PROJECT

DENIAL OF SERVICE USING MYSQL

RELATIONAL DATABASE STRUCTURE BASED ON

NETWORK SECURITY

Prepared By:

Anuja Dixit

Guided By:

Zakir Hussain

TABLE OF CONTENTS

1) Denial of Service (DoS):

A Denial of Service (DoS) attack is a type of cyberattack where an attacker attempts to make a computer or network resource unavailable by overwhelming it with traffic or requests. The goal of a DoS attack is to exhaust the resource's capacity, making it unable to handle legitimate requests.

Types of DoS attacks:

1. **Volume-Based Attacks:** Volume-based attacks flood a network with too much data, overpowering its bandwidth and making the network unusable. Examples include UDP floods and ICMP floods.
2. **Protocol Attacks:** Protocol attacks exploit weaknesses in network protocols to use up server resources. Examples are SYN floods and the Ping of Death.
3. **Application Layer Attacks:** Application layer attacks target specific applications or services, causing them to crash or become very slow. Examples include HTTP floods and Slowloris.
4. **Distributed Denial-of-Service (DDoS) Attacks:** DDoS attacks use multiple systems, often compromised computers , to attack a single target. Examples are amplification attacks and botnet-based attacks.
5. **Resource Exhaustion:** This is when the hacker repeatedly requests access to a resource and eventually overloads the web application. The application slows down and finally crashes. In this case, the user is unable to get access to the webpage.
6. **Reflective Attacks:** Reflective attacks involve sending requests to third-party servers with the victim’s IP address. The servers unknowingly send responses to the victim, overwhelming it. Examples are DNS reflection and NTP reflection.

DoS attack common technique:

1. **Flooding:** This technique involves sending an overwhelming amount of requests or traffic to a target system, such as a server or network. The goal is to overload the target's resources, such as CPU, memory, or bandwidth, making it impossible to respond to legitimate traffic.
   Example: UDP flood, ICMP flood (Ping flood), HTTP flood.

2. **Buffer Overflow:** Buffer overflow occurs when an attacker sends more data to a buffer (a temporary storage area in memory) than it can handle.

This overflow causes the application to behave unexpectedly, often leading to crashes or even allowing the attacker to execute malicious code.

Example: An attacker exploits a vulnerable program to cause a buffer overflow, crashing the system or running malicious code.

3. **Malformed packets:** In this technique, the attacker sends packets (units of data transmitted over a network) that contain incorrect or malicious data. These malformed packets can exploit vulnerabilities in network protocols, causing errors or crashes in the target system.
   Example: The Ping of Death attack, where oversized or fragmented ICMP packets are sent to crash the target.

4. **SYN Flooding:** SYN flooding is a type of TCP-based attack where an attacker sends a large number of SYN (synchronize) requests, initiating a TCP connection but never completing the handshake. The server allocates resources for each incomplete connection, quickly exhausting its capacity to handle new connections.
   Example: An attacker floods a web server with SYN requests, making it unavailable to legitimate users.

DDoS attacks can be launched using various techniques, including:

1. **Botnets:** A botnet is a network of compromised devices (also known as "zombies") that are controlled remotely by an attacker, often without the device owner's knowledge. These devices can be used collectively to launch a Distributed Denial of Service (DDoS) attack by overwhelming a target with traffic from multiple sources.

2. **Malware:** Malware refers to malicious software designed to infiltrate and damage a computer system. In the context of DoS attacks, malware can be used to infect devices and turn them into bots for use in a botnet or directly attack systems by generating harmful traffic.

3. **Scripting:** Scripts can be used to automate the process of launching a DoS attack. By writing simple scripts, attackers can repeatedly send requests or malformed packets to the target without requiring extensive resources.

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 DoS 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 databases 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)
```

- To display the names of created databases:

show databases;

```
mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| abc                  |
| attack_detection     |
| incident_response    |
| information_schema   |
| mysql                |
| network_traffic      |
| performance_schema   |
| prod                 |
| revature             |
| sakila               |
| security_information |
| sys                  |
| system_resources     |
| world                |
+----------------------+
14 rows in set (0.21 sec)

mysql>
```

3) Tables used in each of the Databases:

Attack_Detection Database:

This database is focused on detecting and managing DoS attacks by logging information about ongoing and past attacks, the rules for detecting attacks, and generating alerts.

Using **first database**, named as **'Attack_Detection'**:

```
mysql> use Attack_Detection;
Database changed
```

Attacks Table:

This table logs each detected attack, storing details such as the type of attack, date, and the source IP from where the attack originated. This helps in tracking the specifics of each attack.

- Creating first table, named as 'Attacks':

```
mysql> create table Attacks(Id int, Attack_Type int, Attack_Date
 datetime, Source_IP varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into Attacks values(1, 1, "2022-01-01 12:00:00", "192.168.1.100");
Query OK, 1 row affected (0.03 sec)

mysql> insert into Attacks values(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, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

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)
```

Attack_Types:

It stores different types of attacks (e.g., DDoS ,SQL Injection) along with their descriptions. This helps in categorizing and understanding the nature of detected attacks.

-Creating second table, named as ‘Attack_Types’ and displaying it:

```
mysql> create table Attack_Types(Id int, Type_Name varchar(30), Description varchar(80));
Query OK, 0 rows affected (0.02 sec)

mysql> insert into Attack_Types values(1, "DDoS", "Distributed Denial of Service"),(2, "SQL Injection", "Structured Query Language In
jection"), (3, "Cross-Site Scripting", "XSS"), (4, "Brute Force", "Password Guessing"), (5, "Phishing", "Social Engineering");
Query OK, 5 rows affected (0.01 sec)
Records: 5  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                             |
|    4 | Brute Force        | Password Guessing                 |
|    5 | Phishing           | Social Engineering                |
+------+--------------------+-----------------------------------+
5 rows in set (0.00 sec)
```

Sources Table:

This table logs the source of the attack, including the IP address and geographical location. This is important for identifying the origin of malicious traffic.

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

```
mysql> create table Sources(Id int, Source_IP varchar(30), Source_Country varchar(30));
Query OK, 0 rows affected (0.03 sec)

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 (0.01 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)
```

-

Detection_Rules Table:

It contains rules for detecting different types of attacks, which could be used by an IDS (Intrusion Detection System). These rules help in automating the detection process.

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

```
mysql> create table Detection_Rules(Id int, Rule_Name varchar(20), Rule_Description varchar(50));
Query OK, 0 rows affected (0.03 sec)

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 (0.01 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)
```

Alerts Table:

This table logs alerts generated when an attack is detected, with details like the level of the alert (e.g., high, medium, low) and when it was raised. Alerts provide real-time notifications of potential threats.

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

-

```
mysql> create table Alerts(Id int, Attack_Id int, Alert_Date datetime, Alert_Level varchar(20));
Query OK, 0 rows affected (0.03 sec)

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)
```

-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 (0.00 sec)
```

Network_Traffic Database:

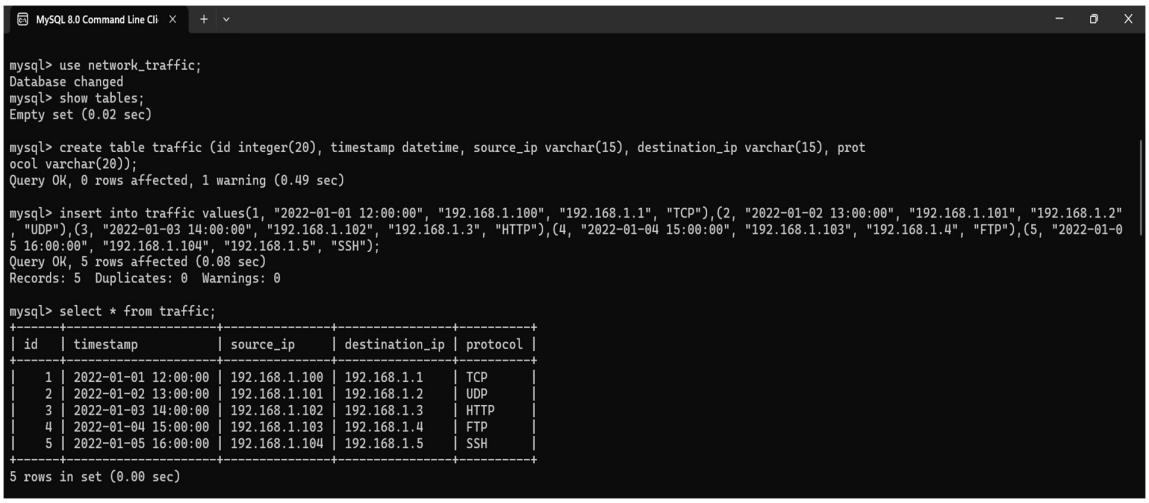
This database stores logs of network activity to monitor normal and abnormal traffic patterns, which are crucial in identifying potential DoS attacks.

Using **second database**, named as **'Network_Traffic'** :

Traffic Table:

Logs every packet that passes through the network, including the source IP, destination IP, and protocol used. Monitoring traffic is key to detecting anomalies.

- Creating first table, named as 'Traffic' and displaying it:

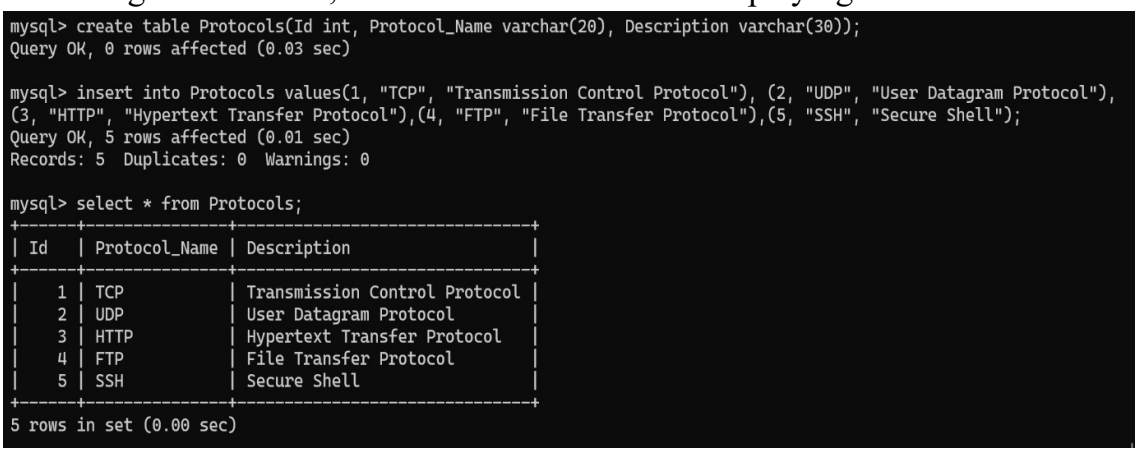


```
MySQL 8.0 Command Line Cli

mysql> use network_traffic;
Database changed
mysql> show tables;
Empty set (0.02 sec)

mysql> create table traffic (id integer(20), timestamp datetime, source_ip varchar(15), destination_ip varchar(15), prot
ocol varchar(20));
Query OK, 0 rows affected, 1 warning (0.49 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, "2022-01-0
5 16:00:00", "192.168.1.104", "192.168.1.5", "SSH");
Query OK, 5 rows affected (0.08 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)
```

Protocols Table:

Stores information about different network protocols (e.g., TCP, UDP, FTP) used in traffic. This helps identify which protocol is being exploited in the attack.

- Creating second table, named as 'Protocols' and displaying it:



```
mysql> create table Protocols(Id int, Protocol_Name varchar(20), Description varchar(30));
Query OK, 0 rows affected (0.03 sec)

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 (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from Protocols;
+------+---------------+-------------------------------+
| Id   | Protocol_Name | Description                   |
+------+---------------+-------------------------------+
|    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)
```

IP_Addresses Table:

Logs IP addresses (internal and external) along with their type (public, private), used to identify attackers and victims.

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

-
```
mysql> create table ip_addresses (id integer(20), ip_address varchar(15), ip_type varchar(10));
Query OK, 0 rows affected, 1 warning (1.07 sec)

mysql> insert into ip_addresses values(1, "192.168.1.100", "Public"),(2, "192.168.1.101", "Private"),(3, "192.168.1.102", "Public"),(4, "192.168.1.103", "Pr
ivate"),(5, "192.168.1.104", "Public");
Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from ip_addresses;
+------+---------------+---------+
| 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)
```

Network_devices Table:

Stores details about devices on the network (e.g., routers, firewalls). Monitoring device traffic can help identify bottlenecks or compromised devices.

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

```
5 rows in set (0.00 sec)

mysql> create table network_devices (id integer(20), device_name varchar(20), device_type varchar(20));
Query OK, 0 rows affected, 1 warning (0.54 sec)
```

```
MySQL 8.0 Command Line Cli

mysql> insert into  network_devices values(1, "Router", "Cisco"),(2, "Switch", "HP"),(3, "Firewall", "Juniper"),(4, "Server", "Dell"),(5, "Client", "Laptop"
);
Query OK, 5 rows affected (0.14 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select* from network_devices;
+------+-------------+-------------+
| 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)
```

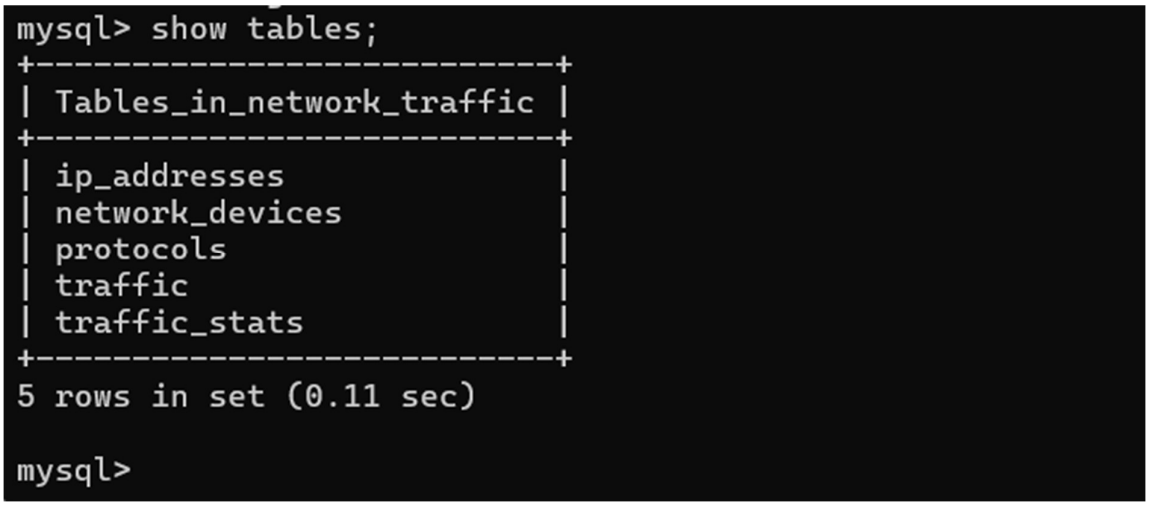
Traffic_Stats Table:

Logs traffic volume at different timestamps, helping to identify traffic spikes, which is a common sign of DoS attacks.

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

```
mysql> create table traffic_stats (id integer(20), timestamp datetime, traffic_volume integer(20));
Query OK, 0 rows affected, 2 warnings (0.39 sec)

mysql> insert into traffic_stats values(1, "2022-01-01 12:00:00", 1000),(2, "2022-01-03 10:00:00", 3000),(3, "2022-05-01
 15:55:00", 400),(1, "2021-01-01 20:00:00", 5500),(5, "2021-08-06 12:00:00", 1000);
Query OK, 5 rows affected (0.17 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from traffic_stats;
+------+---------------------+----------------+
| id   | timestamp           | traffic_volume |
+------+---------------------+----------------+
|    1 | 2022-01-01 12:00:00 |           1000 |
|    2 | 2022-01-03 10:00:00 |           3000 |
|    3 | 2022-05-01 15:55:00 |            400 |
|    1 | 2021-01-01 20:00:00 |           5500 |
|    5 | 2021-08-06 12:00:00 |           1000 |
+------+---------------------+----------------+
5 rows in set (0.00 sec)
```

-To list the tables available in database ‘Network_Traffic’:



```
mysql> show tables;
+---------------------------+
| Tables_in_network_traffic |
+---------------------------+
| ip_addresses              |
| network_devices           |
| protocols                 |
| traffic                   |
| traffic_stats             |
+---------------------------+
5 rows in set (0.11 sec)

mysql>
```

System_Resources Database:

This database monitors system resource usage, which is important in assessing how a system responds to increased load from DoS attacks.

Using **third database**, named as **‘System_Resources’**:

Resource_usage Table:

Logs CPU, memory, and disk usage over time. Monitoring resource usage helps detect when system resources are being overwhelmed by an attack.

- Creating first table, named as ‘Resource_usage’ and displaying it:

```
mysql> use system_resources;
Database changed
mysql> create table resource_usage (id integer(20), timestamp datetime, cpu_usage integer(10), memory_usage interger(10), disk_usage integer(10));
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 'i
nterger(10), disk_usage integer(10))' at line 1
mysql> create table resource_usage (id integer(20), timestamp datetime, cpu_usage integer(10), memory_usage integer(10), disk_usage integer(10));
Query OK, 0 rows affected, 4 warnings (0.47 sec)

mysql> insert into resource_usage values (1, "2022-09-17 10:15:00", 45, 60, 70),(2, "2022-11-01 14:00:00", 40, 62,65),(3
, "2022-04-09 19:05:00", 55, 48, 77),(4, "2021-05-01 12:10:000", 32, 60, 66),(5, "2021-01-10 10:30:00", 61, 52, 75);
Query OK, 5 rows affected (0.22 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from resource_usage;
+------+---------------------+-----------+--------------+------------+
| id   | timestamp           | cpu_usage | memory_usage | disk_usage |
+------+---------------------+-----------+--------------+------------+
|    1 | 2022-09-17 10:15:00 |        45 |           60 |         70 |
|    2 | 2022-11-01 14:00:00 |        40 |           62 |         65 |
|    3 | 2022-04-09 19:05:00 |        55 |           48 |         77 |
|    4 | 2021-05-01 12:10:00 |        32 |           60 |         66 |
|    5 | 2021-01-10 10:30:00 |        61 |           52 |         75 |
+------+---------------------+-----------+--------------+------------+
5 rows in set (0.00 sec)
```

Resources Table:

Stores metadata about the system resources, providing information on the types of resources being monitored.

-Creating second table, named as ‘Resourses’ and displaying it:

```
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)
```

System_Stats Table:

Logs system load and uptime, helping administrators assess the system's overall health and resilience to attacks.

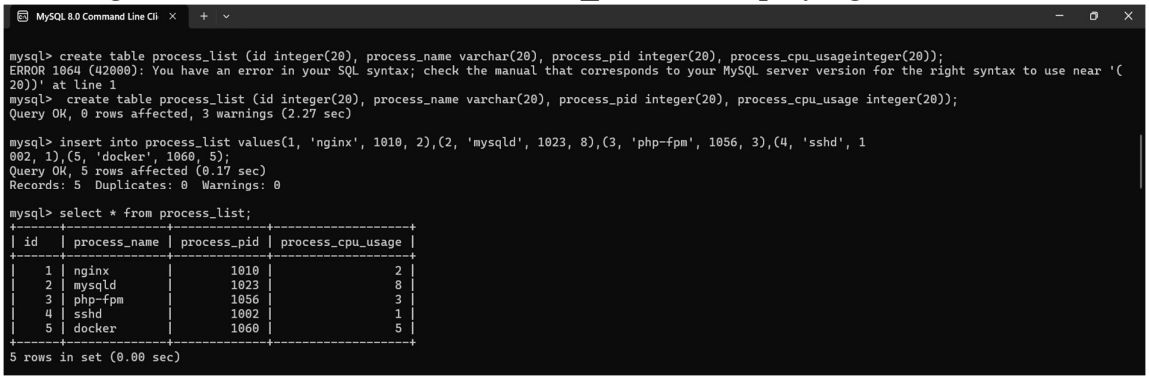
- Creating third table, named as ‘System_stats’ and displaying it:

```
mysql> create table system_stats (id integer(20), timestamp datetime, system_load integer(10), system_uptime time);
Query OK, 0 rows affected, 2 warnings (0.60 sec)

mysql> insert into system_stats values(1,"2022-01-01 10:00:00",2,"16:00:00");
Query OK, 1 row affected (0.07 sec)

mysql> insert into system_stats values(2,"2022-09-01 12:00:00",1,"10:00:00"),(3,"2021-01-01 16:00:00",3,"07:30:00"),(4,"
2021-06-05 10:00:00",1,"13:00:00"),(5,"2021-05-01 08:15:00",2,"18:00:00");
Query OK, 4 rows affected (0.11 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from system_stats;
+------+---------------------+-------------+---------------+
| id   | timestamp           | system_load | system_uptime |
+------+---------------------+-------------+---------------+
|    1 | 2022-01-01 10:00:00 |           2 | 16:00:00      |
|    2 | 2022-09-01 12:00:00 |           1 | 10:00:00      |
|    3 | 2021-01-01 16:00:00 |           3 | 07:30:00      |
|    4 | 2021-06-05 10:00:00 |           1 | 13:00:00      |
|    5 | 2021-05-01 08:15:00 |           2 | 18:00:00      |
+------+---------------------+-------------+---------------+
5 rows in set (0.00 sec)

mysql>
```

Process_list Table:

Contains a list of running processes, including their CPU usage, which helps identify resource-hogging processes that may be part of the attack.

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



User_sessions Table:

Tracks user session data (start and end times), which helps identify unusual login patterns during attacks.

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

To list the tables available in database ‘System_resources’:

```
mysql> show tables;
+----------------------------+
| Tables_in_system_resources |
+----------------------------+
| process_list               |
| resource_usage             |
| resources                  |
| system_stats               |
| user_sessions              |
+----------------------------+
5 rows in set (0.02 sec)

mysql>
```

Incident_Response Database:

This database is focused on handling incidents, documenting responses, and managing teams involved in responding to DoS attacks.

Using **fourth database**, named as **'Incident_Response'**:

Incident_Types Table:

Defines different types of incidents (e.g., DoS, data breach), helping categorize and manage responses.

- Creating first table, named as 'Incident_types' and displaying it:

```
mysql>  insert into incident_types values(1, 'Volume-Based Attacks', 'Denial-of-Service attacks that flood the target with high volumes of traffic'),(2, 'Pr
otocol Attacks', 'DoS attacks that exploit weaknesses in the protocol stack'),(3,'Application Layer Attacks', 'Denial-of-Service attacks targeting the appli
cation layer'),(4, 'Resource Exhaustion', 'Attacks that exhaust the resources of the target system'),(5, 'DDOS', 'A DDoS attack involves multiple systems at
tacking the
target at the same time');
Query OK, 5 rows affected (0.20 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> select * from incident_types;
+------+--------------------------+-------------------------------------------------------------------------------+
| id   | type_name                | type_description                                                              |
+------+--------------------------+-------------------------------------------------------------------------------+
|    1 | Volume-Based Attacks     | Denial-of-Service attacks that flood the target with high volumes of traffic  |
|    2 | Protocol Attacks         | DoS attacks that exploit weaknesses in the protocol stack                     |
|    3 | Application Layer Attacks | Denial-of-Service attacks targeting the application layer                     |
|    4 | Resource Exhaustion      | Attacks that exhaust the resources of the target system                       |
|    5 | DDOS                     | A DDoS attack involves multiple systems attacking the target at the same time |
+------+--------------------------+-------------------------------------------------------------------------------+
5 rows in set (0.00 sec)
```

Response_Plans Table:

Stores predefined response plans for handling different types of incidents, allowing the organization to respond quickly and effectively.

-Creating second table, named as 'Response_plans' and displaying it:

```
mysql>  insert into response_plans values(1, 'Volume-Based Attack Mitigation Plan', 'Plan to mitigate volume-based DoS')
,(2, 'Protocol Attack Response Plan', 'mitigate protocol-based DoS attacks'),(3, 'Application Layer Attack  Mitigation P
lan', 'Plan to handle application-layer attacks'),(4, 'Resource Exhaustion Prevention Plan', 'Plan to prevent resource e
xhaustion attacks'),(5, 'DDoS', 'Plan to mitigate DDoS');
Query OK, 5 rows affected (0.09 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> select * from response_plans;
+------+---------------------------------------+-----------------------------------------+
| id   | plan_name                             | plan_description                        |
+------+---------------------------------------+-----------------------------------------+
|    1 | Volume-Based Attack Mitigation Plan   | Plan to mitigate volume-based DoS       |
|    2 | Protocol Attack Response Plan         | mitigate protocol-based DoS attacks     |
|    3 | Application Layer Attack  Mitigation Plan | Plan to handle application-layer attacks |
|    4 | Resource Exhaustion Prevention Plan   | Plan to prevent resource exhaustion attacks |
|    5 | DDoS                                  | Plan to mitigate DDoS                   |
+------+---------------------------------------+-----------------------------------------+
5 rows in set (0.00 sec)
```

Response_Teams Table:

Logs information about response teams, including team names and leads. This helps coordinate incident response activities.

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

```
mysql> create table response_teams (id integer(20), team_name varchar(50), team_lead varchar(50));
Query OK, 0 rows affected, 1 warning (0.66 sec)

mysql> insert into response_teams values (1, 'Volume-Based Attack', 'Alice Johnson'),(2, 'Protocol Attack', 'Bob Smith')
,(3, 'Application Layer Attack', 'Carol Davis'),(4, 'Resource Exhaustion Defense', 'David Clark'),(5, 'DDoS', 'Eve Thomp
son');
Query OK, 5 rows affected (0.22 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from response_teams;
+------+-----------------------------+---------------+
| id   | team_name                   | team_lead     |
+------+-----------------------------+---------------+
|    1 | Volume-Based Attack         | Alice Johnson |
|    2 | Protocol Attack             | Bob Smith     |
|    3 | Application Layer Attack    | Carol Davis   |
|    4 | Resource Exhaustion Defense | David Clark   |
|    5 | DDoS                        | Eve Thompson  |
+------+-----------------------------+---------------+
5 rows in set (0.00 sec)
```

-

Incident_report Table:

Contains reports generated after an incident is handled, including details of how the attack was mitigated. This is useful for post-incident analysis.

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

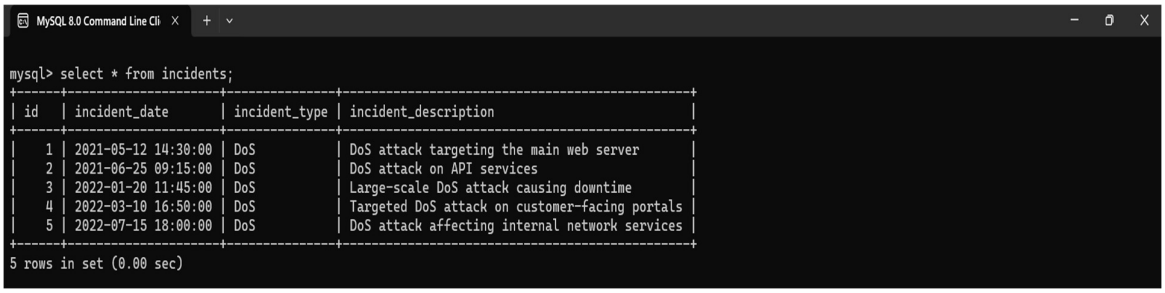
```
MySQL 8.0 Command Line Cli

mysql> insert into incident_reports values (1, 101, '2022-01-10 10:30:00', 'Report on Volume-Based attack affecting network availability'),(2, 102, '2022-02
-15 11:45:00', 'Protocol attack mitigated successfully by adjusting SYN timeouts'),(
3, 103, '2022-03-20 09:25:00', 'Application layer attack detected and mitigated using WAF'),(4, 104, '2021-04-25 14:15:0
0', 'Resource exhaustion prevented by upgrading server capacity'),(5, 105, '2021-01-01 16:40:00', 'DDoS attack handled u
sing traffic rerouting techniques');
Query OK, 5 rows affected (0.15 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from incident_reports;
+------+-------------+---------------------+-----------------------------------------------------------------+
| id   | incident_id | report_date         | report_description                                              |
+------+-------------+---------------------+-----------------------------------------------------------------+
|    1 |         101 | 2022-01-10 10:30:00 | Report on Volume-Based attack affecting network availability    |
|    2 |         102 | 2022-02-15 11:45:00 | Protocol attack mitigated successfully by adjusting SYN timeouts |
|    3 |         103 | 2022-03-20 09:25:00 | Application layer attack detected and mitigated using WAF       |
|    4 |         104 | 2021-04-25 14:15:00 | Resource exhaustion prevented by upgrading server capacity      |
|    5 |         105 | 2021-01-01 16:40:00 | DDoS attack handled using traffic rerouting techniques          |
+------+-------------+---------------------+-----------------------------------------------------------------+
5 rows in set (0.00 sec)
```

-

Incidents Table:

Logs incidents related to DoS attacks, including the date, type, and description. This is important for tracking and responding to different security events.

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



```
MySQL 8.0 Command Line Cli

mysql> select * from incidents;
+------+---------------------+---------------+-----------------------------------------------+
| id   | incident_date       | incident_type | incident_description                          |
+------+---------------------+---------------+-----------------------------------------------+
|    1 | 2021-05-12 14:30:00 | DoS           | DoS attack targeting the main web server      |
|    2 | 2021-06-25 09:15:00 | DoS           | DoS attack on API services                    |
|    3 | 2022-01-20 11:45:00 | DoS           | Large-scale DoS attack causing downtime       |
|    4 | 2022-03-10 16:50:00 | DoS           | Targeted DoS attack on customer-facing portals |
|    5 | 2022-07-15 18:00:00 | DoS           | DoS attack affecting internal network services |
+------+---------------------+---------------+-----------------------------------------------+
5 rows in set (0.00 sec)
```

To list the tables available in database ‘Incident_response’:

```
mysql> show tables;
+------------------------------+
| Tables_in_incident_response |
+------------------------------+
| incident_reports             |
| incident_types               |
| incidents                    |
| response_plans               |
| response_teams               |
+------------------------------+
5 rows in set (0.00 sec)

mysql>
```

Security_Information Database:

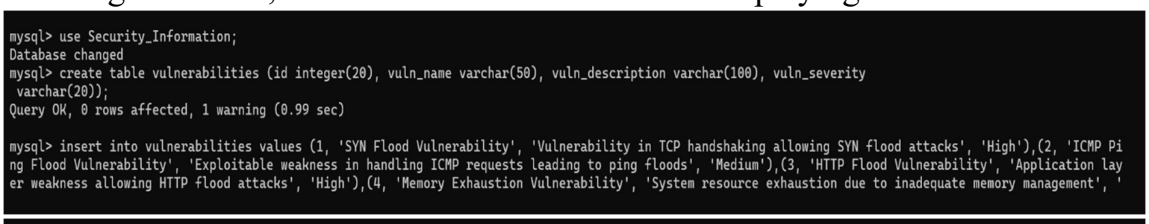
This database stores information about vulnerabilities, patches, and threat intelligence to enhance the organization’s security posture against DoS attacks.

Using **fifth database**, named as **‘Security_Information’**:

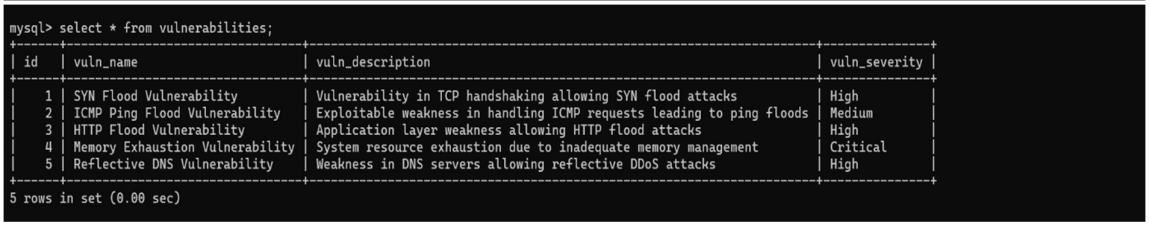
Vulnerabilities Table:

Logs system vulnerabilities, which is essential for understanding weaknesses that attackers might exploit.

- Creating first table, named as ‘vulnerabilities’ and displaying it:



```
mysql> use Security_Information;
Database changed
mysql> create table vulnerabilities (id integer(20), vuln_name varchar(50), vuln_description varchar(100), vuln_severity
 varchar(20));
Query OK, 0 rows affected, 1 warning (0.99 sec)

mysql> insert into vulnerabilities values (1, 'SYN Flood Vulnerability', 'Vulnerability in TCP handshaking allowing SYN flood attacks', 'High'),(2, 'ICMP Pi
ng Flood Vulnerability', 'Exploitable weakness in handling ICMP requests leading to ping floods', 'Medium'),(3, 'HTTP Flood Vulnerability', 'Application lay
er weakness allowing HTTP flood attacks', 'High'),(4, 'Memory Exhaustion Vulnerability', 'System resource exhaustion due to inadequate memory management', '
```



```
mysql> select * from vulnerabilities;
+------+-------------------------------+-----------------------------------------------------------------------+---------------+
| id   | vuln_name                     | vuln_description                                                      | vuln_severity |
+------+-------------------------------+-----------------------------------------------------------------------+---------------+
|    1 | SYN Flood Vulnerability       | Vulnerability in TCP handshaking allowing SYN flood attacks           | High          |
|    2 | ICMP Ping Flood Vulnerability | Exploitable weakness in handling ICMP requests leading to ping floods | Medium        |
|    3 | HTTP Flood Vulnerability      | Application layer weakness allowing HTTP flood attacks                | High          |
|    4 | Memory Exhaustion Vulnerability | System resource exhaustion due to inadequate memory management      | Critical      |
|    5 | Reflective DNS Vulnerability  | Weakness in DNS servers allowing reflective DDoS attacks              | High          |
+------+-------------------------------+-----------------------------------------------------------------------+---------------+
5 rows in set (0.00 sec)
```

Patches Table:

Stores information about security patches that address vulnerabilities, helping ensure the system is kept up-to-date and secure.

-Creating second table, named as ‘patches’ and displaying it:

```
mysql> create table patches (id integer(20), patch_name varchar(50), patch_description varchar(100), patch_release_date datetime);
Query OK, 0 rows affected, 1 warning (1.11 sec)

mysql> insert into patches values(1, 'SYN Flood Protection Patch', 'Patch to secure the system from SYN flood attacks', '2022-06-15 10:00:00'),(2, 'ICMP Flo
od Rate-Limiting Patch', 'Patch to limit ICMP request rates', '2022-07-10 12:00:00')
,(3, 'WAF Patch for HTTP Flood', 'Web application firewall (WAF) enhancement to block HTTP flood attacks', '2022-08-01 10:00:00'),(4, 'Memory Management Pat
ch', 'Patch to optimize memory management and prevent resource exhaustion', '2021-09
-05 09:00:00'),(5, 'DNS Reflection Patch', 'Patch to prevent DNS servers from participating in reflective attacks', '2021-01-01 10:00:00');
Query OK, 5 rows affected (0.09 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from patches;
+------+------------------------------+-----------------------------------------------------------------------+---------------------+
| id   | patch_name                   | patch_description                                                     | patch_release_date  |
+------+------------------------------+-----------------------------------------------------------------------+---------------------+
|    1 | SYN Flood Protection Patch   | Patch to secure the system from SYN flood attacks                     | 2022-06-15 10:00:00 |
|    2 | ICMP Flood Rate-Limiting Patch | Patch to limit ICMP request rates                                   | 2022-07-10 12:00:00 |
|    3 | WAF Patch for HTTP Flood     | Web application firewall (WAF) enhancement to block HTTP flood attacks | 2022-08-01 10:00:00 |
|    4 | Memory Management Patch      | Patch to optimize memory management and prevent resource exhaustion   | 2021-09-05 09:00:00 |
|    5 | DNS Reflection Patch         | Patch to prevent DNS servers from participating in reflective attacks | 2021-01-01 10:00:00 |
+------+------------------------------+-----------------------------------------------------------------------+---------------------+
```

Security_Advisories Table:

Logs security advisories issued by vendors or security organizations, providing crucial information on emerging threats.

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

```
mysql> create table security_advisories (id integer(20), advisory_name varchar(50), advisory_description varchar(100));
Query OK, 0 rows affected, 1 warning (0.70 sec)

mysql> insert into security_advisories values(1, 'SYN Flood Advisory', 'Advisory about recent SYN flood attack vulnerabilities and patches.'),(2, 'ICMP Floo
d Advisory', 'Advisory on the potential for ICMP ping flood attacks.'),(3, 'HTTP Flood Advisory', 'Advisory on application layer HTTP flood attack preventio
n measures.'),(4, 'Resource Exhaustion Advisory', 'Advisory regarding system capacity and defense against resource exhaustion.'),(5, 'DNS Reflection Advisor
y', 'Advisory on mitigating reflective DNS attacks.');
Query OK, 5 rows affected (0.12 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from security_advisories;
+------+------------------------------+-----------------------------------------------------------------------------+
| id   | advisory_name                | advisory_description                                                        |
+------+------------------------------+-----------------------------------------------------------------------------+
|    1 | SYN Flood Advisory           | Advisory about recent SYN flood attack vulnerabilities and patches.         |
|    2 | ICMP Flood Advisory          | Advisory on the potential for ICMP ping flood attacks.                      |
|    3 | HTTP Flood Advisory          | Advisory on application layer HTTP flood attack prevention measures.        |
|    4 | Resource Exhaustion Advisory | Advisory regarding system capacity and defense against resource exhaustion. |
|    5 | DNS Reflection Advisory      | Advisory on mitigating reflective DNS attacks.                              |
+------+------------------------------+-----------------------------------------------------------------------------+
5 rows in set (0.00 sec)
```

Threat_Intelligence Table:

Contains information about known threats and their severity, helping in proactive detection and defense against attacks.

-Creating fourth table, named as ‘threat_intelligence’ and displaying it:

```
mysql> insert into threat_intelligence values(1, 'SYN Flood Threat', 'Increased activity of SYN flood attacks targeting vulnerable servers.', 'High'),(2, 'I
CMP Flood Threat', 'Detecting an uptick in ICMP ping flood attacks across networks.', 'Medium'),(3, 'HTTP Flood Threat', 'Increase in HTTP flood attacks tar
geting application layer services.', 'High'),(4, 'Resource Exhaustion Threat', 'Threat of resource exhaustion due to unoptimized memory usage.', 'Critical')
,(5, 'Reflective DNS Threat', 'High activity of reflective DNS attacks impacting multiple organizations.', 'High');
Query OK, 5 rows affected (0.47 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from threat_intelligence;
+------+---------------------------+-----------------------------------------------------------------------+--------------+
| id   | threat_name               | threat_description                                                    | threat_level |
+------+---------------------------+-----------------------------------------------------------------------+--------------+
|    1 | SYN Flood Threat          | Increased activity of SYN flood attacks targeting vulnerable servers. | High         |
|    2 | ICMP Flood Threat         | Detecting an uptick in ICMP ping flood attacks across networks.       | Medium       |
|    3 | HTTP Flood Threat         | Increase in HTTP flood attacks targeting application layer services.  | High         |
|    4 | Resource Exhaustion Threat | Threat of resource exhaustion due to unoptimized memory usage.       | Critical     |
|    5 | Reflective DNS Threat     | High activity of reflective DNS attacks impacting multiple organizations. | High         |
+------+---------------------------+-----------------------------------------------------------------------+--------------+
5 rows in set (0.00 sec)
```

Security_Incidents Table:

Logs any security-related incidents (DoS or otherwise), helping build a historical record of incidents for analysis.

-Creating fifth table, named as ‘security_incidents’ and displaying it:

```
mysql> create table security_incidents (id integer(20), incident_id integer(20), security_incident_date datetime);
Query OK, 0 rows affected, 2 warnings (0.67 sec)

mysql> insert into security_incidents values(1, 101, '2022-01-01 10:00:00'),(2, 102, '2022-02-16 10:00:00'),(3, 103, '2022-03-21 08:50:00'),(4, 104, '2021-0
9-05 09:00:00'),(5, 105, '2021-01-01 10:00:00');
Query OK, 5 rows affected (0.21 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from security_incidents;
+------+-------------+-----------------------+
| id   | incident_id | security_incident_date |
+------+-------------+-----------------------+
|    1 |         101 | 2022-01-01 10:00:00    |
|    2 |         102 | 2022-02-16 10:00:00    |
|    3 |         103 | 2022-03-21 08:50:00    |
|    4 |         104 | 2021-09-05 09:00:00    |
|    5 |         105 | 2021-01-01 10:00:00    |
+------+-------------+-----------------------+
5 rows in set (0.00 sec)
```

To list the tables available in database ‘Security_information’:

```
mysql> show tables;
+---------------------------------+
| Tables_in_security_information |
+---------------------------------+
| patches                         |
| security_advisories             |
| security_incidents              |
| threat_intelligence             |
| vulnerabilities                 |
+---------------------------------+
5 rows in set (0.06 sec)
```

4) Queries identified by the Network Infra security team:

Retrieve all attack where attack_type =1:

```
mysql> SELECT * FROM attacks WHERE attack_type = 1;
+------+-------------+---------------------+---------------+
| id   | attack_type | attack_date         | source_ip     |
+------+-------------+---------------------+---------------+
|    1 |           1 | 2022-01-01 12:00:00 | 192.168.1.100 |
|    4 |           1 | 2022-01-04 15:00:00 | 192.168.1.103 |
+------+-------------+---------------------+---------------+
2 rows in set (0.00 sec)
```

Retrieve all attacks detail whose attack date is between 2022-01-01 and 2022-01-31:

```
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 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 (0.02 sec)
```

Retrieve all data from attacks where source_ip =192.168.1.100:

```
mysql> SELECT * FROM attacks WHERE source_ip = '192.168.1.100';
+------+-------------+---------------------+---------------+
| id   | attack_type | attack_date         | source_ip     |
+------+-------------+---------------------+---------------+
|    1 |           1 | 2022-01-01 12:00:00 | 192.168.1.100 |
+------+-------------+---------------------+---------------+
1 row in set (0.00 sec)
```

Retrieve data from attack_types where type_name = DDoS:

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

Retrieve data from attack_types where description has word flooding work in it:

```
mysql> SELECT * FROM attack_types WHERE description LIKE '%flooding%';
Empty set (0.00 sec)
```

Retrieve data from sources where source ip is 192.168.1.100:

```
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 (0.14 sec)
```

Retrieve data from sources where source country is USA:

```
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 data from detection_rules where rule_name is 1:

```
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 (0.12 sec)
```

Retrieve data from detection_rules where rule_description is DDoS:

```
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)
```

Retrieve all data from alerts where alert_level is high:

```
mysql> SELECT * FROM alerts WHERE alert_level = 'High';
+------+-----------+---------------------+-------------+
| id   | attack_id | alert_date          | alert_level |
+------+-----------+---------------------+-------------+
|    1 |         1 | 2022-01-01 12:00:00 | High        |
|    4 |         4 | 2022-01-04 15:00:00 | High        |
+------+-----------+---------------------+-------------+
2 rows in set (0.00 sec)
```

Retrieve all data from alerts where date is between 2022-01-01 and 2022-01-31:

```
mysql> SELECT * FROM alerts WHERE alert_date BETWEEN '2022-01-01' AND '2022-01-31';
+------+-----------+---------------------+-------------+
| 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)
```

5) 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 DoS. By integrating real-time alerts, detection rules, and advanced analytics, the system aims to enhance overall cybersecurity and protect critical assets from threats.