```
/
  ********************************************************************************
  *********
Use Case: Investigating Multiple Failed Login Attempts (Potential Brute -Force
  Attack)
task is to investigate and identify:

1. Which accounts had the most failed login attempts

2. When the attempts happened

3. The IP addresses involved

4. Patterns that might indicate brute-force or credential-stuffing attacks
********************************************************************************
  ********/

USE master;     --this is used to flush the Cybersecurity4_DB sql fails to flush
  the current DB in use which is Cybersecurity4_DB
GO

DROP DATABASE IF EXISTS Cybersecurity4_DB;
GO
--Step 1: Create the Database
CREATE  DATABASE Cybersecurity4_DB;
GO
USE Cybersecurity4_DB;
GO

-- Drop the table if it exists
DROP TABLE IF EXISTS LoginAudit;

--step 2: Create the table again
CREATE TABLE LoginAudit (
    LogID INT IDENTITY(1,1) PRIMARY KEY,
    Username NVARCHAR(100),
    LoginTime DATETIME,
    Success BIT, -- 1 = success, 0 = failed
    IPAddress VARCHAR(50)
);

-- Insert extracted data into the LoginAudit table
INSERT INTO LoginAudit (Username, LoginTime, Success, IPAddress)
VALUES
('admin', GETDATE() - 1, 0, '192.168.1.10'),
('admin', GETDATE() - 1, 0, '192.168.1.10'),
('admin', GETDATE() - 1, 0, '192.168.1.10'),
('admin', GETDATE() - 1, 0, '192.168.1.11'),
('user1', GETDATE() - 1, 1, '192.168.1.15'),
```

```
('user1', GETDATE() - 2, 0, '192.168.1.15'),
('user1', GETDATE() - 2, 0, '192.168.1.15'),
('user2', GETDATE() - 2, 1, '10.0.0.5'),
('hacker', GETDATE() - 1, 0, '203.0.113.1'),
('hacker', GETDATE() - 1, 0, '203.0.113.1'),
('hacker', GETDATE() - 1, 0, '203.0.113.1'),
('hacker', GETDATE() - 1, 0, '203.0.113.1'),
('hacker', GETDATE() - 1, 0, '203.0.113.23'),
('hacker', GETDATE() - 1, 0, '203.0.113.23'),
('hacker', GETDATE() - 1, 0, '203.0.113.23'),
('hacker', GETDATE() - 1, 0, '203.0.113.23'),
('hacker', GETDATE() - 1, 0, '203.0.113.23'),
('hacker', GETDATE() - 1, 0, '203.0.113.23');

-- Display the contents of the LoginAudit table
SELECT * FROM LoginAudit;

GO

-----------------------------------------------------

--Step 3: Identify Accounts With the Most Failed Login Attempts
SELECT
    Username,
    COUNT(*) AS FailedAttempts
FROM LoginAudit
WHERE success = 0   --it is chosen success instead of failed based on the column of
   the table which is named success
GROUP BY Username
ORDER BY FailedAttempts DESC;

GO
-----------------------------------------------------

--Step 4: Timeline of Failed Attempts for a Suspicious Account
SELECT
    LoginTime,
    IPAddress
FROM LoginAudit
WHERE Username = 'admin' AND Success = 0
ORDER BY LoginTime;

GO

-----------------------------------------------------

--Step 3: IP Addresses With Most Failed Attempts
SELECT
    IPAddress,
```

```
    COUNT(*) AS FailedAttempts
FROM LoginAudit
WHERE Success = 0
GROUP BY IPAddress
ORDER BY FailedAttempts DESC;

GO
------------------------------------------------------

--Step 4: Detect Brute-Force Behavior (e.g. > 3 failures in 5 minutes)
-- This shows accounts that had >3 failed attempts from the same IP in 5-minute
  intervals
--CTE below
WITH AttemptsWindow AS (
    SELECT
        Username,
        IPAddress,
        CAST(LoginTime AS DATE) AS AttemptDate,
        DATEPART(HOUR, LoginTime) AS Hour,
        (DATEPART(MINUTE, LoginTime) / 5) AS FiveMinWindow
    FROM LoginAudit
    WHERE Success = 0
)


------------------------------------------------------

--The Query That Uses the CTE above, see below
SELECT
    Username,
    IPAddress,
    AttemptDate,
    Hour,
    FiveMinWindow,
    COUNT(*) AS AttemptCount
FROM AttemptsWindow
GROUP BY Username, IPAddress, AttemptDate, Hour, FiveMinWindow
HAVING COUNT(*) >= 3
ORDER BY AttemptCount DESC;

GO
------------------------------------------------------------
DROP VIEW if exists vw_FailedLoginBursts
GO
--creating a view which is permanent using the above CTE
-- Create the view for failed login bursts
CREATE VIEW vw_FailedLoginBursts AS
SELECT
    Username,
```

```
    IPAddress,
    CAST(LoginTime AS DATE) AS AttemptDate,
    DATEPART(HOUR, LoginTime) AS Hour,
    (DATEPART(MINUTE, LoginTime) / 5) AS FiveMinWindow,
    COUNT(*) AS AttemptCount
FROM LoginAudit
WHERE Success = 0
GROUP BY
    Username,
    IPAddress,
    CAST(LoginTime AS DATE),
    DATEPART(HOUR, LoginTime),
    (DATEPART(MINUTE, LoginTime) / 5)
HAVING COUNT(*) >= 3;
GO

-- Now, select the data from the view
SELECT * FROM vw_FailedLoginBursts
ORDER BY AttemptCount DESC;
GO



------------------------------------------------------------
DROP TABLE IF EXISTS SuspiciousIPBlocklist;
GO
--Step 5: Generate a Blocklist of Malicious IPs
-- Identify IPs with more than 5 failed attempts
SELECT DISTINCT IPAddress AS Dropped_IPAddress, COUNT(*) AS FailedAttempts  --
  Count the number of failed login attempts
INTO SuspiciousIPBlocklist
FROM LoginAudit
WHERE Success = 0
GROUP BY IPAddress
HAVING COUNT(*) > 5;

-- View the blocklist
SELECT * FROM SuspiciousIPBlocklist;

GO
DROP TABLE IF EXISTS SuspiciousIPBlocklist;
GO
```