



PROJECT0:

Enhancing Network Security with MySQL: Strategies for Denial of Service (DoS) Attack Mitigation

17.09.2024

Prepared By:

K. Dushyant Reddy



Guided By:

Zakir Hussain



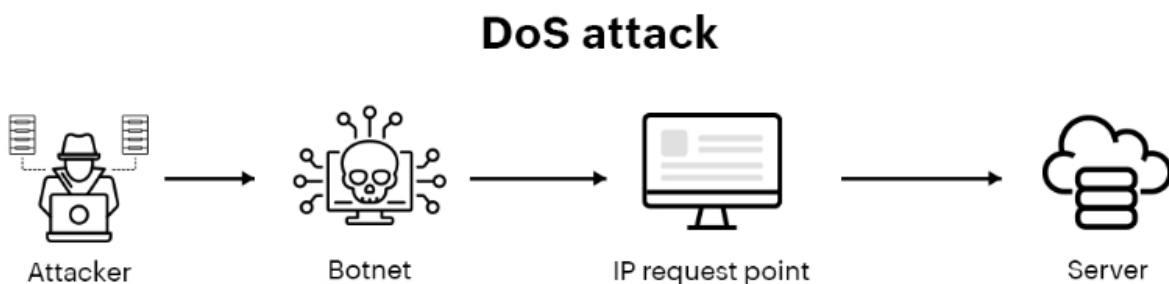
INDEX

S.NO.	TABLE OF CONTENTS	PAGES
1.	Unveiling DoS Attacks: An In-Depth Description	03 - 10
2.	Databases in Focus: Tools Employed in This Project	11 - 12
3.	Inside the Data: Tables and Structures Across Databases	13 - 28
4.	Critical Queries: Insights from the Network Security Team	29 - 31
5.	Project Output: Defining Our Ultimate Objective	32 - 33
6.	Conclusion	34

CHAPTER1: UNVEILING DOS ATTACKS

Introduction

When an attacker prevents authorized users from accessing computer systems, networks, services, or other information technology (IT) resources, result as a **Denial-of-Service (DoS) attack**.



Lets, Picture yourself hosting a party when all of a sudden a massive crowd of people shows up. Your house is so packed with them that you can't move around or enjoy the festivities. That is similar to an assault on a website or service that causes a denial of service. Regular visitors are no longer able to use the website due to the excessive amount of malicious traffic it receives. It is comparable to having a sizable number of guests who aren't even trying to have fun interrupt your celebration.

How does DoS Attack work ?

A DoS attack's main goal is to overload the capacity of a target system in order to deny service to additional requests. By their similarities, the various DoS attack vectors can be categorized.

Attacks via DoS specifically target one or more of the OSI model's seven layers. The most popular OSI targets are Layers 3 (network), 4 (transport), 6 (presentation), and 7 (application).

Data Format	Layer	Function
Data	Application Layer	Applications access network services
Data	Presentation Layer	Encryption and Compression of data
Data	Session Layer	Connection management b/w nodes
Segment	Transport Layer	Maintains data flow during transmission
Packet	Network Layer	Determine the path for data transfer
Frame	Data Link Layer	Connect physical nodes for transfer
Bit	Physical Layer	Transfer raw bits using physical mode

7 OSI Layers

DoS attacks usually fall into one of two categories:

1. Buffer Overflow Attacks

A type of attack when a computer's memory, hard drive space, or CPU time is all consumed by a memory buffer overflow. This type of hack typically results in denial-of-service by slowing down, crashing systems, or causing other malicious server behaviors.

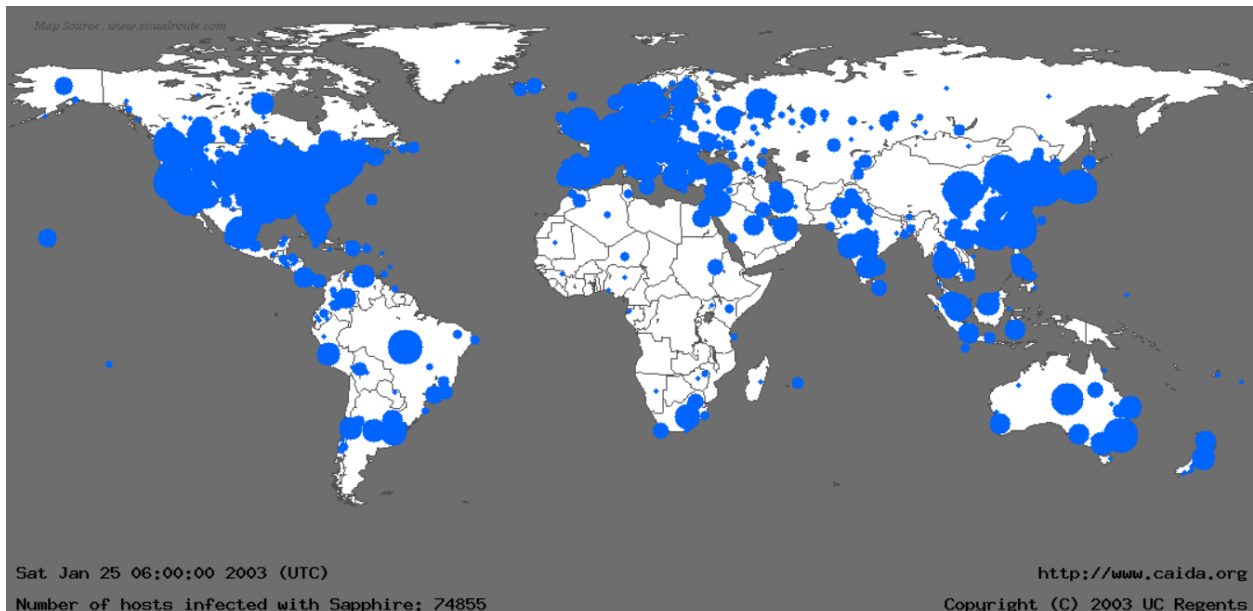
Buffer Overflow Attack Example: The 2003 Slammer Worm

What Happened: The Slammer Worm exploited a buffer overflow vulnerability in Microsoft SQL Server 2000. It sent malicious data packets that overflowed the server's memory buffer, allowing the worm to execute its code.

Impact:

- **Network Congestion:** Massive traffic caused slowdowns and outages.
- **System Crashes:** Many servers crashed, leading to a denial of service.

- **Widespread Disruption:** Affected organizations, including financial institutions and government agencies.



Key Takeaway: The Slammer Worm highlighted the importance of regular software updates and robust security practices to prevent buffer overflow attacks and protect against similar threats.

2. Flood Attacks

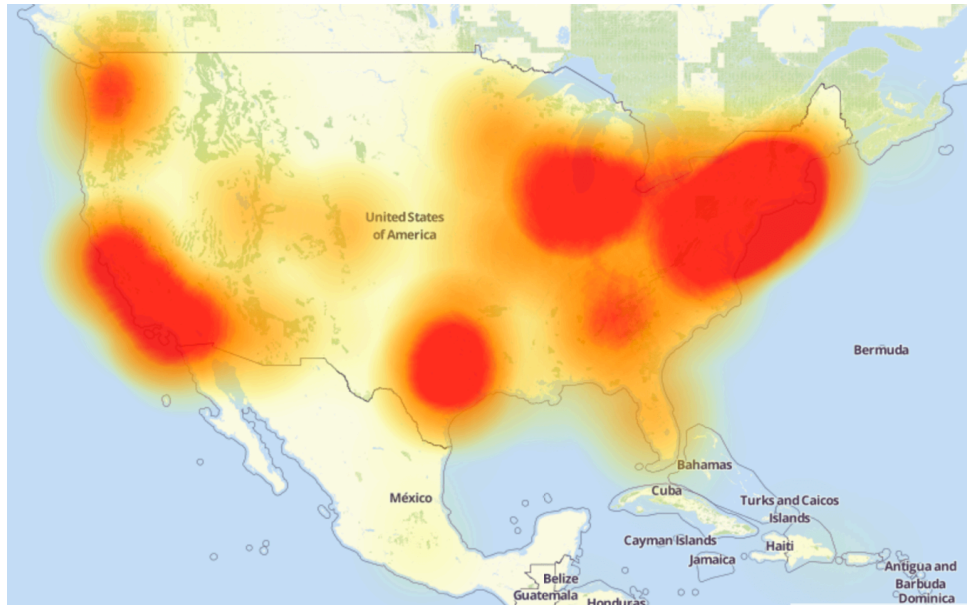
By flooding a targeted server with too many packets, a malicious attacker can overload its capacity and induce a denial-of-service. For the bad actor to be effective, the majority of DoS flood attacks require greater bandwidth than the target.

Flood Attack Example: The 2017 Dyn DNS Attack

What Happened: In October 2017, a massive Distributed Denial of Service (DDoS) flood attack targeted Dyn, a major DNS provider. Attackers used a botnet to overwhelm Dyn's servers with an enormous volume of traffic.

Impact:

- **Service Disruptions:** Major websites like Twitter and Netflix experienced outages and slowdowns.
- **Widespread Effects:** The attack highlighted vulnerabilities in internet infrastructure and affected many users globally.

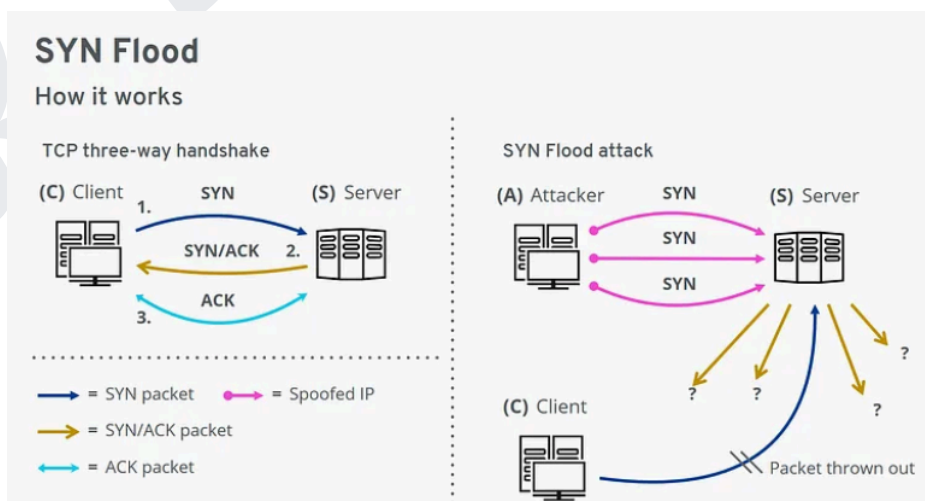


Key Takeaway: The Dyn attack demonstrated how flood attacks can cripple essential online services by overwhelming server capacity, emphasizing the need for robust DDoS protection and traffic monitoring.

Types of DoS Attacks

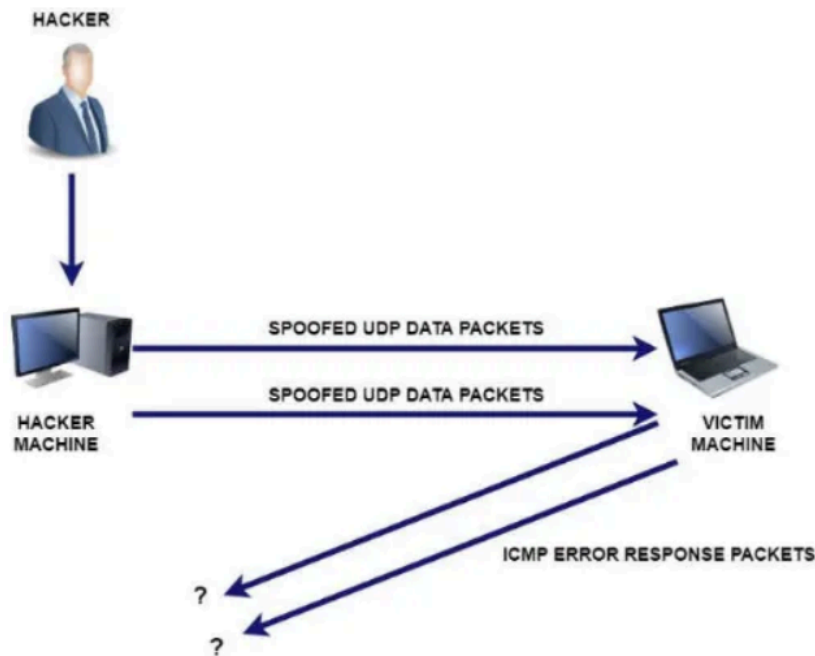
1. SYN Flood Attack

Attackers who use the SYN flood technique overwhelm a target server with numerous SYN requests, exhausting its resources.



2. UDP Flood Attack

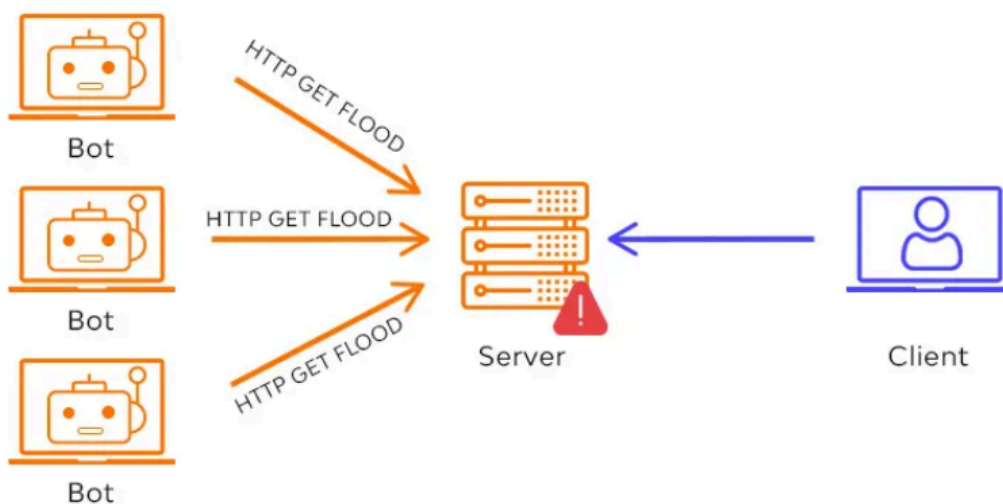
Attackers who use UDP floods send an abundance of UDP packets to a target's network, overwhelming it.



3. HTTP Flood Attack

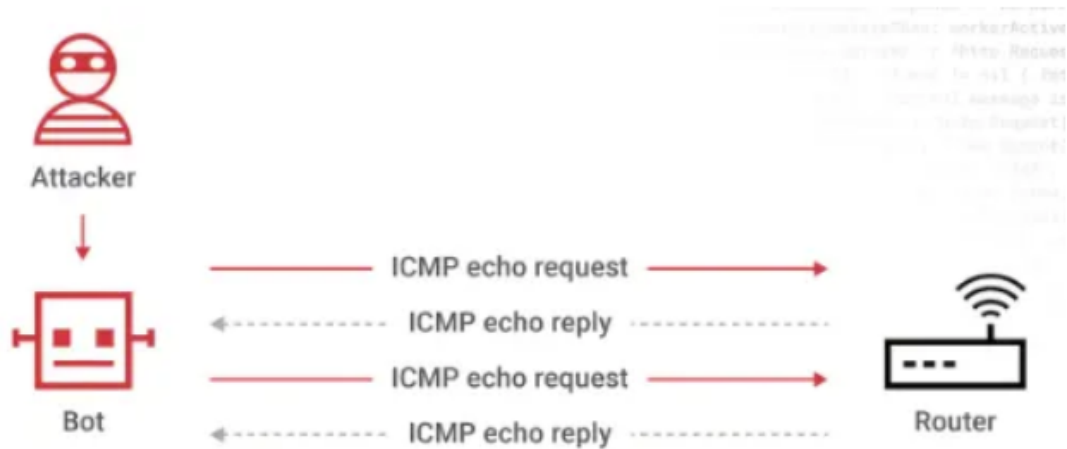
Attackers who use HTTP flood tactics overload a web server with too many HTTP requests.

HTTP Flood Attack



4. ICMP Flood Attack

Attackers who use the ICMP flood technique bombard a target with ICMP Echo Request (ping) packets, using up network resources.

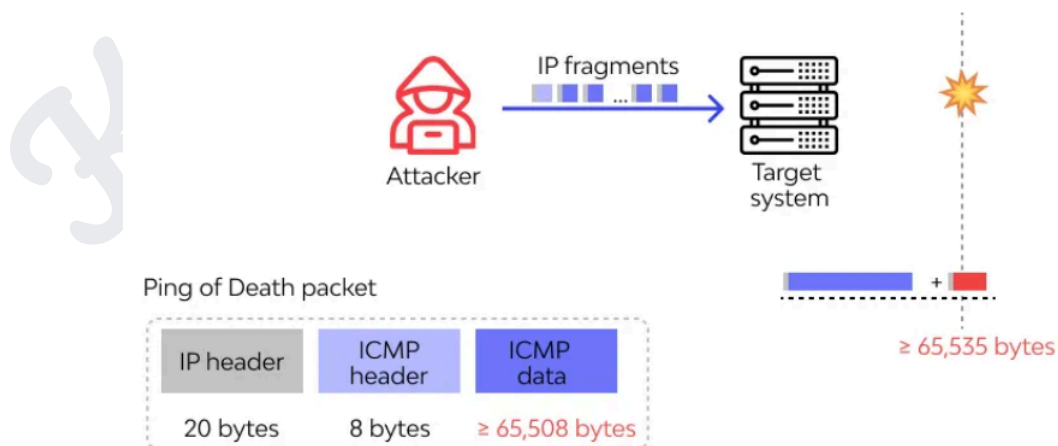


5. Ping of Death

A "Ping of Death" attack entails delivering an enormous or improperly formatted ICMP ping packet to a target, which causes the target's system to crash or stop responding because it cannot manage the size of the packet.

Ping of Death

How it works



Motivation behind DoS Attacks

The following are the main motivations of Denial of Service (DoS) attacks:

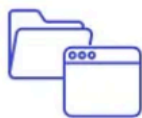
1. **Hactivism:** Attackers launch DoS attacks to obstruct the internet operations of institutions or organizations they disagree with in order to express their political, intellectual, or social concerns.
2. **Financial Gain:** Some cybercriminals use denial-of-service (DoS) assaults as distractions to enable data breaches, financial fraud, or other cybercrimes with the intention of making money off the ensuing mayhem.
3. **Competitive Advantage:** Companies or individuals may target the websites or services of rivals in an effort to obtain an advantage by interfering with their online visibility and client access.
4. **Extortion:** Attackers use the fear of disruption to their advantage by threatening to launch or escalate a DoS attack unless a victim pays a ransom.
5. **Cyberwarfare:** DoS attacks may be used by nation-states or state-sponsored individuals as part of cyberwarfare campaigns to disrupt vital communications and infrastructure.

Signs of DoS Attacks

The following are the top 5 indicators of a Denial of Service (DoS) attack:

- **Slow Response Times:** The assault overwhelms the system's ability to respond fast, resulting in noticeably long loading times and delays when attempting to access websites or services.
- **Frequent Connection Timeouts:** Timeouts and disconnections that occur frequently when attempting to connect to online services are signs that the server is unavailable.
- **Accessibility Issues:** Total inability to access a given service, website, or network as a result of the attack making it unavailable.
- **Unexplained Network Congestion:** Unexpectedly heavy network traffic and congestion affects the performance of the entire network.
- **Resource Exhaustion:** When the target system's CPU, memory, or bandwidth are all used up, it performs poorly and is unresponsive.

Signs of a Dos attack



Degradation in network performance

Especially when trying to open files stored online or when visiting websites



Specific website unavailable

A certain site does not open or cannot be found



Inability to access any website

Site is not available online



High volume of email spam

Higher-than-normal volume of spam in emails

CHAPTER2: DATABASES IN FOCUS

Purpose

Databases play a critical role in managing and analyzing the vast amounts of data generated. The primary purposes of incorporating databases into this project include:

1. **Logging Attack Data:** Storing detailed records of attack patterns, source IPs, timestamps, and other relevant information. This data is essential for understanding attack vectors and planning countermeasures.
2. **Tracking Security Events:** Maintaining a historical record of security incidents, system responses, and resolution outcomes to help in future attack predictions and strategy adjustments.
3. **Managing Network Traffic:** Analyzing traffic patterns, identifying anomalies, and making data-driven decisions to optimize network defenses and response strategies.

Data Flow

In this project, we are using the MySQL database is a widely used relational database management system known for its reliability and ease of use. In this project, MySQL is chosen for:

- **Logging Security Events:** Its robust querying capabilities allow for complex searches and aggregations of log data, which is crucial for analyzing patterns and trends in security events.
- **Complex Queries:** MySQL supports complex joins and subqueries, making it suitable for detailed analysis of attack data.

- **Scalability:** Although it is traditionally used for smaller datasets, its scalability options (such as sharding and replication) make it adaptable to larger volumes of log data.

I have created 5 Databases:

1. **Attack_Detection**
2. **Network_Traffic**
3. **System_Resources**
4. **Incident_Response**
5. **Security_Information**

DATABASE 1

Attack_Detection - Stores data related to various attacks, their types, sources, and detection rules.

```
CREATE DATABASE Attack_Detection;
```

DATABASE 2

Network_Traffic - Manages details about network traffic, protocols, and device information.

```
CREATE DATABASE Network_Traffic;
```

DATABASE 3

System_Resources - Monitors resource usage and system statistics.

```
CREATE DATABASE System_Resources;
```

DATABASE 4

Incident_Response - Handles records of security incidents, response plans, and reports.

```
CREATE DATABASE Incident_Response;
```

DATABASE 5

Security_Information - Contains information on vulnerabilities, patches, advisories, and threat intelligence.

```
CREATE DATABASE Security_Information;
```

CHAPTER3: INSIDE THE DATA

Schema Overview

Each database comprises several tables designed to capture and organize data effectively. Below is a brief overview of the table structures within each database:

Attack_Detection Database:

```
USE Attack_Detection;
```

TABLES

- **attacks:** Records information about attacks including ID, type, date, and source IP.

```
CREATE TABLE attacks (id INT PRIMARY KEY, attack_type INT,  
attack_date DATETIME, source_ip VARCHAR(15));
```

- **attack_types:** Defines types of attacks and their descriptions.

```
CREATE TABLE attack_types (id INT PRIMARY KEY, type_name VARCHAR(50),  
description VARCHAR(255));
```

- **sources:** Stores information about the source IP addresses and their geographical locations.

```
CREATE TABLE sources (id INT PRIMARY KEY, source_ip VARCHAR(15),  
source_country VARCHAR(50));
```

- **detection_rules:** Lists rules used for detecting various attack types.

```
CREATE TABLE detection_rules (id INT PRIMARY KEY, rule_name  
VARCHAR(50), rule_description VARCHAR(255));
```

- **alerts:** Contains alerts generated based on detected attacks, including alert levels and dates.

```
CREATE TABLE alerts (id INT PRIMARY KEY, attack_id INT, alert_date  
DATETIME, alert_level VARCHAR(20));
```

INSERT DATA IN ABOVE TABLES FOR Attack_Detection DB

```
INSERT INTO attacks (id, attack_type, attack_date, source_ip) 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');
```

```
INSERT INTO attack_types (id, type_name, description) VALUES (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');
```

```
INSERT INTO sources (id, source_ip, source_country) 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');
```

```
INSERT INTO detection_rules (id, rule_name, rule_description) 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');
```

```
INSERT INTO alerts (id, attack_id, alert_date, alert_level) 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');
```

RETRIEVE DATA IN ABOVE TABLES FOR Attack_Detection DB

```
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
NULL	NULL	NULL	NULL

```
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
NULL	NULL	NULL

```
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
NULL	NULL	NULL

```
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
NULL	NULL	NULL

```
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
NULL	NULL	NULL	NULL

Network_Traffic Database:

```
USE Network_Detection;
```

TABLES

- **traffic:** Logs network traffic details such as timestamp, source IP, destination IP, and protocol.

```
CREATE TABLE traffic ( id INT PRIMARY KEY, timestamp DATETIME,
source_ip VARCHAR(15), destination_ip VARCHAR(15), protocol
VARCHAR(10) );
```

- **protocols:** Defines network protocols with their descriptions.

```
CREATE TABLE protocols ( id INT PRIMARY KEY, protocol_name
VARCHAR(10), protocol_description VARCHAR(255) );
```


- **ip_addresses:** Stores information about IP addresses and their types (public/private).

```
CREATE TABLE ip_addresses ( id INT PRIMARY KEY, ip_address  
VARCHAR(15), ip_type VARCHAR(10) );
```

- **network_devices:** Details about network devices including their names and types.

```
CREATE TABLE network_devices ( id INT PRIMARY KEY, device_name  
VARCHAR(50), device_type VARCHAR(50) );
```

- **traffic_stats:** Records statistical data on traffic volume over time.

```
CREATE TABLE traffic_stats ( id INT PRIMARY KEY, timestamp DATETIME,  
traffic_volume INT );
```

INSERT DATA IN ABOVE TABLES FOR Network_Traffic DB

```
INSERT INTO traffic (id, timestamp, source_ip, destination_ip, protocol)  
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');
```

```
INSERT INTO protocols (id, protocol_name, protocol_description) 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');
```

```
INSERT INTO ip_addresses (id, ip_address, ip_type) 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');
```

```
INSERT INTO network_devices (id, device_name, device_type) VALUES (1,  
'Router', 'Cisco'), (2, 'Switch', 'HP'), (3, 'Firewall', 'Juniper'), (4,  
'Server', 'Dell'), (5, 'Client', 'Laptop');
```

```
INSERT INTO traffic_stats (id, timestamp, traffic_volume) VALUES (1,
```

```
'2022-01-01 12:00:00', 1000),(2, '2022-01-02 13:00:00', 1500),  
(3, '2022-01-03 14:00:00', 2000), (4, '2022-01-04 15:00:00', 2500),  
(5, '2022-01-05 16:00:00', 3000);
```

RETRIEVE DATA IN ABOVE TABLES FOR Network_Traffic DB

```
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
NULL	NULL	NULL	NULL	NULL

```
SELECT * FROM protocols;
```

id	protocol_name	protocol_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
NULL	NULL	NULL

```
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
NULL	NULL	NULL

```
SELECT * FROM network_devices;
```

id	device_name	device_type
1	Router	Cisco
2	Switch	HP
3	Firewall	Juniper
4	Server	Dell
5	Client	Laptop
NULL	NULL	NULL

```
SELECT * FROM traffic_stats;
```

id	timestamp	traffic_volume
1	2022-01-01 12:00:00	1000
2	2022-01-02 13:00:00	1500
3	2022-01-03 14:00:00	2000
4	2022-01-04 15:00:00	2500
5	2022-01-05 16:00:00	3000
NULL	NULL	NULL

System_Resources Database:

```
USE System_Resources;
```

TABLES

- **resource_usage:** Captures resource usage metrics including CPU, memory, and disk usage.

```
CREATE TABLE resource_usage ( id INT PRIMARY KEY, timestamp DATETIME,
cpu_usage FLOAT, memory_usage FLOAT, disk_usage FLOAT );
```

- **resources:** Details various system resources with descriptions.

```
CREATE TABLE resources ( id INT PRIMARY KEY, resource_name
VARCHAR(50), resource_description VARCHAR(255) );
```

- **system_stats:** Provides system load and uptime statistics.

```
CREATE TABLE system_stats ( id INT PRIMARY KEY, timestamp DATETIME,
system_load FLOAT, system_uptime INT );
```

- **process_list**: Records running processes with CPU usage information.

```
CREATE TABLE process_list ( id INT PRIMARY KEY, process_name
VARCHAR(50), process_pid INT, process_cpu_usage FLOAT );
```

- **user_sessions**: Tracks user session details including start and end times.

```
CREATE TABLE user_sessions ( id INT PRIMARY KEY, user_id INT,
session_start DATETIME, session_end DATETIME );
```

INSERT DATA IN ABOVE TABLES FOR System_Response DB

```
INSERT INTO resource_usage (id, timestamp, cpu_usage, memory_usage,
disk_usage) VALUES (1, '2022-01-01 12:00:00', 50.5, 65.3, 75.1), (2,
'2022-01-02 13:00:00', 55.0, 70.0, 80.0), (3, '2022-01-03 14:00:00', 60.5,
72.5, 85.0), (4, '2022-01-04 15:00:00', 65.0, 75.0, 90.0),
(5, '2022-01-05 16:00:00', 70.0, 78.0, 95.0);
```

```
INSERT INTO resources (id, resource_name, resource_description) VALUES (1,
'CPU', 'Central Processing Unit'), (2, 'Memory', 'Random Access Memory'),
(3, 'Disk', 'Storage Disk');
```

```
INSERT INTO system_stats (id, timestamp, system_load, system_uptime) VALUES
(1, '2022-01-01 12:00:00', 2.5, 3600), (2, '2022-01-02 13:00:00', 3.0,
7200), (3, '2022-01-03 14:00:00', 3.5, 10800), (4, '2022-01-04 15:00:00',
4.0, 14400), (5, '2022-01-05 16:00:00', 4.5, 18000);
```

```
INSERT INTO process_list (id, process_name, process_pid, process_cpu_usage)
VALUES (1, 'nginx', 1234, 10.5), (2, 'apache', 2345, 20.0),
(3, 'mysql', 3456, 25.5), (4, 'redis', 4567, 15.0), (5, 'mongodb', 5678,
22.5);
```

```
INSERT INTO user_sessions (id, user_id, session_start, session_end) VALUES
(1, 101, '2022-01-01 10:00:00', '2022-01-01 11:00:00'), (2, 102,
'2022-01-02 10:00:00', '2022-01-02 11:30:00'), (3, 103, '2022-01-03
11:00:00', '2022-01-03 12:45:00'), (4, 104, '2022-01-04 09:00:00',
'2022-01-04 10:15:00'), (5, 105, '2022-01-05 14:00:00', '2022-01-05
15:30:00');
```

RETRIEVE DATA IN ABOVE TABLES FOR System_Response DB

```
SELECT * FROM resource_usage;
```

id	timestamp	cpu_usage	memory_usage	disk_usage
1	2022-01-01 12:00:00	50.5	65.3	75.1
2	2022-01-02 13:00:00	55	70	80
3	2022-01-03 14:00:00	60.5	72.5	85
4	2022-01-04 15:00:00	65	75	90
5	2022-01-05 16:00:00	70	78	95
NULL	NULL	NULL	NULL	NULL

```
SELECT * FROM resources;
```

id	resource_name	resource_description
1	CPU	Central Processing Unit
2	Memory	Random Access Memory
3	Disk	Storage Disk
NULL	NULL	NULL

```
SELECT * FROM system_stats;
```

id	timestamp	system_load	system_uptime
1	2022-01-01 12:00:00	2.5	3600
2	2022-01-02 13:00:00	3	7200
3	2022-01-03 14:00:00	3.5	10800
4	2022-01-04 15:00:00	4	14400
5	2022-01-05 16:00:00	4.5	18000
NULL	NULL	NULL	NULL

```
SELECT * FROM process_list;
```

id	process_name	process_pid	process_cpu_usage
1	nginx	1234	10.5
2	apache	2345	20
3	mysql	3456	25.5
4	redis	4567	15
5	mongodb	5678	22.5
NULL	NULL	NULL	NULL

```
SELECT * FROM user_sessions;
```

id	user_id	session_start	session_end
1	101	2022-01-01 10:00:00	2022-01-01 11:00:00
2	102	2022-01-02 10:00:00	2022-01-02 11:30:00
3	103	2022-01-03 11:00:00	2022-01-03 12:45:00
4	104	2022-01-04 09:00:00	2022-01-04 10:15:00
5	105	2022-01-05 14:00:00	2022-01-05 15:30:00
NULL	NULL	NULL	NULL

Incident_Response Database:

```
USE Incident_Response;
```

TABLES

- **incidents:** Logs information about security incidents, their dates, types, and descriptions.

```
CREATE TABLE incidents ( id INT PRIMARY KEY, incident_date DATETIME,
incident_type INT, incident_description TEXT );
```

- **incident_types:** Defines types of incidents and their descriptions.

```
CREATE TABLE incident_types ( id INT PRIMARY KEY, type_name
VARCHAR(50), type_description TEXT );
```

- **response_plans:** Outlines plans for responding to different types of incidents.

```
CREATE TABLE response_plans ( id INT PRIMARY KEY, plan_name
VARCHAR(50), plan_description TEXT );
```

- **response_teams:** Contains details about response teams and their leaders.

```
CREATE TABLE response_teams ( id INT PRIMARY KEY, team_name
VARCHAR(50), team_lead VARCHAR(50) );
```

- **incident_reports:** Includes reports related to incidents with descriptions and dates.

```
CREATE TABLE incident_reports ( id INT PRIMARY KEY, incident_id INT,
report_date DATETIME, report_description TEXT );
```

INSERT DATA IN ABOVE TABLES FOR Incident_Response DB

```
INSERT INTO incidents (id, incident_date, incident_type,
incident_description) VALUES (1, '2022-01-01 12:00:00', 1, 'DDoS attack on
web server. '), (2, '2022-01-02 13:00:00', 2, 'SQL Injection detected on
login page. '), (3, '2022-01-03 14:00:00', 3, 'XSS attack identified on user
profile page. '), (4, '2022-01-04 15:00:00', 4, 'Brute Force attack observed
on FTP server. '), (5, '2022-01-05 16:00:00', 5, 'Phishing attempt detected
via email. ');
```

```
INSERT INTO incident_types (id, type_name, type_description) VALUES (1,
'DDoS Attack', 'Distributed Denial of Service attack'), (2, 'SQL Injection',
'Injection of malicious SQL statements into query fields. '), (3, 'XSS',
'Cross-Site Scripting, injecting malicious scripts into webpages. '),
(4, 'Brute Force', 'Repeatedly trying different passwords to gain
access. '), (5, 'Phishing', 'Attempting to deceive users into providing
sensitive information. ');
```

```
INSERT INTO response_plans (id, plan_name, plan_description) VALUES (1,
'Plan A', 'Mitigate DDoS attack by filtering traffic. '), (2, 'Plan B',
'Mitigate SQL Injection by sanitizing inputs. '), (3, 'Plan C', 'Implement
Content Security Policy to prevent XSS. '), (4, 'Plan D', 'Deploy account
lockout mechanism for brute force. '), (5, 'Plan E', 'Conduct user awareness
training to recognize phishing attempts. ');
```

```
INSERT INTO response_teams (id, team_name, team_lead) VALUES (1, 'Incident
Response Team', 'Alice Smith'), (2, 'Database Security Team', 'Bob
Johnson'), (3, 'Web Security Team', 'Carol Davis'), (4, 'Network Security
Team', 'David Brown'), (5, 'User Awareness Team', 'Eva Green');
```

```
INSERT INTO incident_reports (id, incident_id, report_date,
report_description) VALUES (1, 1, '2022-01-01 12:00:00', 'DDoS attack
detected and mitigated. '), (2, 2, '2022-01-02 13:30:00', 'SQL Injection
attack identified and patched. '), (3, 3, '2022-01-03 15:00:00', 'XSS
vulnerability fixed with updated security policy. '), (4, 4, '2022-01-04
16:00:00', 'Brute force attempts were blocked and account logout
implemented. '), (5, 5, '2022-01-05 17:00:00', 'Phishing attempt reported
and user education increased. ');
```

RETRIEVE DATA IN ABOVE TABLES FOR Incident_Response DB

```
SELECT * FROM incidents;
```

id	incident_date	incident_type	incident_description
1	2022-01-01 12:00:00	1	DDoS attack on web server.
2	2022-01-02 13:00:00	2	SQL Injection detected on login page.
3	2022-01-03 14:00:00	3	XSS attack identified on user profile page.
4	2022-01-04 15:00:00	4	Brute Force attack observed on FTP server.
5	2022-01-05 16:00:00	5	Phishing attempt detected via email.
NULL	NULL	NULL	NULL

```
SELECT * FROM incident_types;
```

id	type_name	type_description
1	DDoS Attack	Distributed Denial of Service attack
2	SQL Injection	Injection of malicious SQL statements into quer...
3	XSS	Cross-Site Scripting, injecting malicious scripts in...
4	Brute Force	Repeatedly trying different passwords to gain a...
5	Phishing	Attempting to deceive users into providing sensi...
NULL	NULL	NULL

```
SELECT * FROM response_plans;
```

id	plan_name	plan_description
1	Plan A	Mitigate DDoS attack by filtering traffic.
2	Plan B	Mitigate SQL Injection by sanitizing inputs.
3	Plan C	Implement Content Security Policy to prevent X...
4	Plan D	Deploy account logout mechanism for brute for...
5	Plan E	Conduct user awareness training to recognize p...
NULL	NULL	NULL


```
SELECT * FROM response_teams;
```

id	team_name	team_lead
1	Incident Response Team	Alice Smith
2	Database Security Team	Bob Johnson
3	Web Security Team	Carol Davis
4	Network Security Team	David Brown
5	User Awareness Team	Eva Green
NULL	NULL	NULL

```
SELECT * FROM incident_reports;
```

id	team_name	team_lead
1	Incident Response Team	Alice Smith
2	Database Security Team	Bob Johnson
3	Web Security Team	Carol Davis
4	Network Security Team	David Brown
5	User Awareness Team	Eva Green
NULL	NULL	NULL

Security_Information Database:

```
USE Security_Information;
```

TABLES

- **vulnerabilities:** Lists known vulnerabilities, their descriptions, and severities.

```
CREATE TABLE vulnerabilities ( id INT PRIMARY KEY, vuln_name  
VARCHAR(50), vuln_description TEXT, vuln_severity VARCHAR(20) );
```

- **patches:** Records information about patches released to fix vulnerabilities.

```
CREATE TABLE patches ( id INT PRIMARY KEY, patch_name VARCHAR(50),  
patch_description TEXT, patch_release_date DATETIME );
```

- **security_advisories:** Contains advisories related to security issues.

```
CREATE TABLE security_advisories ( id INT PRIMARY KEY, advisory_name VARCHAR(50), advisory_description TEXT );
```

- **threat_intelligence:** Provides threat intelligence including descriptions and threat levels.

```
CREATE TABLE threat_intelligence ( id INT PRIMARY KEY, threat_name VARCHAR(50), threat_description TEXT, threat_level VARCHAR(20) );
```

- **security_incidents:** Logs security incidents with dates.

```
CREATE TABLE security_incidents ( id INT PRIMARY KEY, incident_id INT, security_incident_date DATETIME );
```

INSERT DATA IN ABOVE TABLES FOR Security_Information DB

```
INSERT INTO vulnerabilities (id, vuln_name, vuln_description, vuln_severity) VALUES (1, 'SQL Injection', 'Allows attackers to execute SQL commands', 'High'), (2, 'Cross-Site Scripting', 'Allows attackers to inject scripts into web pages', 'Medium'), (3, 'Brute Force', 'Repeatedly attempting different passwords to gain unauthorized access', 'High'), (4, 'Phishing', 'Attempts to deceive users into revealing confidential information', 'Critical');
```

```
INSERT INTO patches (id, patch_name, patch_description, patch_release_date) VALUES (1, 'Patch 1', 'Fixes SQL Injection vulnerability', '2022-01-01 00:00:00'), (2, 'Patch 2', 'Fixes Cross-Site Scripting vulnerability', '2022-01-02 00:00:00'), (3, 'Patch 3', 'Mitigates Brute Force attack vulnerability', '2022-01-03 00:00:00'), (4, 'Patch 4', 'Updates security measures against Phishing', '2022-01-04 00:00:00');
```

```
INSERT INTO security_advisories (id, advisory_name, advisory_description) VALUES (1, 'Advisory 1', 'Advisory on SQL Injection vulnerabilities'), (2, 'Advisory 2', 'Advisory on Cross-Site Scripting attacks and mitigation strategies'), (3, 'Advisory 3', 'Brute Force attack prevention best practices'), (4, 'Advisory 4', 'Phishing prevention and user awareness tips');
```

```
INSERT INTO threat_intelligence (id, threat_name, threat_description,
```

```
threat_level) VALUES (1, 'DDoS Threat', 'Potential DDoS attack targeting critical infrastructure', 'High'), (2, 'SQL Injection Threat', 'Potential SQL Injection attack targeting web applications', 'High'), (3, 'XSS Threat', 'Potential XSS attack targeting user input fields', 'Medium'), (4, 'Brute Force Threat', 'Potential brute force attack targeting login pages', 'High'), (5, 'Phishing Threat', 'Increased phishing attempts targeting organizational email', 'Critical');
```

```
INSERT INTO security_incidents (id, incident_id, security_incident_date) VALUES (1, 1, '2022-01-01 12:00:00'), (2, 2, '2022-01-02 13:00:00'), (3, 3, '2022-01-03 14:00:00'), (4, 4, '2022-01-04 15:00:00'), (5, 5, '2022-01-05 16:00:00');
```

RETRIEVE DATA IN ABOVE TABLES FOR Security_Information DB

```
SELECT * FROM vulnerabilities;
```

id	vuln_name	vuln_description	vuln_severity
1	SQL Injection	Allows attackers to execute SQL commands	High
2	Cross-Site Scripting	Allows attackers to inject scripts into web pages	Medium
3	Brute Force	Repeatedly attempting different passwords to ...	High
4	Phishing	Attempts to deceive users into revealing confid...	Critical
NULL	NULL	NULL	NULL

```
SELECT * FROM patches;
```

id	patch_name	patch_description	patch_release_date
1	Patch 1	Fixes SQL Injection vulnerability	2022-01-01 00:00:00
2	Patch 2	Fixes Cross-Site Scripting vulnerability	2022-01-02 00:00:00
3	Patch 3	Mitigates Brute Force attack vulnerability	2022-01-03 00:00:00
4	Patch 4	Updates security measures against Phishing	2022-01-04 00:00:00
NULL	NULL	NULL	NULL

```
SELECT * FROM security_advisories;
```

id	advisory_name	advisory_description
1	Advisory 1	Advisory on SQL Injection vulnerabilities
2	Advisory 2	Advisory on Cross-Site Scripting attacks and mit...
3	Advisory 3	Brute Force attack prevention best practices
4	Advisory 4	Phishing prevention and user awareness tips
NULL	NULL	NULL

```
SELECT * FROM threat_intelligence;
```

id	threat_name	threat_description	threat_level
1	DDoS Threat	Potential DDoS attack targeting critical infrastru...	High
2	SQL Injection Threat	Potential SQL Injection attack targeting web ap...	High
3	XSS Threat	Potential XSS attack targeting user input fields	Medium
4	Brute Force Threat	Potential brute force attack targeting login pages	High
5	Phishing Threat	Increased phishing attempts targeting organiza...	Critical
NULL	NULL	NULL	NULL

```
SELECT * FROM security_incidents;
```

id	incident_id	security_incident_date
1	1	2022-01-01 12:00:00
2	2	2022-01-02 13:00:00
3	3	2022-01-03 14:00:00
4	4	2022-01-04 15:00:00
5	5	2022-01-05 16:00:00
NULL	NULL	NULL

CHAPTER4: CRITICAL QUERIES FROM NETWORK SECURITY

Analyze

Queries are used to identify signs of potential DDoS attacks or other malicious activities and analyze patterns and trends in the data, providing insights into attack characteristics and effectiveness. Effective queries can assist in making real-time decisions to mitigate ongoing attacks and adjust security measures accordingly.

Retrieve All Attacks and Associated Details

```
USE Attack_Detection;

SELECT a.id AS attack_id,
       at.type_name AS attack_type,
       a.attack_date,
       s.source_country,
       al.alert_level
FROM attacks a
JOIN attack_types at ON a.attack_type = at.id
JOIN sources s ON a.source_ip = s.source_ip
JOIN alerts al ON a.id = al.attack_id;
```

attack_id	attack_type	attack_date	source_country	alert_level
1	DDoS	2022-01-01 12:00:00	USA	High
2	SQL Injection	2022-01-02 13:00:00	China	Medium
3	Cross-Site Scripting	2022-01-03 14:00:00	Russia	Low
4	DDoS	2022-01-04 15:00:00	India	High
5	SQL Injection	2022-01-05 16:00:00	Brazil	Medium

Get Network Traffic Details for a Specific IP

```
USE Network_Traffic;

SELECT * FROM traffic
WHERE source_ip = '192.168.1.100';
```

id	timestamp	source_ip	destination_ip	protocol
1	2022-01-01 12:00:00	192.168.1.100	192.168.1.1	TCP
NULL	NULL	NULL	NULL	NULL

Monitor System Resource Usage During an Attack

```
USE System_Resources;

SELECT * FROM resource_usage
WHERE timestamp BETWEEN '2022-01-01 12:00:00' AND '2022-01-01 13:00:00';
```

id	timestamp	cpu_usage	memory_usage	disk_usage
1	2022-01-01 12:00:00	50.5	65.3	75.1
NULL	NULL	NULL	NULL	NULL

Retrieve Incident Reports and Response Plans

```
USE Incident_Response;

SELECT i.id AS incident_id,
       i.incident_date,
       it.type_name AS incident_type,
       i.incident_description,
       ir.report_date,
       ir.report_description,
       rp.plan_name,
       rp.plan_description
FROM incidents i
JOIN incident_types it ON i.incident_type = it.id
JOIN incident_reports ir ON i.id = ir.incident_id
JOIN response_plans rp ON i.id = rp.id;
```

incident_id	incident_date	incident_type	incident_description	report_date	report_description	plan_name	plan_description
1	2022-01-01 12:00:00	DDoS Attack	DDoS attack on web server.	2022-01-01 12:00:00	DDoS attack detected and mitigated.	Plan A	Mitigate DDoS attack by filtering
2	2022-01-02 13:00:00	SQL Injection	SQL Injection detected on login page.	2022-01-02 13:30:00	SQL Injection attack identified and patched.	Plan B	Mitigate SQL Injection by sanitiz
3	2022-01-03 14:00:00	XSS	XSS attack identified on user profile page.	2022-01-03 15:00:00	XSS vulnerability fixed with updated security pol...	Plan C	Implement Content Security Pol
4	2022-01-04 15:00:00	Brute Force	Brute Force attack observed on FTP server.	2022-01-04 16:00:00	Brute force attempts were blocked and account...	Plan D	Deploy account lockout mechan
5	2022-01-05 16:00:00	Phishing	Phishing attempt detected via email.	2022-01-05 17:00:00	Phishing attempt reported and user education l...	Plan E	Conduct user awareness traini

Check for Known Vulnerabilities and Their Patches

```
USE Security_Information;

SELECT v.id,
       v.vuln_description,
       v.vuln_severity,
       p.id,
       p.patch_description,
       p.patch_release_date
FROM vulnerabilities v
LEFT JOIN patches p ON v.id = p.id;
```

id	vuln_description	vuln_severity	id	patch_description	patch_release_date
1	Allows attackers to execute SQL commands	High	1	Fixes SQL Injection vulnerability	2022-01-01 00:00:00
2	Allows attackers to inject scripts into web pages	Medium	2	Fixes Cross-Site Scripting vulnerability	2022-01-02 00:00:00
3	Repeatedly attempting different passwords to ...	High	3	Mitigates Brute Force attack vulnerability	2022-01-03 00:00:00
4	Attempts to deceive users into revealing confid...	Critical	4	Updates security measures against Phishing	2022-01-04 00:00:00

CHAPTER 5: PROJECT OUTPUT


In this project, we have constructed a framework for managing and analyzing Denial of Service (DoS) attacks through a multifaceted approach involving several specialized databases. Each database plays a critical role in capturing and processing different aspects of network security, from attack detection and network traffic analysis to system resource monitoring and incident response.

The **ATTACK_DETECTION** database provides a foundational repository for logging attack details, classifying attack types, and generating alerts based on detection rules. This structured data allows for a nuanced understanding of attack patterns and source information, which is essential for effective mitigation strategies.

The **NETWORK_TRAFFIC** database complements this by offering detailed insights into network activity, including traffic volume and protocol usage. This information is crucial for identifying unusual traffic patterns that may indicate ongoing or potential attacks.

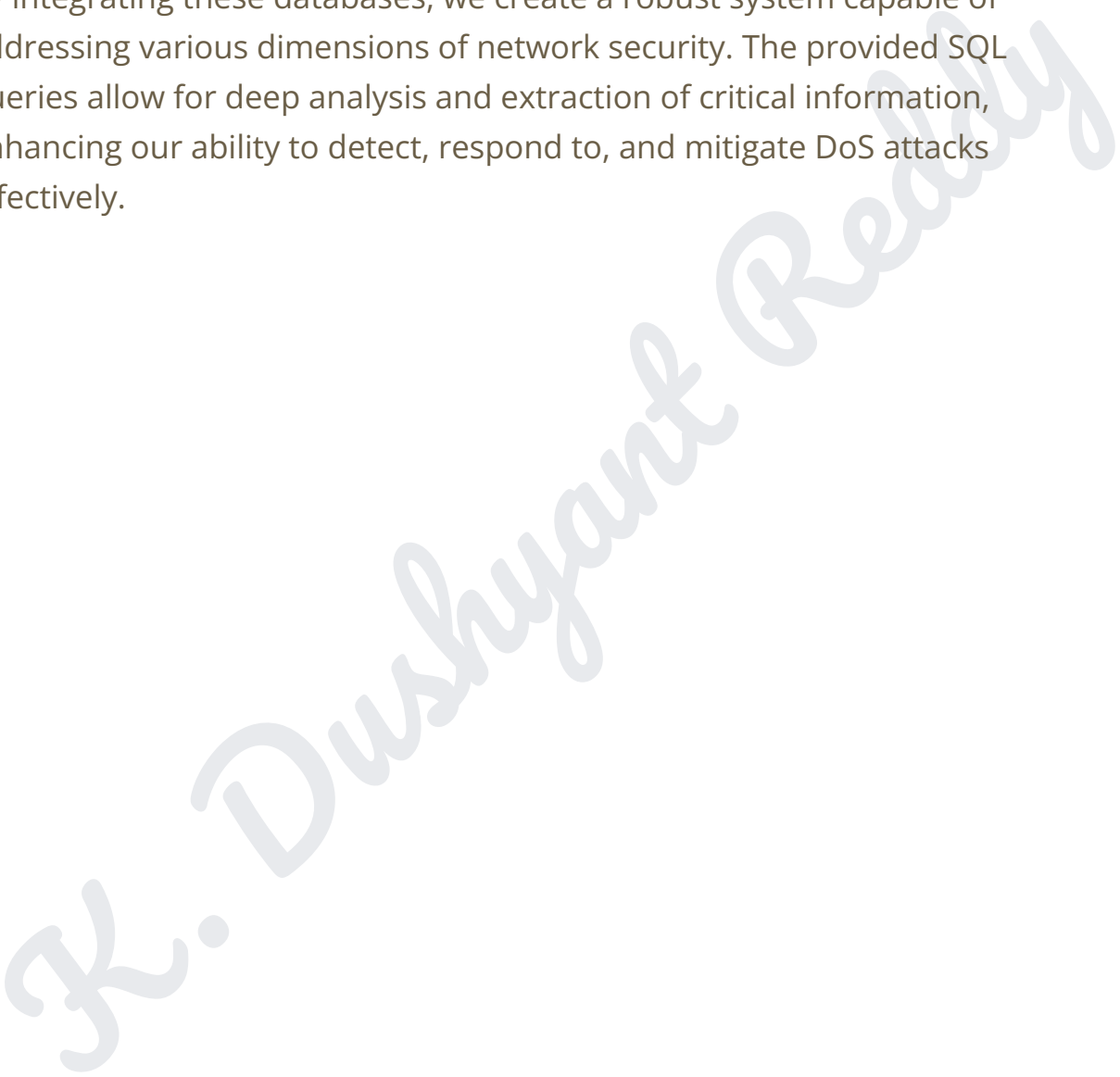
The **SYSTEM_RESOURCES** is pivotal for monitoring system performance metrics such as CPU, memory, and disk usage. By analyzing this data during attack events, we can assess the impact on system resources and optimize our response to prevent system overloads and outages.

The **INCIDENT_RESPONSE** database captures all aspects of incident management, including detailed records of incidents, response plans, and team activities. This facilitates a structured approach to addressing security incidents and ensures that response efforts are documented and refined.



Finally, the **SECURITY_INFORMATION** database consolidates data on vulnerabilities, patches, and threat intelligence. This comprehensive repository supports proactive security measures by tracking known vulnerabilities and ensuring timely application of patches and updates.

By integrating these databases, we create a robust system capable of addressing various dimensions of network security. The provided SQL queries allow for deep analysis and extraction of critical information, enhancing our ability to detect, respond to, and mitigate DoS attacks effectively.



CHAPTER6: CONCLUSION

This project demonstrates the effectiveness of a multi-database approach in managing Denial of Service (DoS) attacks. By utilizing the **ATTACK_DETECTION**, **NETWORK_TRAFFIC**, **SYSTEM_RESOURCES**, **INCIDENT_RESPONSE**, and **SECURITY_INFORMATION** databases, we create a comprehensive framework for monitoring and responding to security threats.

- **ATTACK_DETECTION** captures attack details and alerts.
- **NETWORK_TRAFFIC** monitors traffic patterns.
- **SYSTEM_RESOURCES** tracks system performance.
- **INCIDENT_RESPONSE** manages incident records and response plans.
- **SECURITY_INFORMATION** provides data on vulnerabilities and patches.

Together, these databases allow for detailed analysis and effective mitigation of DoS attacks, improving overall network security. This integrated approach ensures a well-rounded defense strategy and enhances our capability to address and prevent security threats.