PROJECT

DENIAL OF SERVICE

USING MYSQL

RELATIONAL DATABASE STRUCTURE BASED ON NETWORK SECURITY

Prepared By:

Mansi Singh

Guided By:

Zakir Hussain

TABLE OF CONTENTS

1. DoS Attack Description.

What is a Denial of Service (DoS) Attack?

A **Denial of Service (DoS) attack** is a malicious attempt to make a network service, server, or website unavailable to its intended users by overwhelming it with a flood of illegitimate traffic or requests. This disrupts normal operations and prevents legitimate users from accessing the resource.

Here's how a DoS attack works:

1. **Target Overload**: The attacker sends an excessive amount of traffic (e.g., data packets) to the target, which can be a server, network, or website. The goal is to overload the system's capacity to process requests.
2. **Resource Exhaustion**: Servers and networks have limited resources like bandwidth, memory, and processing power. The overwhelming traffic depletes these resources, causing the system to slow down or crash.
3. **Service Disruption**: Once the system is overwhelmed, it can no longer handle legitimate user requests, effectively denying service to real users.

Types of DoS Attacks

There are three main types of DoS attacks, each with its own unique characteristics:

- **Volume-based attacks**: These attacks focus on overwhelming the bandwidth of a target system by sending massive amounts of data or requests.

  ICMP Flood (Ping Flood): Overwhelms the target with Internet Control Message Protocol (ICMP) echo requests (pings).

  UDP Flood: Bombards a target with User Datagram Protocol (UDP) packets, overwhelming its ability to process them.

- **Protocol attacks**: These attacks exploit weaknesses in network protocols to consume resources. It's like finding a vulnerability in a system's security software, allowing attackers to gain unauthorized access. Protocol attacks can be particularly devastating, as they can affect multiple systems and networks.

Application attacks: These attacks target specific applications or services to consume resources.

HTTP Flood: Sends seemingly legitimate HTTP requests, overwhelming web servers with excessive requests.

DNS Flood: Bombards the DNS server with requests, disrupting the resolution of domain names.

Common DoS Attack Techniques

- **Flooding**: Sending a large number of requests or traffic to overwhelm the resource. This is like trying to drink from a firehose, where the sheer volume of water is too much for the system to handle.
- **Buffer overflow**: Sending more data than a buffer can handle, causing it to crash. This is like trying to stuff too many people into a small room, causing it to become overcrowded and unusable.
- **Malformed packets**: Sending packets with incorrect or malicious data to cause errors. This is like sending a package with the wrong address, causing it to get lost in transit.
- **SYN flooding**: Sending a large number of SYN requests to consume server resources. This is like sending a flood of requests to a server, overwhelming it with traffic and causing it to slow down or crash.
- **Smurf Attack:**The attacker sends ICMP requests with a spoofed source address to a broadcast IP address. The responses are directed to the victim's system, overwhelming it with replies.

How DoS Attacks are Launched

DoS attacks can be launched using various tools and techniques, including:

- **Botnets**: Networks of compromised devices used to launch attacks. These devices can be infected with malware, allowing attackers to remotely control them and launch attacks.
- **Software Bugs**: Attackers exploit known vulnerabilities in the software or operating system running on the target machine. For example, a buffer overflow attack sends more data than the system can handle, causing it to crash.

- **Malware**: Software designed to harm or exploit systems. Malware can be used to launch DoS attacks, as well as steal sensitive data or disrupt system operations.
- **Scripting**: Using scripts to automate attack processes. This allows attackers to launch complex attacks with minimal effort, making it easier to carry out large-scale DoS attacks.

Protecting Against DoS Attacks

To protect against DoS attacks, organizations can use a variety of techniques, including:

- **Firewalls**: To block malicious traffic and prevent attacks from reaching the system.
- **Intrusion Detection/Prevention Systems (IDS/IPS)**: To detect and prevent attacks in real-time, reducing the risk of system compromise.
- **Load balancing**: To distribute traffic across multiple resources, reducing the risk of overload and improving system performance.
- **Content Delivery Networks (CDNs)**: To cache content and reduce server load, making it more difficult for attackers to launch successful DoS attacks.

- **DDoS mitigation services**: Specialized services that detect and mitigate DoS attacks, providing an additional layer of protection for organizations.

2. Databases used in this Project.

- Create five database using the below syntax:

create database [name of database];

```
mysql> CREATE DATABASE AttackDetection;
Query OK, 1 row affected (0.04 sec)

mysql> CREATE DATABASE Networktraffic;
Query OK, 1 row affected (0.04 sec)

mysql> CREATE DATABASE SystemResources;
Query OK, 1 row affected (0.03 sec)

mysql> CREATE DATABASE IncidentResponse;
Query OK, 1 row affected (0.01 sec)

mysql> CREATE DATABASE SecurityInformation;
Query OK, 1 row affected (0.02 sec)
```

3) Tables used in each of the Databases:

Using first database, ‘Attack_Detection’ ,

Create 5 tables namely,

1) Attacks
2) Attack_types
3) Sources
4) Detection_rules
5) Alerts

```
mysql> USE AttackDetection;
Database changed
mysql> CREATE TABLE Attacks(id int(10 ) Primary key,attacktype varchar(50), attackdate datetime , sourceip varchar(15));
Query OK, 0 rows affected, 1 warning (0.09 sec)

mysql> CREATE TABLE attack_types (id INT PRIMARY KEY, type_name VARCHAR(50), description VARCHAR(255);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
mysql> CREATE TABLE attack_types (id INT PRIMARY KEY, type_name VARCHAR(50), description VARCHAR(255));
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE sources (id INT PRIMARY KEY,source_ip VARCHAR(15),source_country VARCHAR(50));
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE detection_rules (id INT PRIMARY KEY,rule_name VARCHAR(50),rule_description VARCHAR(200));
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE alerts (id INT PRIMARY KEY, attack_id INT, alert_date DATETIME, alert_level VARCHAR(10), FOREIGN KEY (attack_id) REFERENCES attacks(id));
Query OK, 0 rows affected (0.10 sec)
```

Inserting data in each tables and displaying the entire table:

1.’Attacks’ table:

An attack table in the context of database security or network security typically refers to a table used for storing information about security incidents, attack attempts, or suspicious activities on a system. In the case of your project, if you're simulating attacks or monitoring security events, an attack table could store detailed records of those attacks.

```
mysql> INSERT INTO attacks values(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");
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM Attacks;
+----+------------+---------------------+--------------+
| id | attacktype | attackdate          | sourceip     |
+----+------------+---------------------+--------------+
|  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 (0.01 sec)
```

2.'Attack_types' Table:

The Attack_types table is designed to store information about various categories of attacks. This table can be used to define and classify different types of attacks that may be logged in an attack monitoring or incident tracking system.

```
mysql> INSERT INTO attack_types values(1, "DDoS", "Distributed Denial of Service"),(2, "SQL Injection", "Structured Query Language Injection"),(3, "Cross-Site Scripting", "X
SS");
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

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                               |
+----+---------------------+-----------------------------------+
3 rows in set (0.00 sec)
```

3.'Sources' Table:

The Sources table is used to store information about the origin of attacks or traffic, such as the IP address and the country of the attacker or the traffic source. This table is particularly useful in monitoring, tracking, and analyzing where attacks or suspicious activities are coming from, aiding in geo-analysis and threat detection.

```
mysql> INSERT INTO sources values(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");
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

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)
```

4.'Detection Rules' Table:

The Detection_rules table is designed to store the rules used by a system to detect suspicious activities, intrusions, or attacks. Each rule typically defines a pattern, condition, or threshold that triggers an alert or logs an incident when matched or exceeded.

```
mysql> INSERT INTO detection_rules VALUES(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");
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

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)
```

5.'Alerts' Table:

The Alerts table is designed to log alert notifications that are generated when a potential attack or suspicious activity is detected. Each alert is associated with an attack or incident and contains information such as the date of the alert and its severity level.

```
mysql> INSERT INTO alerts values(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");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

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)
```

Using second database, named as ‘Network_Traffic’,
Create 5 tables namely,

1.Traffic
2.Protocols
3.Ipadd
4.NetworkDevices
5.TrafficStats

```
mysql> USE networktraffic;
Database changed
mysql> CREATE TABLE traffic(id INT PRIMARY KEY, timestamp DATETIME, source_ip VARCHAR(15), destination_ip VARCHAR(15), protocol VARCHAR(10));
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE protocols(id INT PRIMARY KEY,protocolname VARCHAR(30),protocoldescription VARCHAR(200));
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE ipadd(id INT PRIMARY KEY, ip_address VARCHAR(15), ip_type VARCHAR(10));
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE networkdevices(id INT PRIMARY KEY, device_name VARCHAR(50), device_type VARCHAR(50));
Query OK, 0 rows affected (0.05 sec)

mysql> CRAETE TABLE trafficstats(id INT PRIMARY KEY, timestamp DATETIME, traffic_volume INT);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'CRAETE TABLE trafficstats(id INT PRIMARY KEY, timestamp DATETIME, traffic_volume'
at line 1
mysql> CREATE TABLE trafficstats(id INT PRIMARY KEY, timestamp DATETIME, traffic_volume INT);
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO traffic values(1, "2022-01-01 12:00:00", "192.168.1.100", "192.168.1.1", "TCP"),(2, "2022-01-02 13:00:00", "192.168.1.101", "192.168.1.2", "UDP"),(3, "2022-01-03 14:00:00", "192.168.1.102", "192.168.1.3", "HTTP"),(4, "
2022-01-04 15:00:00", "192.168.1.103", "192.168.1.4", "FTP"),(5, "2022-01-05 16:00:00", "192.168.1.104", "192.168.1.5", "SSH");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Inserting data in each tables and displaying the entire table:

1.’Traffic’ Table:

The Traffic table logs network traffic data, storing the source and destination IP addresses, the protocol used, and the timestamp of the traffic event. It helps in monitoring and analyzing network activity to detect suspicious patterns or attacks.

```
mysql> INSERT INTO traffic values(1, "2022-01-01 12:00:00", "192.168.1.100", "192.168.1.1", "TCP"),(2, "2022-01-02 13:00:00", "192.168.1.101", "192.168.1.2", "UDP"),(3, "2022-01-03 14:00:00", "192.168.1.102", "192.168.1.3", "HTTP"),(4, "
2022-01-04 15:00:00", "192.168.1.103", "192.168.1.4", "FTP"),(5, "2022-01-05 16:00:00", "192.168.1.104", "192.168.1.5", "SSH");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM traffic;
+----+---------------------+--------------+----------------+----------+
| id | timestamp           | source_ip    | destination_ip | protocol |
+----+---------------------+--------------+----------------+----------+
|  1 | 2022-01-01 12:00:00 | 192.168.1.100 | 192.168.1.1    | TCP      |
|  2 | 2022-01-02 13:00:00 | 192.168.1.101 | 192.168.1.2    | UDP      |
|  3 | 2022-01-03 14:00:00 | 192.168.1.102 | 192.168.1.3    | HTTP     |
|  4 | 2022-01-04 15:00:00 | 192.168.1.103 | 192.168.1.4    | FTP      |
|  5 | 2022-01-05 16:00:00 | 192.168.1.104 | 192.168.1.5    | SSH      |
+----+---------------------+--------------+----------------+----------+
5 rows in set (0.00 sec)
```

2.’Protocols’ Table:

The Protocols table stores information about network protocols, including a unique ID, the protocol name, and a description of how the protocol operates. It's useful for categorizing and understanding different types of network communication.

```
mysql> INSERT INTO protocols values(1, "TCP", "Transmission Control Protocol"),(2, "UDP", "User Datagram Protocol"),(3, "HTTP", "Hypertext Transfer Protocol"),(4, "FTP", "File Transfer Protocol"),(5, "SSH", "Secure Shell");
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM protocols;
+----+--------------+-----------------------------+
| id | protocolname | protocoldescription         |
+----+--------------+-----------------------------+
|  1 | TCP          | Transmission Control Protocol |
|  2 | UDP          | User Datagram Protocol      |
|  3 | HTTP         | Hypertext Transfer Protocol |
|  4 | FTP          | File Transfer Protocol      |
|  5 | SSH          | Secure Shell                |
+----+--------------+-----------------------------+
5 rows in set (0.00 sec)
```

3.'Ipadd' Table:

The ip_addresses table stores IP addresses along with their type, such as IPv4 or IPv6. This table is useful for tracking, managing, and categorizing IP addresses involved in network traffic, attacks, or other system activities.

```
mysql> INSERT INTO ipadd values(1, "192.168.1.100", "Public"),(2, "192.168.1.101", "Private"),(3, "192.168.1.102", "Public"),(4, "192.168.1.103", "Private"),(5, "192.168.1.104", "Public");
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM ipadd;
+----+--------------+---------+
| id | ip_address   | ip_type |
+----+--------------+---------+
|  1 | 192.168.1.100 | Public  |
|  2 | 192.168.1.101 | Private |
|  3 | 192.168.1.102 | Public  |
|  4 | 192.168.1.103 | Private |
|  5 | 192.168.1.104 | Public  |
+----+--------------+---------+
5 rows in set (0.00 sec)
```

4.'NetworkDevices' Table:

The network_devices table stores information about devices in a network, including their name and type (e.g., router, switch, firewall). This table is useful for managing and identifying the various devices involved in network infrastructure.

```
mysql> INSERT INTO networkdevices values(1, "Router", "Cisco"),(2, "Switch", "HP"),(3, "Firewall", "Juniper"),(4, "Server", "Dell"),(5, "Client", "Laptop");
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM networkdevices;
+----+-------------+-------------+
| id | device_name | device_type |
+----+-------------+-------------+
|  1 | Router      | Cisco       |
|  2 | Switch      | HP          |
|  3 | Firewall    | Juniper     |
|  4 | Server      | Dell        |
|  5 | Client      | Laptop      |
+----+-------------+-------------+
5 rows in set (0.00 sec)
```

5.'TrafficStats' Table:

The traffic_stats table logs network traffic statistics, specifically tracking the volume of traffic over time. This table helps monitor network load, detect unusual spikes, and analyze overall traffic patterns for performance and security.

Structure of the traffic_stats

```
mysql> INSERT INTO trafficstats VALUES(1, '2022-01-01 12:00:00', 1000),(2, '2022-02-15 14:30:00', 1500),(3, '2022-03-10 09:45:00', 2000),(4, '2022-04-22 18:00:00', 2500),(5, '2022-05-05 22:15:00', 1800);
Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM trafficstats;
+----+---------------------+----------------+
| id | timestamp           | traffic_volume |
+----+---------------------+----------------+
|  1 | 2022-01-01 12:00:00 |           1000 |
|  2 | 2022-02-15 14:30:00 |           1500 |
|  3 | 2022-03-10 09:45:00 |           2000 |
|  4 | 2022-04-22 18:00:00 |           2500 |
|  5 | 2022-05-05 22:15:00 |           1800 |
+----+---------------------+----------------+
5 rows in set (0.00 sec)
```

Using Third database, named as 'System_Resources',
Create 5 tables namely,

1.Resource_usage
2.Resources
3.System_stats
4.Process_list
5.User_sessions

```
mysql> USE SystemResources;
Database changed
mysql> CREATE TABLE resource_usage ( id INT PRIMARY KEY, timestamp DATETIME,  cpu_usage FLOAT, memory_usage FLOAT, disk_usage FLOAT);
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE resources ( id INT PRIMARY KEY, resource_name VARCHAR(50), resource_description VARCHAR(200));
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE system_stats ( id INT PRIMARY KEY,timestamp DATETIME,  system_load FLOAT, system_uptime INT);
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE process_list (id INT PRIMARY KEY, process_name VARCHAR(50),process_pid INT, process_cpu_usage FLOAT);
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE user_sessions (id INT PRIMARY KEY,user_id INT,session_start DATETIME,session_end DATETIME);
Query OK, 0 rows affected (0.05 sec)
```

Inserting data in each tables and displaying the entire table:

1.Resource_usage:

The resource_usage table tracks the utilization of system resources over time, including CPU, memory, and disk usage. This table helps monitor system performance and diagnose potential issues related to resource consumption.

```
mysql> INSERT INTO resource_usage (id, timestamp, cpu_usage, memory_usage, disk_usage) VALUES(1, '2022-01-10 10:00:00', 70, 65, 80),(2, '2022-02-12 14:30:00', 50, 55, 60),(3
30, 40, 50),(5, '2022-05-25 09:15:00', 80, 75, 85);
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM resource_usage;
+----+---------------------+-----------+--------------+------------+
| id | timestamp           | cpu_usage | memory_usage | disk_usage |
+----+---------------------+-----------+--------------+------------+
|  1 | 2022-01-10 10:00:00 |        70 |           65 |         80 |
|  2 | 2022-02-12 14:30:00 |        50 |           55 |         60 |
|  3 | 2022-03-15 16:45:00 |        90 |           85 |         95 |
|  4 | 2022-04-20 11:20:00 |        30 |           40 |         50 |
|  5 | 2022-05-25 09:15:00 |        80 |           75 |         85 |
+----+---------------------+-----------+--------------+------------+
5 rows in set (0.00 sec)
```

2.Resources

The resources table stores information about various system resources, including their name and a description. This table helps in managing and categorizing different resources that are monitored or analysed in a system.

```
mysql> INSERT INTO resources (id, resource_name, resource_description) VALUES(1, 'CPU', 'Processes instructions from programs'),(2, 'Memory', 'Stores temporary data for task
ork Adapter', 'Handles network connections'),(5, 'GPU', 'Processes graphics for applications');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM resources;
+----+-----------------+-------------------------------------+
| id | resource_name   | resource_description                |
+----+-----------------+-------------------------------------+
|  1 | CPU             | Processes instructions from programs |
|  2 | Memory          | Stores temporary data for tasks      |
|  3 | Disk            | Stores permanent data on the system  |
|  4 | Network Adapter | Handles network connections          |
|  5 | GPU             | Processes graphics for applications  |
+----+-----------------+-------------------------------------+
5 rows in set (0.00 sec)
```

3.System_stats:

The system stats table tracks overall system performance metrics, including system load and uptime. This table helps in monitoring the health and stability of the system over time.

```
mysql>
mysql> INSERT INTO system_stats (id, timestamp, system_load, system_uptime) VALUES(1, '2022-01-05 08:00:00', 75, 5),(2, '2022-02-15 12:30:00', 50, 3),(3, '2022-03-25 10:45:0
15:00', 90, 7);
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM system_stats;
+----+---------------------+-------------+--------------+
| id | timestamp           | system_load | system_uptime |
+----+---------------------+-------------+--------------+
|  1 | 2022-01-05 08:00:00 |          75 |            5 |
|  2 | 2022-02-15 12:30:00 |          50 |            3 |
|  3 | 2022-03-25 10:45:00 |          85 |            6 |
|  4 | 2022-04-18 14:20:00 |          40 |            2 |
|  5 | 2022-05-21 16:15:00 |          90 |            7 |
+----+---------------------+-------------+--------------+
5 rows in set (0.00 sec)
```

4.Process_list:

The process list table stores information about running processes on a system, including the process name, its process ID (PID), and its CPU usage. This table helps in monitoring and managing individual processes.

```
mysql> INSERT INTO process_list (id, process_name, process_pid, process_cpu_usage) VALUES(1, 'apache2', 101, 20),(2, 'mysql', 102, 35),(3, 'nginx', 103, 25),(4, 'java', 104,
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select* from process_list;
+----+--------------+-------------+-------------------+
| id | process_name | process_pid | process_cpu_usage |
+----+--------------+-------------+-------------------+
|  1 | apache2      |         101 |                20 |
|  2 | mysql        |         102 |                35 |
|  3 | nginx        |         103 |                25 |
|  4 | java         |         104 |                40 |
|  5 | python       |         105 |                15 |
+----+--------------+-------------+-------------------+
5 rows in set (0.00 sec)
```

5.User_sessions:

The user sessions table tracks user login sessions, including the user ID, session start time, and session end time. This table is useful for monitoring user activity, auditing, and analyzing session patterns.

```
mysql> INSERT INTO user_sessions (id, user_id, session_start, session_end) VALUES(1, 1, '2022-01-10 09:00:00', '2022-01-10 11:00:00'),(2, 2, '2022-02-12 13:00:00', '2022-02-
'),(4, 4, '2022-04-20 10:00:00', '2022-04-20 11:20:00'),(5, 5, '2022-05-25 08:15:00', '2022-05-25 09:15:00');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select* from user_sessions;
+----+---------+---------------------+---------------------+
| id | user_id | session_start       | session_end         |
+----+---------+---------------------+---------------------+
|  1 |       1 | 2022-01-10 09:00:00 | 2022-01-10 11:00:00 |
|  2 |       2 | 2022-02-12 13:00:00 | 2022-02-12 15:30:00 |
|  3 |       3 | 2022-03-15 15:00:00 | 2022-03-15 16:45:00 |
|  4 |       4 | 2022-04-20 10:00:00 | 2022-04-20 11:20:00 |
|  5 |       5 | 2022-05-25 08:15:00 | 2022-05-25 09:15:00 |
+----+---------+---------------------+---------------------+
5 rows in set (0.00 sec)
```

Using Fourth database, named as ‘Incident_Response’,

Create 5 tables namely,

1.Incidents

2.Incident_type

3.Response_plans

4.Response_teams

5.Incident_reports

```
mysql> USE IncidentResponse;
Database changed
mysql> CREATE TABLE incidents (id INT PRIMARY KEY, incident_date DATETIME,    incident_type INT, incident_description VARCHAR(200));
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE incident_types (id INT PRIMARY KEY,type_name VARCHAR(40),type_description VARCHAR(200));
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE response_plans ( id INT PRIMARY KEY, plan_name VARCHAR(50), plan_description VARCHAR(200));
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE response_teams ( id INT PRIMARY KEY, team_name VARCHAR(20), team_lead VARCHAR(30));
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE incident_reports (id INT PRIMARY KEY, incident_id INT, report_date DATETIME, report_description VARCHAR(200));
Query OK, 0 rows affected (0.05 sec)
```

Inserting data in each tables and displaying the entire table:

1.Incidents:

The incidents table is used to log details about various incidents, such as security breaches or system failures. It includes the date of the incident, the type of incident, and a description of what occurred.

```
mysql> INSERT INTO incidents (id, incident_date, incident_type, incident_description) VALUES(1, '2022-01-10 08:45:00', 1, 'DDoS attack targeting web server causing significa
pt detected on the user authentication system'),(3, '2022-03-05 16:00:00', 1, 'DDoS attack resulting in network congestion and service disruption'),(4, '2022-04-12 09:15:00'
5, '2022-05-25 14:00:00', 1, 'Brute force login attempt led to temporary system slowdown');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM incidents;
+----+---------------------+---------------+-----------------------------------------------------------------+
| id | incident_date       | incident_type | incident_description                                            |
+----+---------------------+---------------+-----------------------------------------------------------------+
|  1 | 2022-01-10 08:45:00 |             1 | DDoS attack targeting web server causing significant downtime   |
|  2 | 2022-02-20 11:30:00 |             2 | SQL Injection attempt detected on the user authentication system |
|  3 | 2022-03-05 16:00:00 |             1 | DDoS attack resulting in network congestion and service disruption |
|  4 | 2022-04-12 09:15:00 |             3 | Cross-Site Scripting (XSS) detected in web application          |
|  5 | 2022-05-25 14:00:00 |             1 | Brute force login attempt led to temporary system slowdown      |
+----+---------------------+---------------+-----------------------------------------------------------------+
5 rows in set (0.00 sec)
```

2.Incident_type:

The incident_types table categorizes different types of incidents by storing a unique identifier, a name for the type of incident, and a description of what each type entails. This table helps in classifying and managing incidents more effectively.

```
mysql> INSERT INTO incident_types (id, type_name, type_description) VALUES(1, 'DoS', 'Attack that makes a system unavailable'),(2, 'SQL Injection', 'Inserting harmful SQL in
y'),(4, 'Cross-Site Scripting (XSS)', 'Injecting harmful scripts into websites'),(5, 'Phishing', 'Tricking people to steal their data');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM incident_types;
+----+---------------------------+---------------------------------------+
| id | type_name                 | type_description                      |
+----+---------------------------+---------------------------------------+
|  1 | DoS                       | Attack that makes a system unavailable  |
|  2 | SQL Injection             | Inserting harmful SQL into a database   |
|  3 | Brute Force               | Guessing passwords repeatedly           |
|  4 | Cross-Site Scripting (XSS) | Injecting harmful scripts into websites |
|  5 | Phishing                  | Tricking people to steal their data     |
+----+---------------------------+---------------------------------------+
5 rows in set (0.00 sec)
```

3.Response_plans:

The response_plans table stores information about predefined plans or procedures for responding to various incidents. This helps ensure that responses to incidents are systematic and organized.

```
mysql> INSERT INTO response_plans (id, plan_name, plan_description) VALUES(1, 'DoS Mitigation Plan', 'Limit and filter traffic to stop DoS attacks'),(2, 'SQL Injection Plan'
, 'Use account lockout and CAPTCHA for protection'),(4, 'XSS Mitigation Plan', 'Clean inputs and fix web vulnerabilities'),(5, 'Phishing Plan', 'Train employees and monitor
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM response_plans;
+----+--------------------+----------------------------------------------+
| id | plan_name          | plan_description                             |
+----+--------------------+----------------------------------------------+
|  1 | DoS Mitigation Plan | Limit and filter traffic to stop DoS attacks   |
|  2 | SQL Injection Plan  | Block and prevent harmful SQL inputs           |
|  3 | Brute Force Plan    | Use account lockout and CAPTCHA for protection |
|  4 | XSS Mitigation Plan | Clean inputs and fix web vulnerabilities       |
|  5 | Phishing Plan       | Train employees and monitor for phishing       |
+----+--------------------+----------------------------------------------+
5 rows in set (0.00 sec)
```

4.Response_teams:

The response_teams table stores information about teams responsible for handling incidents, including the team's name and the team lead. This helps in organizing and managing the personnel involved in incident response.

```
mysql> INSERT INTO response_teams (id, team_name, team_lead) VALUES(1, 'Network Security Team', 'Rajesh Sharma'),(2, 'Application Security Team', 'Priya Iyer'),(3, 'Incident
 'Sneha Rao'),(5, 'Forensics Team', 'Vikram Desai');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM response_teams;
+----+---------------------------+--------------+
| id | team_name                 | team_lead    |
+----+---------------------------+--------------+
|  1 | Network Security Team     | Rajesh Sharma |
|  2 | Application Security Team | Priya Iyer    |
|  3 | Incident Management Team  | Amit Patel    |
|  4 | Threat Intelligence Team  | Sneha Rao     |
|  5 | Forensics Team            | Vikram Desai  |
+----+---------------------------+--------------+
5 rows in set (0.00 sec)
```

5.Incident_reports:

The incident_reports table records detailed reports for specific incidents, including the date of the report and a description of the incident. It helps in maintaining a comprehensive record of incidents and their handling.

```
mysql> INSERT INTO incident_reports (id, incident_id, report_date, report_description) VALUES(1, 1, '2022-01-16', 'DoS attack stopped by blocking IPs and limiting traffic'),
locking IPs'),(3, 3, '2022-03-23', 'SQL Injection blocked, inputs cleaned'),(4, 4, '2022-04-06', 'Brute force stopped with account lockout'),(5, 5, '2022-05-13', 'ICMP flood
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM incident_reports;
+----+-------------+---------------------+-----------------------------------------------------+
| id | incident_id | report_date         | report_description                                  |
+----+-------------+---------------------+-----------------------------------------------------+
|  1 |           1 | 2022-01-16 00:00:00 | DoS attack stopped by blocking IPs and limiting traffic |
|  2 |           2 | 2022-02-19 00:00:00 | SYN flood fixed by updating firewall and blocking IPs   |
|  3 |           3 | 2022-03-23 00:00:00 | SQL Injection blocked, inputs cleaned                   |
|  4 |           4 | 2022-04-06 00:00:00 | Brute force stopped with account lockout                |
|  5 |           5 | 2022-05-13 00:00:00 | ICMP flood blocked by stopping ICMP requests            |
+----+-------------+---------------------+-----------------------------------------------------+
5 rows in set (0.00 sec)
```

Using Fifth database, named as 'Security_Information',

Create 5 tables namely,

1.Vulnerabilities

2.Patches

3.Security_Advisories

4.Threat_Intelligence

5.Security_incidents

```
mysql> USE SecurityInformation;
Database changed
mysql> CREATE TABLE vulnerabilities ( id INT PRIMARY KEY, vuln_name VARCHAR(50), vuln_description VARCHAR(150), vuln_severity VARCHAR(10));
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE patches ( id INT PRIMARY KEY, patch_name VARCHAR(50), patch_description VARCHAR(200), patch_release_date DATE);
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE security_advisories (id INT PRIMARY KEY,advisory_name VARCHAR(40),advisory_description VARCHAR(150));
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE threat_intelligence (id INT PRIMARY KEY,threat_name VARCHAR(50),threat_description VARCHAR(150),threat_level VARCHAR(10));
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE security_incidents (id INT PRIMARY KEY,incident_id INT,security_incident_date DATETIME);
Query OK, 0 rows affected (0.05 sec)
```

Inserting data in each tables and displaying the entire table:

1.Vulnerabilities:

The vulnerabilities table is used to store information about security vulnerabilities, including their names, descriptions, and severity levels. This helps in tracking and managing vulnerabilities within a system.

```
mysql> INSERT INTO vulnerabilities (id, vuln_name, vuln_description, vuln_severity) VALUES(1, 'SQL Injection', 'Injecting SQL code in input fields', 'High'),(
2, 'XSS', 'Injecting malicious scripts in websites', 'Medium'),(3, 'Buffer Overflow', 'Sending excessive data to crash system', 'High'),(4, 'Weak Encryption',
 'Using weak encryption methods', 'Low'),(5, 'Phishing', 'Tricking users to steal data', 'Medium');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT* FROM vulnerabilities;
+----+-----------------+----------------------------------------+---------------+
| id | vuln_name       | vuln_description                       | vuln_severity |
+----+-----------------+----------------------------------------+---------------+
|  1 | SQL Injection   | Injecting SQL code in input fields     | High          |
|  2 | XSS             | Injecting malicious scripts in websites | Medium       |
|  3 | Buffer Overflow | Sending excessive data to crash system | High          |
|  4 | Weak Encryption | Using weak encryption methods          | Low           |
|  5 | Phishing        | Tricking users to steal data           | Medium        |
+----+-----------------+----------------------------------------+---------------+
5 rows in set (0.00 sec)
```

2.Patches

The patches table records information about patches or updates applied to systems or software to address vulnerabilities or improve functionality. It includes details about the patch name, description, and release date.

```
mysql> INSERT INTO patches (id, patch_name, patch_description, patch_release_date) VALUES(1, 'SQL Patch v1.2', 'Fix for SQL Injection vulnerability', '2022-01
-10'),(2, 'XSS Patch v2.1', 'Fix for Cross-Site Scripting', '2022-02-15'),(3, 'Buffer Overflow Patch v3.3', 'Fix for buffer overflow issue', '2022-03-20'),(4,
 'Encryption Patch v1.5', 'Update to strengthen encryption', '2022-04-05'),(5, 'Phishing Protection v2.0', 'Update to block phishing attempts', '2022-05-12');

Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT* FROM patches;
+----+---------------------------+-----------------------------------+--------------------+
| id | patch_name                | patch_description                 | patch_release_date |
+----+---------------------------+-----------------------------------+--------------------+
|  1 | SQL Patch v1.2            | Fix for SQL Injection vulnerability | 2022-01-10       |
|  2 | XSS Patch v2.1            | Fix for Cross-Site Scripting      | 2022-02-15         |
|  3 | Buffer Overflow Patch v3.3 | Fix for buffer overflow issue    | 2022-03-20         |
|  4 | Encryption Patch v1.5     | Update to strengthen encryption   | 2022-04-05         |
|  5 | Phishing Protection v2.0  | Update to block phishing attempts | 2022-05-12         |
+----+---------------------------+-----------------------------------+--------------------+
5 rows in set (0.00 sec)
```

3.Security_Advisories

The security_advisories table stores information about security advisories, which provide details about vulnerabilities, threats, or other security issues and recommended actions to address them. This table is essential for keeping track of important security updates and guidance.

```
mysql> INSERT INTO security_advisories (id, advisory_name, advisory_description) VALUES(1, 'SQL Injection Alert', 'Warning about SQL Injection attacks'),(2, '
XSS Alert', 'Advisory on XSS vulnerability in web apps'),(3, 'Buffer Overflow Alert', 'Advisory on buffer overflow risks'),(4, 'Weak Encryption Alert', 'Alert
 on using outdated encryption'),(5, 'Phishing Alert', 'Advisory on phishing attack risks');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT* FROM security_advisories;
+----+----------------------+-----------------------------------------+
| id | advisory_name        | advisory_description                    |
+----+----------------------+-----------------------------------------+
|  1 | SQL Injection Alert  | Warning about SQL Injection attacks     |
|  2 | XSS Alert            | Advisory on XSS vulnerability in web apps |
|  3 | Buffer Overflow Alert | Advisory on buffer overflow risks      |
|  4 | Weak Encryption Alert | Alert on using outdated encryption     |
|  5 | Phishing Alert       | Advisory on phishing attack risks       |
+----+----------------------+-----------------------------------------+
5 rows in set (0.00 sec)
```

4.Threat_Intelligence

The threat_intelligence table captures information about potential threats, including their names, descriptions, and threat levels. This helps in assessing and prioritizing threats based on their severity and potential impact.

```
mysql> INSERT INTO threat_intelligence (id, threat_name, threat_description, threat_level) VALUES(1, 'SQL Injection Threat', 'Hackers exploiting SQL vulnerabi
lities', 'High'),(2, 'XSS Threat', 'Attackers injecting malicious scripts', 'Medium'),(3, 'Phishing Threat', 'Cybercriminals targeting users via email', 'Medi
um'),(4, 'DDoS Threat', 'Hackers launching large-scale DDoS attacks', 'High'),(5, 'Ransomware Threat', 'Malware encrypting user data for ransom', 'High');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT* FROM threat_intelligence;
+----+----------------------+-----------------------------------------+--------------+
| id | threat_name          | threat_description                      | threat_level |
+----+----------------------+-----------------------------------------+--------------+
|  1 | SQL Injection Threat | Hackers exploiting SQL vulnerabilities  | High         |
|  2 | XSS Threat           | Attackers injecting malicious scripts   | Medium       |
|  3 | Phishing Threat      | Cybercriminals targeting users via email | Medium      |
|  4 | DDoS Threat          | Hackers launching large-scale DDoS attacks | High      |
|  5 | Ransomware Threat    | Malware encrypting user data for ransom | High         |
+----+----------------------+-----------------------------------------+--------------+
5 rows in set (0.00 sec)
```

5.Security_incidents:

The security_incidents table is designed to log specific security-related incidents, linking them to broader incident records and capturing the date when these security incidents occurred.

```
mysql> INSERT INTO security_incidents (id, incident_id, security_incident_date) VALUES(1, 1, '2022-01-25'),(2, 2, '2022-02-18'),(3, 3, '2022-03-12'),(4, 4, '2
022-04-05'),(5, 5, '2022-05-16');
Query OK, 5 rows affected (0.04 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT* FROM security_incidents;
+----+-------------+-----------------------+
| id | incident_id | security_incident_date |
+----+-------------+-----------------------+
|  1 |           1 | 2022-01-25 00:00:00    |
|  2 |           2 | 2022-02-18 00:00:00    |
|  3 |           3 | 2022-03-12 00:00:00    |
|  4 |           4 | 2022-04-05 00:00:00    |
|  5 |           5 | 2022-05-16 00:00:00    |
+----+-------------+-----------------------+
5 rows in set (0.00 sec)
```

4. Queries identified by the Network Infra security team.

Retrieve attacks by date range

```
mysql> SELECT * FROM attacks WHERE attack_date BETWEEN '2022-01-01' AND '2022-01-31';
+----+-------------+-------------+---------------+
| id | attack_type | attack_date | source_ip     |
+----+-------------+-------------+---------------+
|  1 | 1           | 2022-01-01  | 192.168.1.100 |
|  2 | 2           | 2022-01-02  | 192.168.1.101 |
|  3 | 3           | 2022-01-03  | 192.168.1.102 |
|  4 | 1           | 2022-01-04  | 192.168.1.103 |
|  5 | 2           | 2022-01-05  | 192.168.1.104 |
+----+-------------+-------------+---------------+
5 rows in set (0.00 sec)
```

Retrieve attacks by type (e.g., DDoS)

```
mysql> SELECT * FROM attacks WHERE attack_type = 1;
+----+-------------+-------------+---------------+
| id | attack_type | attack_date | source_ip     |
+----+-------------+-------------+---------------+
|  1 | 1           | 2022-01-01  | 192.168.1.100 |
|  4 | 1           | 2022-01-04  | 192.168.1.103 |
+----+-------------+-------------+---------------+
2 rows in set (0.01 sec)
```

Retrieve attacks by source IP

```
mysql> SELECT * FROM attacks WHERE source_ip = '192.168.1.100';
+----+-------------+-------------+---------------+
| id | attack_type | attack_date | source_ip     |
+----+-------------+-------------+---------------+
|  1 | 1           | 2022-01-01  | 192.168.1.100 |
+----+-------------+-------------+---------------+
1 row in set (0.01 sec)
```

Retrieve all attack_type

```
mysql> SELECT * FROM attack_type;
+----+---------------------+-----------------------------------+
| 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 (0.03 sec)
```

Retrieve attack type by name

```
mysql>  SELECT * FROM attack_type WHERE type_name = 'DDoS';
+----+-----------+-------------------------------+
| id | type_name | description                   |
+----+-----------+-------------------------------+
|  1 | DDoS      | Distributed Denial of Service |
+----+-----------+-------------------------------+
1 row in set (0.00 sec)
```

Retrieve attack types by description

```
mysql> SELECT * FROM attack_type WHERE description LIKE '%flooding%';
Empty set (0.01 sec)
```

Retrieve source by IP

```
mysql> SELECT * FROM sources WHERE source_ip = '192.168.1.100';
+----+---------------+----------------+
| id | source_ip     | source_country |
+----+---------------+----------------+
|  1 | 192.168.1.100 | USA            |
+----+---------------+----------------+
1 row in set (0.01 sec)
```

Retrieve sources by country

```
mysql> SELECT * FROM sources WHERE source_country = 'USA';
+----+---------------+----------------+
| id | source_ip     | source_country |
+----+---------------+----------------+
|  1 | 192.168.1.100 | USA            |
+----+---------------+----------------+
1 row in set (0.00 sec)
```

Retrieve detection rule by name

```
mysql> SELECT * FROM detection_rules WHERE rule_name = 'Rule 1';
+----+-----------+--------------------+
| id | rule_name | rule_description   |
+----+-----------+--------------------+
|  1 | Rule 1    | Detect DDoS attacks |
+----+-----------+--------------------+
1 row in set (0.00 sec)
```

Retrieve detection rules by description

```
mysql> SELECT * FROM detection_rules WHERE rule_description LIKE '%DDoS%';
+----+-----------+--------------------+
| id | rule_name | rule_description   |
+----+-----------+--------------------+
|  1 | Rule 1    | Detect DDoS attacks |
+----+-----------+--------------------+
1 row in set (0.00 sec)
```

Final Goal of the Project.:

This project's main goal is to create a flexible, secure, and robust system capable of identify, track, and lessen cyberattacks—especially Denial-of-service attacks. Through leveraging dynamic detection criteria, real-time alerts, and advanced analytics, the system will enhance cybersecurity overall and protect vital assets from constantly changing threats.