PROJECT

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

Prepared By: **Guided By:**

Dande Balaji **Zakir Hussain**

TABLE OF CONTENTS:

1. DDoS Attacks Description:

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

Here's how a DDoS attack works:

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

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

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

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

Types of DDoS attacks:

1) **Volume-based attacks:**

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

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

2) **Protocol attacks:**

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

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

3) **Application-layer attacks:**

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

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

DDoS attacks can be launched using various techniques, including:

1) **Botnets:**

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

2) **Malware:**

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

3) **Scripting:**

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

4) **Amplification Attacks:**

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

To protect against DDoS attacks, organizations can use:

1) **Firewalls:**

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

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

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

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

3) **Load Balancing:**

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

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

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

5) **DDoS Mitigation Services:**

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

2) Databases used in this Project:

- Created five databases:

Syntax: create database [name of database];

DATABASES:

Database 1: Attack_Detection

This database is likely used to store information related to cyber attacks detected in a system.

Database 2: Network_Traffic

This database records the flow of data across the network. It typically stores logs of incoming and outgoing traffic, packet information, IP addresses, ports, and protocols.

Database 3: System_Logging

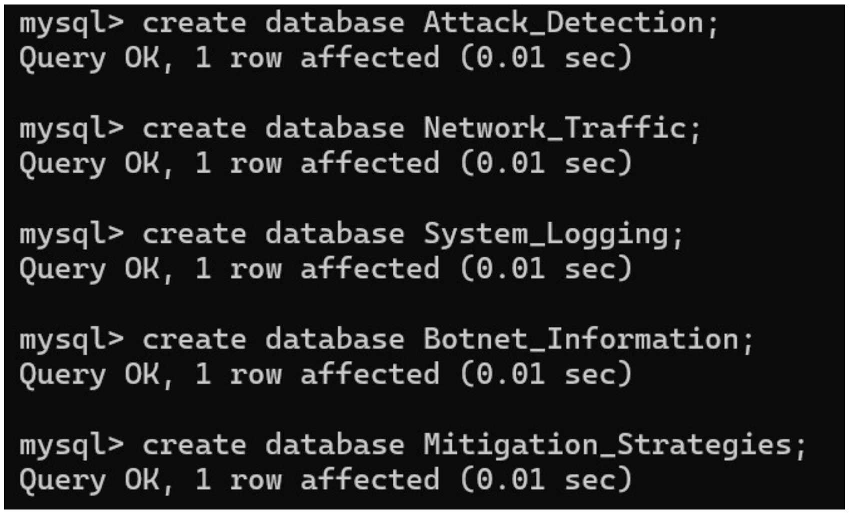
This database stores logs generated by different system components. It collects information from operating systems, applications, and security tools, providing insights into system behavior, errors, and possible malicious activity.

Database 4: Botnet_Information

This database is likely used to store information related to botnets—networks of compromised systems controlled remotely by attackers.

Database 5: Mitigation_Strategies

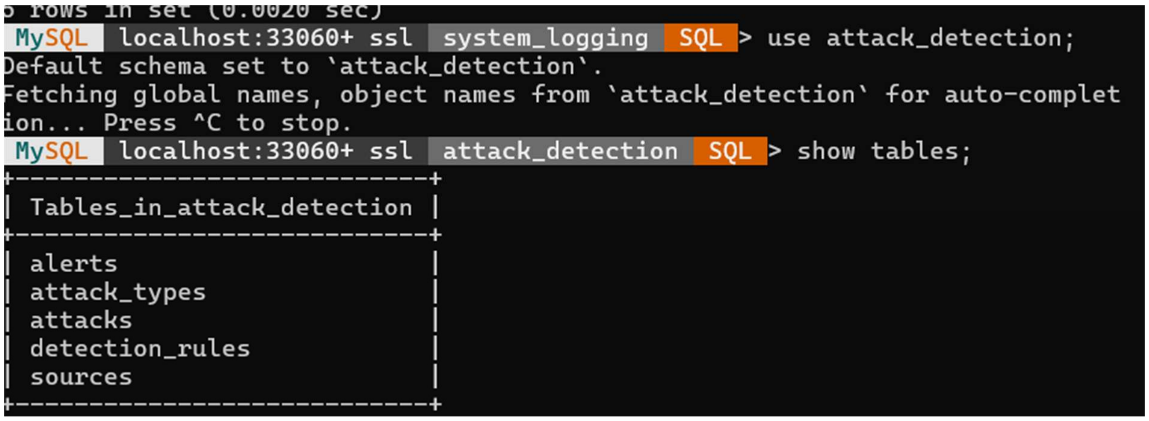
This database stores information on various methods used to prevent or minimize the impact of cyber attacks.



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

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

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

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

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

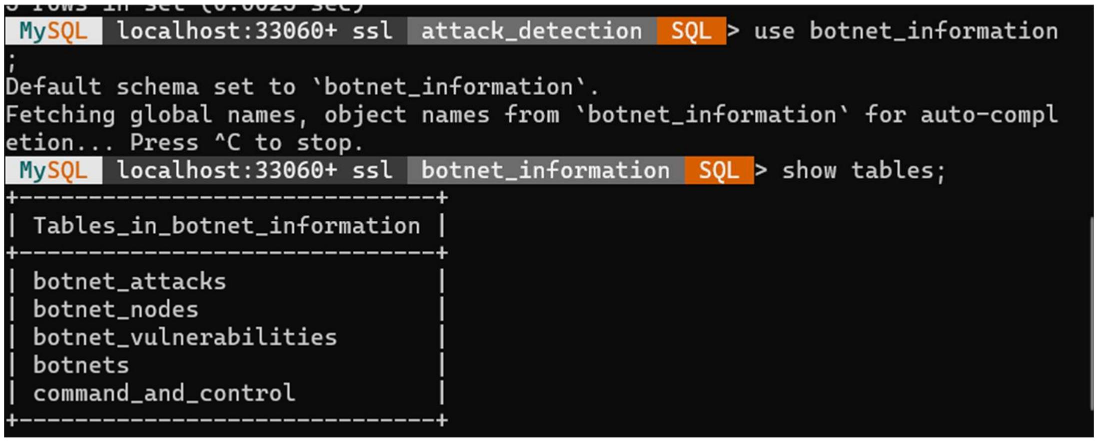
3) Tables used in each of the Databases:

i. Tables used in Attack_Detection database:



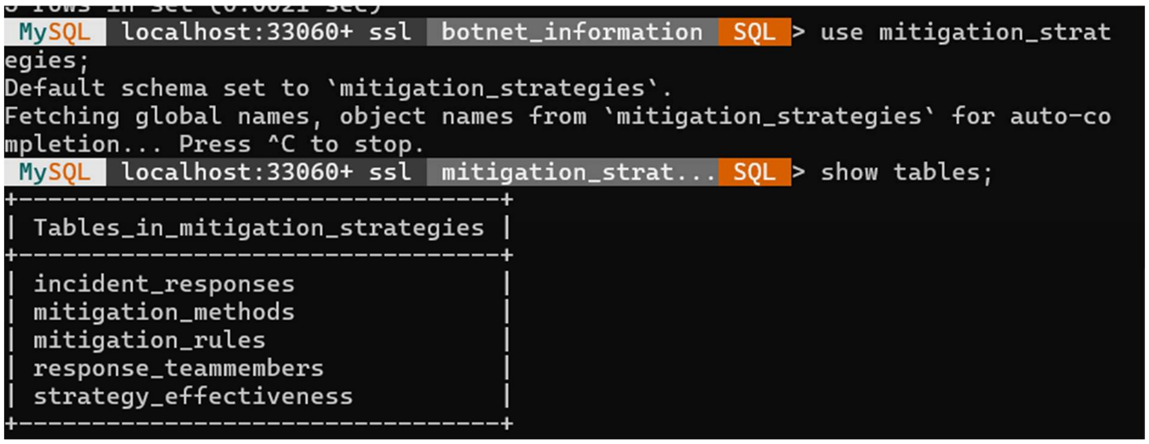
```
5 rows in set (0.0020 sec)
 MySQL  localhost:33060+ ssl  system_logging  SQL > use attack_detection;
Default schema set to `attack_detection`.
Fetching global names, object names from `attack_detection` for auto-complet
ion... Press ^C to stop.
 MySQL  localhost:33060+ ssl  attack_detection  SQL > show tables;
+-----------------------------+
| Tables_in_attack_detection |
+-----------------------------+
| alerts                      |
| attack_types                |
| attacks                     |
| detection_rules             |
| sources                     |
+-----------------------------+
```

ii. Tables used in Network_Traffic database:



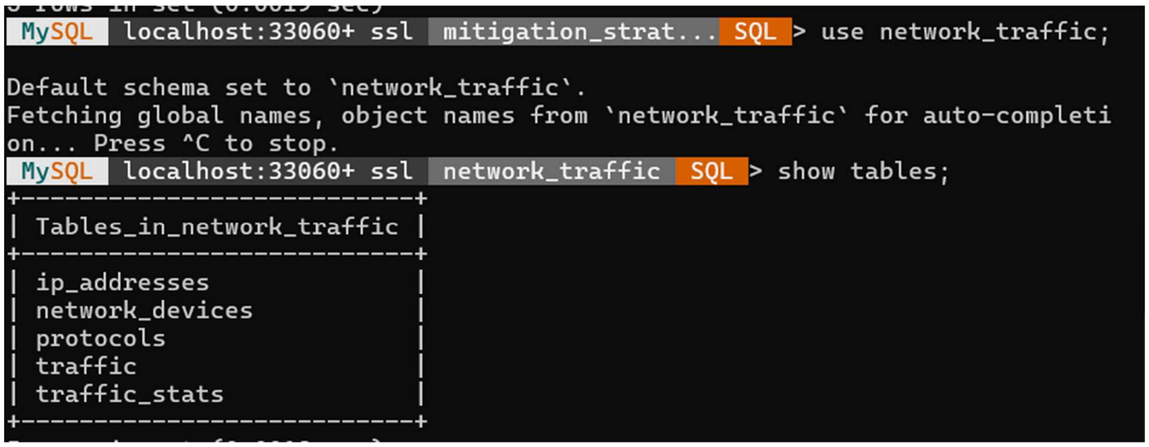
```
5 rows in set (0.0025 sec)
 MySQL  localhost:33060+ ssl  attack_detection  SQL > use botnet_information
;
Default schema set to `botnet_information`.
Fetching global names, object names from `botnet_information` for auto-compl
etion... Press ^C to stop.
 MySQL  localhost:33060+ ssl  botnet_information  SQL > show tables;
+------------------------------+
| Tables_in_botnet_information |
+------------------------------+
| botnet_attacks               |
| botnet_nodes                 |
| botnet_vulnerabilities       |
| botnets                      |
| command_and_control          |
+------------------------------+
```

iii. Tables used in System_Logging database:



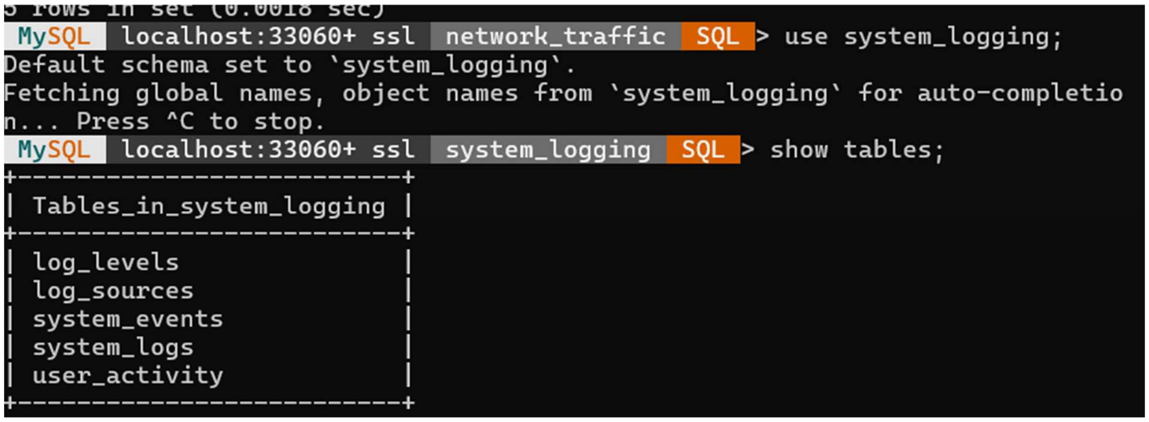
```
5 rows in set (0.0021 sec)
 MySQL  localhost:33060+ ssl  botnet_information  SQL > use mitigation_strat
egies;
Default schema set to `mitigation_strategies`.
Fetching global names, object names from `mitigation_strategies` for auto-co
mpletion... Press ^C to stop.
 MySQL  localhost:33060+ ssl  mitigation_strat...  SQL > show tables;
+---------------------------------+
| Tables_in_mitigation_strategies |
+---------------------------------+
| incident_responses              |
| mitigation_methods              |
| mitigation_rules                |
| response_teammembers            |
| strategy_effectiveness          |
+---------------------------------+
```

iv. Tables used in Botnet_Information database:



```
MySQL  localhost:33060+ ssl  mitigation_strat...  SQL > use network_traffic;

Default schema set to `network_traffic`.
Fetching global names, object names from `network_traffic` for auto-completi
on... Press ^C to stop.
MySQL  localhost:33060+ ssl  network_traffic  SQL > show tables;
+---------------------------+
| Tables_in_network_traffic |
+---------------------------+
| ip_addresses              |
| network_devices           |
| protocols                 |
| traffic                   |
| traffic_stats             |
+---------------------------+
```

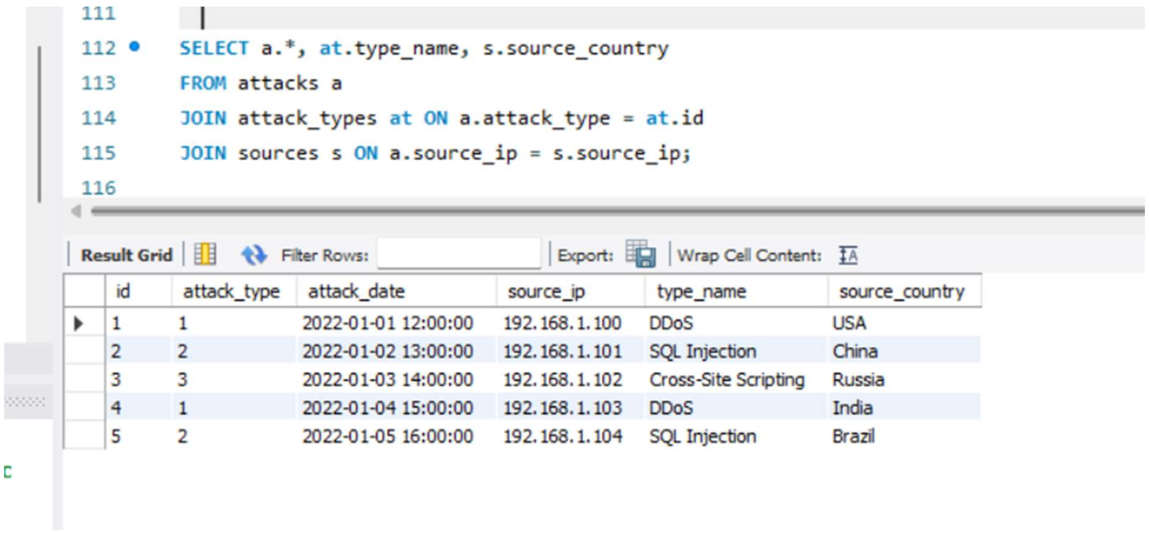
v. Tables used in Mitigation_Strategies database:



```
5 rows in set (0.0018 sec)
MySQL  localhost:33060+ ssl  network_traffic  SQL > use system_logging;
Default schema set to `system_logging`.
Fetching global names, object names from `system_logging` for auto-completio
n... Press ^C to stop.
MySQL  localhost:33060+ ssl  system_logging  SQL > show tables;
+--------------------------+
| Tables_in_system_logging |
+--------------------------+
| log_levels               |
| log_sources              |
| system_events            |
| system_logs              |
| user_activity            |
+--------------------------+
```

4) Queries identified by the Network Infra security team:

1) Retrieve all attacks with corresponding attack type and source information:

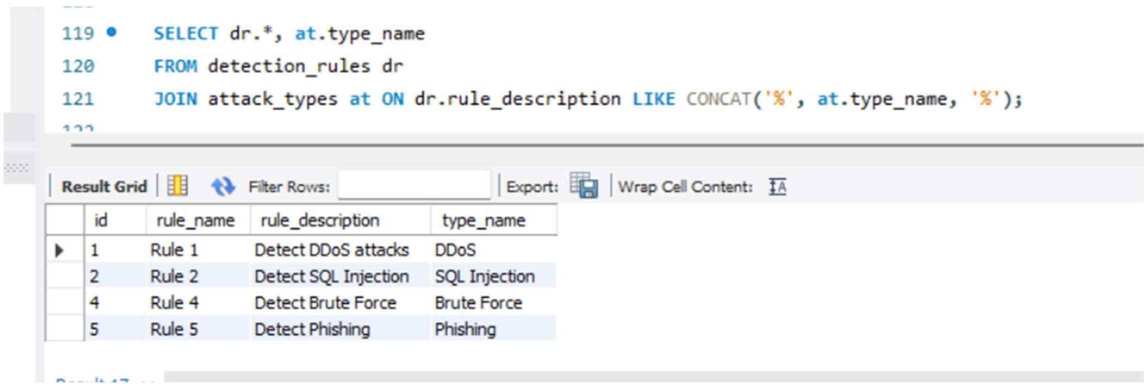


```sql
SELECT a.*, at.type_name, s.source_country
FROM attacks a
JOIN attack_types at ON a.attack_type = at.id
JOIN sources s ON a.source_ip = s.source_ip;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	attack_type	attack_date	source_ip	type_name	source_country
1	1	2022-01-01 12:00:00	192.168.1.100	DDoS	USA
2	2	2022-01-02 13:00:00	192.168.1.101	SQL Injection	China
3	3	2022-01-03 14:00:00	192.168.1.102	Cross-Site Scripting	Russia
4	1	2022-01-04 15:00:00	192.168.1.103	DDoS	India
5	2	2022-01-05 16:00:00	192.168.1.104	SQL Injection	Brazil

2) Retrieve all detection rules with corresponding attack type:

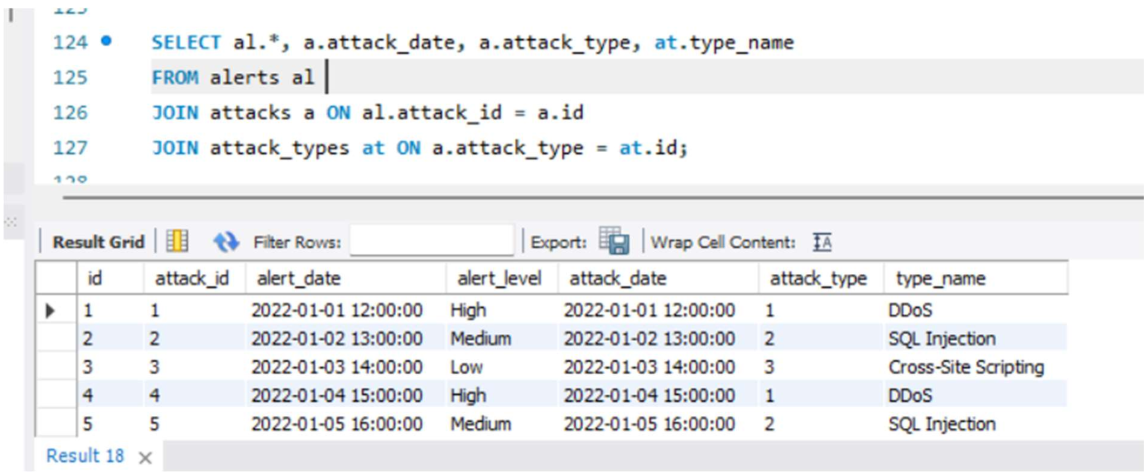


```sql
SELECT dr.*, at.type_name
FROM detection_rules dr
JOIN attack_types at ON dr.rule_description LIKE CONCAT('%', at.type_name, '%');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	rule_name	rule_description	type_name
1	Rule 1	Detect DDoS attacks	DDoS
2	Rule 2	Detect SQL Injection	SQL Injection
4	Rule 4	Detect Brute Force	Brute Force
5	Rule 5	Detect Phishing	Phishing

3) *Retrieve all alerts with corresponding attack information and alert level:*

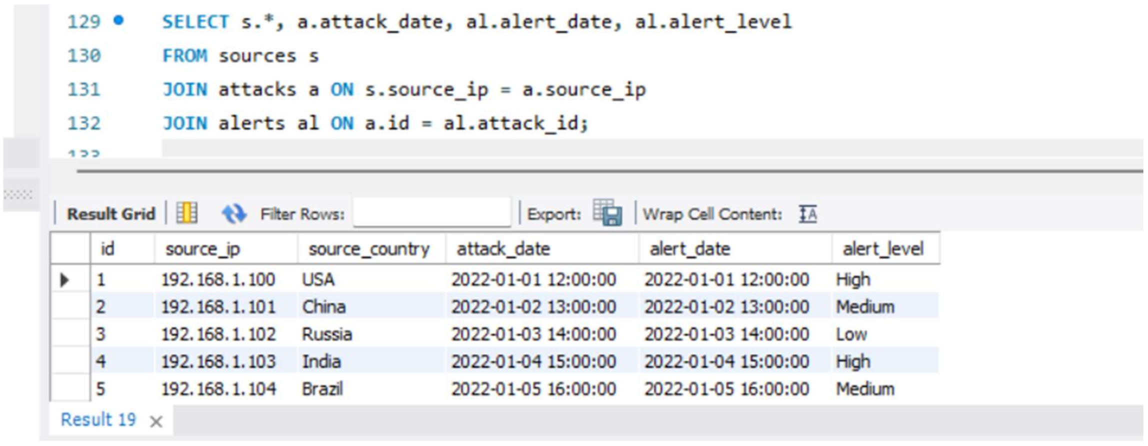


```
SELECT al.*, a.attack_date, a.attack_type, at.type_name
FROM alerts al
JOIN attacks a ON al.attack_id = a.id
JOIN attack_types at ON a.attack_type = at.id;
```

id	attack_id	alert_date	alert_level	attack_date	attack_type	type_name
1	1	2022-01-01 12:00:00	High	2022-01-01 12:00:00	1	DDoS
2	2	2022-01-02 13:00:00	Medium	2022-01-02 13:00:00	2	SQL Injection
3	3	2022-01-03 14:00:00	Low	2022-01-03 14:00:00	3	Cross-Site Scripting
4	4	2022-01-04 15:00:00	High	2022-01-04 15:00:00	1	DDoS
5	5	2022-01-05 16:00:00	Medium	2022-01-05 16:00:00	2	SQL Injection

Result 18

4) *Retrieve all sources with corresponding attack and alert information:*

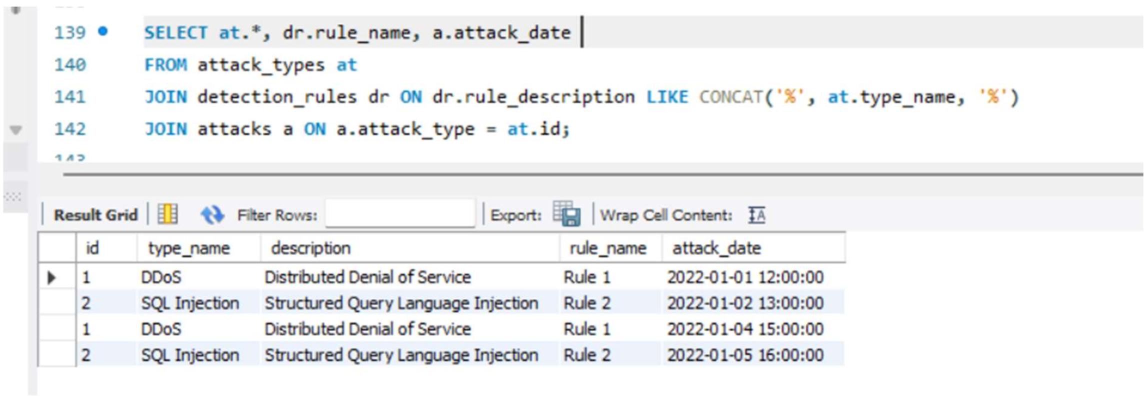


```
SELECT s.*, a.attack_date, al.alert_date, al.alert_level
FROM sources s
JOIN attacks a ON s.source_ip = a.source_ip
JOIN alerts al ON a.id = al.attack_id;
```

id	source_ip	source_country	attack_date	alert_date	alert_level
1	192.168.1.100	USA	2022-01-01 12:00:00	2022-01-01 12:00:00	High
2	192.168.1.101	China	2022-01-02 13:00:00	2022-01-02 13:00:00	Medium
3	192.168.1.102	Russia	2022-01-03 14:00:00	2022-01-03 14:00:00	Low
4	192.168.1.103	India	2022-01-04 15:00:00	2022-01-04 15:00:00	High
5	192.168.1.104	Brazil	2022-01-05 16:00:00	2022-01-05 16:00:00	Medium

Result 19

5) *Retrieve all attack types with corresponding detection rules and attacks:*



```
SELECT at.*, dr.rule_name, a.attack_date
FROM attack_types at
JOIN detection_rules dr ON dr.rule_description LIKE CONCAT('%', at.type_name, '%')
JOIN attacks a ON a.attack_type = at.id;
```

id	type_name	description	rule_name	attack_date
1	DDoS	Distributed Denial of Service	Rule 1	2022-01-01 12:00:00
2	SQL Injection	Structured Query Language Injection	Rule 2	2022-01-02 13:00:00
1	DDoS	Distributed Denial of Service	Rule 1	2022-01-04 15:00:00
2	SQL Injection	Structured Query Language Injection	Rule 2	2022-01-05 16:00:00

5. Final Goal of the Project:

The ultimate goal of the project is to develop a robust, scalable, and secure system capable of detecting, monitoring, and mitigating various types of cyber-attacks, such as DDoS attacks. By integrating real-time alerts, sophisticated detection rules, and advanced analytics, the system aims to significantly enhance overall cybersecurity and safeguard critical assets from emerging threats.