARCHAR(15),
    ->     FOREIGN KEY (attack_type) REFERENCES attack_types(id)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Displaying the entire table:

```
mysql> select * from Attacks;
+------+-------------+---------------------+---------------+
| Id   | Attack_Type | Attack_Date         | Source_IP     |
+------+-------------+---------------------+---------------+
|    1 |           1 | 2022-01-01 12:00:00 | 192.168.1.100 |
|    2 |           2 | 2022-01-02 13:00:00 | 192.168.1.101 |
|    3 |           3 | 2022-01-03 14:00:00 | 192.168.1.102 |
|    4 |           1 | 2022-01-04 15:00:00 | 192.168.1.103 |
|    5 |           2 | 2022-01-05 16:00:00 | 192.168.1.104 |
+------+-------------+---------------------+---------------+
5 rows in set (0.00 sec)
```

- Creating second table, named as ‘Attack_Types’ :

```
mysql> -- Create Tables
mysql> CREATE TABLE attack_types (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     type_name VARCHAR(50),
    ->     description TEXT
    -> );
Query OK, 0 rows affected (0.05 sec)
```

Displaying the entire table:

```
mysql> select * from Attack_Types;
+----+---------------------+-----------------------------------+
| id | type_name           | description                       |
+----+---------------------+-----------------------------------+
|  1 | DDoS                | Distributed Denial of Service     |
|  2 | SQL Injection       | Structured Query Language Injection |
|  3 | Cross-Site Scripting | XSS                               |
|  4 | Brute Force         | Password Guessing                 |
|  5 | Phishing            | Social Engineering                |
+----+---------------------+-----------------------------------+
5 rows in set (0.00 sec)
```

- Creating third table, named as ‘Sources’, and displaying it:

```
mysql>
mysql> CREATE TABLE sources (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     source_ip VARCHAR(15),
    ->     source_country VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select * from Sources;
+----+---------------+----------------+
| id | source_ip     | source_country |
+----+---------------+----------------+
|  1 | 192.168.1.100 | USA            |
|  2 | 192.168.1.101 | China          |
|  3 | 192.168.1.102 | Russia         |
|  4 | 192.168.1.103 | India          |
|  5 | 192.168.1.104 | Brazil         |
+----+---------------+----------------+
5 rows in set (0.00 sec)
```

- Creating Fourth table, named as ‘Detection_Rules’ and displaying it:

```
mysql>
mysql> CREATE TABLE detection_rules (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     rule_name VARCHAR(50),
    ->     rule_description TEXT
    -> );
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select * from Detection_Rules;
+----+-----------+---------------------+
| id | rule_name | rule_description    |
+----+-----------+---------------------+
|  1 | Rule 1    | Detect DDoS attacks |
|  2 | Rule 2    | Detect SQL Injection |
|  3 | Rule 3    | Detect XSS          |
|  4 | Rule 4    | Detect Brute Force  |
|  5 | Rule 5    | Detect Phishing     |
+----+-----------+---------------------+
5 rows in set (0.00 sec)
```

- Creating Fifth table, named as ‘Alerts’ and displaying it:

```
mysql>
mysql> CREATE TABLE alerts (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     attack_id INT,
    ->     alert_date DATETIME,
    ->     alert_level VARCHAR(10),
    ->     FOREIGN KEY (attack_id) REFERENCES attacks(id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> select * from Alerts;
+----+-----------+---------------------+-------------+
| id | attack_id | alert_date          | alert_level |
+----+-----------+---------------------+-------------+
|  1 |         1 | 2022-01-01 12:00:00 | High        |
|  2 |         2 | 2022-01-02 13:00:00 | Medium      |
|  3 |         3 | 2022-01-03 14:00:00 | Low         |
|  4 |         4 | 2022-01-04 15:00:00 | High        |
|  5 |         5 | 2022-01-05 16:00:00 | Medium      |
+----+-----------+---------------------+-------------+
5 rows in set (0.00 sec)
```

To list the tables available in database ‘Attack_Detection’:

```
mysql> show tables;
+----------------------------+
| Tables_in_attack_detection |
+----------------------------+
| alerts                     |
| attack_types               |
| attacks                    |
| detection_rules            |
| sources                    |
+----------------------------+
5 rows in set (0.05 sec)
```

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 sources 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 |           4 | 2022-01-04 15:00:00 | 192.168.1.103 | Brute Force          | India          |
|  5 |           5 | 2022-01-05 16:00:00 | 192.168.1.104 | Phishing             | Brazil         |
+----+-------------+---------------------+--------------+----------------------+----------------+
5 rows in set (0.04 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 (0.05 sec)
```

Retrieve all alerts with corresponding attack information and alert level:

```
mysql> SELECT al.*, a.attack_date, a.attack_type, at.type_name
    -> FROM alerts al
    -> JOIN attacks a ON al.attack_id = a.id
    -> JOIN attack_types at ON a.attack_type = at.id;
+----+-----------+---------------------+-------------+---------------------+-------------+----------------------+
| id | attack_id | alert_date          | alert_level | attack_date         | attack_type | type_name            |
+----+-----------+---------------------+-------------+---------------------+-------------+----------------------+
|  1 |         1 | 2022-01-01 12:00:00 | High        | 2022-01-01 12:00:00 |           1 | DDoS                 |
|  2 |         2 | 2022-01-02 13:00:00 | Medium      | 2022-01-02 13:00:00 |           2 | SQL Injection        |
|  3 |         3 | 2022-01-03 14:00:00 | Low         | 2022-01-03 14:00:00 |           3 | Cross-Site Scripting |
|  4 |         4 | 2022-01-04 15:00:00 | High        | 2022-01-04 15:00:00 |           4 | Brute Force          |
|  5 |         5 | 2022-01-05 16:00:00 | Medium      | 2022-01-05 16:00:00 |           5 | Phishing             |
+----+-----------+---------------------+-------------+---------------------+-------------+----------------------+
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 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 (0.05 sec)
```

5) Final Goal of the Project:

The ultimate objective of the project is to create a highly secure, scalable, and reliable system capable of detecting, monitoring, and mitigating a wide range of cyber-attacks, including DDoS. By incorporating real-time alerts, detection rules, and sophisticated analytics, the system seeks to strengthen cybersecurity defenses and safeguard essential resources against potential threats.