Cybersecurity Database Management

1. Introduction

In the ever-evolving landscape of cybersecurity, effective management and analysis of network infrastructure data are paramount. This paper guides you through the process of creating a comprehensive database, cansamzone, designed to store and organize critical information related to domains, network devices, security incidents, and more.A black screen with a black background

Description automatically generated

2. Database Creation

2.1 Step 1: Creating the Database

Our journey begins with the creation of the cansamzone database. Using MySQL, execute the following command to initialize the database:

CREATE DATABASE cansamzone;

This command establishes the foundation for our cybersecurity management system.

2.2 Step 2: Selecting the Database

Once the database is created, select it to perform operations within its scope:

USE cansamzone;

Now, any subsequent queries will be directed to the cansamzone database.

3. Tables and Structure

3.1 Domains Table

Our first table, domains, is designed to store information about registered domains:

A screenshot of a computer program

Description automatically generated

CREATE TABLE domains (

id INT AUTO\_INCREMENT PRIMARY KEY,

domain\_name VARCHAR(255) NOT NULL UNIQUE,

registrar\_name VARCHAR(255),

registration\_date DATE,

expiration\_date DATE,

updated\_date DATE,

registrar VARCHAR(255)

);

This table captures crucial details such as domain name, registrar information, and registration dates.

3.2 Network Devices Table

The network\_devices table manages information about various network devices:

CREATE TABLE network\_devices (

device\_id INT AUTO\_INCREMENT PRIMARY KEY,

device\_name VARCHAR(255),

device\_type VARCHAR(255),

ip\_address VARCHAR(15),

mac\_address VARCHAR(17),

location VARCHAR(255),

last\_scan\_date TIMESTAMP

);

This table allows us to track device names, types, IP addresses, and more, aiding in network infrastructure management.

3.3 Nmap Scan Results Table

The nmap\_scan\_results table records results from Nmap scans:

A screenshot of a computer program

Description automatically generated

CREATE TABLE nmap\_scan\_results (

id INT AUTO\_INCREMENT PRIMARY KEY,

ip\_address VARCHAR(50),

scan\_type VARCHAR(50),

open\_ports VARCHAR(255),

scan\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Here, details such as IP addresses, scan types, and open ports are stored for analysis.

3.4 RMA Table

The rma table handles records of Return Merchandise Authorizations (RMAs) for network devices:

A screen shot of a computer

Description automatically generated

CREATE TABLE rma (

id INT AUTO\_INCREMENT PRIMARY KEY,

device\_id INT,

description TEXT,

resolution TEXT,

FOREIGN KEY (device\_id) REFERENCES network\_devices(device\_id)

);

This table links to the network\_devices table, providing a comprehensive view of device maintenance.

3.5 Scan Results Table

The scan\_results table stores results from scans performed on network devices:

A screenshot of a computer program

Description automatically generated

CREATE TABLE scan\_results (

id INT AUTO\_INCREMENT PRIMARY KEY,

device\_id INT,

result\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

result\_text TEXT,

FOREIGN KEY (device\_id) REFERENCES network\_devices(device\_id)

);

This table complements the nmap\_scan\_results table, offering detailed insights into device scans.

3.6 Security Incidents Table

Incidents are logged in the security\_incidents table:

A screen shot of a computer screen

Description automatically generated

CREATE TABLE security\_incidents (

id INT AUTO\_INCREMENT PRIMARY KEY,

incident\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

description TEXT,

resolution TEXT

);

This table aids in the documentation and resolution of security incidents.

3.7 Servermap Table

The servermap table provides a mapping of servers:

A screenshot of a computer program

Description automatically generated

CREATE TABLE servermap (

id INT AUTO\_INCREMENT PRIMARY KEY,

server\_name VARCHAR(255) NOT NULL,

location VARCHAR(255) NOT NULL,

ip\_address VARCHAR(255) NOT NULL,

status VARCHAR(255) NOT NULL,

last\_updated TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

This table helps track server names, locations, IP addresses, and their status.

3.8 Users Table

User information is managed in the users table:

A screen shot of a computer program

Description automatically generated

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

role VARCHAR(255) NOT NULL,

company\_name VARCHAR(255) DEFAULT NULL

);

This table facilitates user management, including roles and associated company names.

3.9 Vulnerabilities Table

The vulnerabilities table tracks vulnerabilities associated with network devices:

A screen shot of a computer screen

Description automatically generated

CREATE TABLE vulnerabilities (

id INT AUTO\_INCREMENT PRIMARY KEY,

device\_id INT,

vulnerability\_name VARCHAR(255),

severity VARCHAR(50),

description TEXT,

FOREIGN KEY (device\_id) REFERENCES network\_devices(device\_id)

);

This table provides insights into potential security risks related to network devices.

4. Database Utilization

The diverse set of tables in the cansamzone database offers a comprehensive approach to cybersecurity management. From tracking domain information to monitoring network devices, conducting scans, and managing user access, this database serves as a robust foundation for analyzing and addressing cybersecurity challenges.

Utilizing the cansamzone Database for Cybersecurity Analysis

1. Accessing the Database

Ensure you have the necessary credentials to access the MySQL server where the cansamzone database resides. Use the following command to log in:

mysql -u your\_username -p your\_password

Select the cansamzone database:

USE cansamzone;

-- Basic Retrieval Queries:

-- Retrieve all records from a specific table:

SELECT \* FROM scan\_results;A screenshot of a computer

Description automatically generated

-- Retrieve specific columns from a table:

SELECT ip\_address, scan\_type FROM nmap\_scan\_results;

A screenshot of a computer program

Description automatically generated

-- Retrieve a specific number of records:

SELECT \* FROM vulnerabilities LIMIT 10;

-- Filtering Queries:

-- Retrieve records based on a condition:

SELECT \* FROM domains WHERE domain\_name = 'example.com';

-- Retrieve records with multiple conditions:

SELECT \* FROM network\_devices WHERE device\_type = 'Router' AND location = 'Datacenter';

-- Sorting Queries:

-- Sort records in ascending order:

SELECT \* FROM users ORDER BY username ASC;

-- Sort records in descending order:

SELECT \* FROM servermap ORDER BY server\_name DESC;

-- Aggregate Queries:

-- Calculate the total number of records:

SELECT COUNT(\*) FROM security\_incidents;

-- Calculate the average value of a column:

SELECT AVG(scan\_date) FROM nmap\_scan\_results;

-- Calculate the sum of a column:

SELECT SUM(device\_id) FROM network\_devices;

-- Grouping Queries:

-- Group records based on a column:

SELECT device\_type, COUNT(\*) FROM network\_devices GROUP BY device\_type;

-- Apply aggregate functions on grouped data:

SELECT device\_type, AVG(device\_id) FROM network\_devices GROUP BY device\_type;

-- Join Queries:

-- Combine records from two or more tables:

SELECT \* FROM domains JOIN scan\_results ON domains.domain\_name = scan\_results.domain\_name;

-- Subqueries:

-- Use a subquery to retrieve data:

SELECT \* FROM vulnerabilities WHERE vulnerability\_type IN (SELECT vulnerability\_type FROM security\_incidents);

-- Use a subquery for comparison:

SELECT \* FROM vulnerabilities WHERE device\_id > (SELECT AVG(device\_id) FROM network\_devices);

-- Updating Data:

-- Update records in a table:

UPDATE users SET role = 'Moderator' WHERE username = 'admin';

-- Deleting Data:

-- Delete records from a table:

DELETE FROM security\_incidents WHERE severity = 'Low';

-- Combining Queries:

-- Combine results from multiple queries:

(SELECT username FROM users) UNION (SELECT device\_name FROM network\_devices);

