# Formative Assessment 2

# CSP632

NAME: TAMEEZ
SURNAME: HOOSAIN
ICAS NO: 20231803
ID NO: 0010165255083
HAND OUT DATE: 17/08/2023
DUE DATE:11/09/2023
HAND IN DATE: 07/09/2023

## Contents

## Data Collection (5 marks):

Your first task is to create a mock CSV file named MediHealthLogs.csv containing relevant data. The file should include columns for Time Generated, User, Action, Target, and IP Address. Populate the CSV with simulated log entries representing different actions by users.

Created CSV File

# Define the data

$data = @"

TimeGenerated,User,Action,Target,IPAddress,Details

2023-08-30 10:15:00,DoctorSmith,Login,PatientRecordsDB,192.168.1.101,"Attempted SQL Injection: SELECT * FROM users".

2023-08-30 10:30:12,NurseJohnson,View,LabResults,192.168.1.203,""

2023-08-30 11:05:28,AdminUser,Update,MedicationSchedule,192.168.1.55,""

2023-08-30 11:20:05,DoctorWhite,Login,PatientRecordsDB,192.168.1.128,"Attempted SQL Injection: UNION ALL SELECT * FROM users".

2023-08-30 12:45:18,PatientBrown,View,PrescriptionHistory,192.168.1.76,""

2023-08-30 13:30:40,DoctorSmith,Update,LabReports,192.168.1.101,""

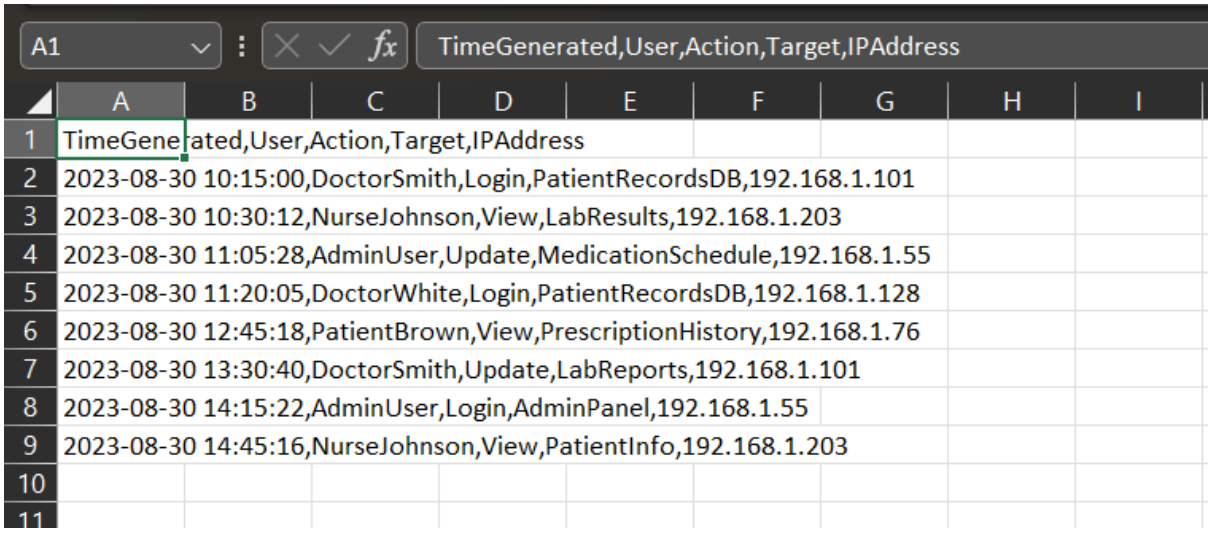
2023-08-30 14:15:22,AdminUser,Login,AdminPanel,192.168.1.55,"Suspicious Outbound Connection: OUTBOUND_CONNECTION"

2023-08-30 14:45:16,NurseJohnson,View,PatientInfo,192.168.1.203,""

"@

# Convert the data to a CSV file

$data | Convert From-Csv | Export-Csv -Path "C:\Users\Tameez\Desktop\Campus\2nd Semester\CSP632\FA2\MediHealthlogs.csv" -NoTypeInformation


A1
TimeGenerated,User,Action,Target,IPAddress
A B C D E F G H I
1 TimeGenerated,User,Action,Target,IPAddress
2 2023-08-30 10:15:00,DoctorSmith,Login,PatientRecordsDB,192.168.1.101
3 2023-08-30 10:30:12,NurseJohnson,View,LabResults,192.168.1.203
4 2023-08-30 11:05:28,AdminUser,Update,MedicationSchedule,192.168.1.55
5 2023-08-30 11:20:05,DoctorWhite,Login,PatientRecordsDB,192.168.1.128
6 2023-08-30 12:45:18,PatientBrown,View,PrescriptionHistory,192.168.1.76
7 2023-08-30 13:30:40,DoctorSmith,Update,LabReports,192.168.1.101
8 2023-08-30 14:15:22,AdminUser,Login,AdminPanel,192.168.1.55
9 2023-08-30 14:45:16,NurseJohnson,View,PatientInfo,192.168.1.203
10
11

## PowerShell Query (15 marks):

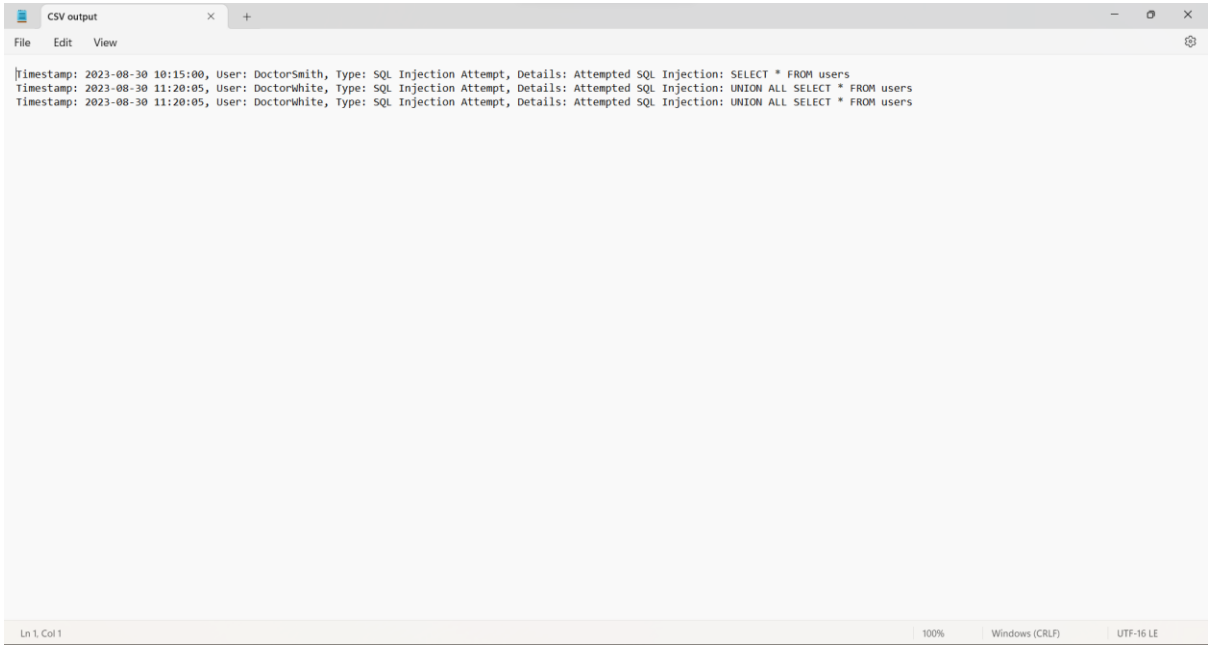
Write a PowerShell script to read the CSV file and run Kusto Query Language (KQL) queries to identify potential unauthorized access and data exfiltration activities. Focus on finding SQL injection attempts and suspicious outbound connections.

```
# Define the CSV file path

$csvFilePath = "C:\Users\Tameez\Desktop\Campus\2nd Semester\CSP632\FA2\MediHealthLogs Output.txt"


# Define the output file for suspicious activities

$outputFilePath = "C:\Users\Tameez\Documents\CSV output.txt"


# Define keywords for SQL injection and outbound connections

$keywords = @("SELECT * FROM", "UNION ALL SELECT", "INSERT INTO", "UPDATE", "DELETE FROM", "OUTBOUND_CONNECTION")


# Initialize an array to store suspicious activities

$suspiciousActivities = @()


# Read the CSV file

try {

   $csvData = Import-Csv -Path $csvFilePath

} catch {

   Write-Host "Error reading CSV file: $_"

   exit

}


# Initialize a StringBuilder to store the output

$outputStringBuilder = [System.Text.StringBuilder]::new()


# Iterate through each log entry

foreach ($entry in $csvData) {

   # Check for SQL injection attempts

   foreach ($keyword in $keywords) {
```

```
        if ($entry.Action -eq "Login" -and $entry.Target -eq "PatientRecordsDB" -and $entry.User -notlike
"Admin*") {
            # Check if the login attempt contains SQL injection keywords
            if ($entry.Details -like "*$keyword*") {
                $suspiciousActivities += [PSCustomObject]@{
                    Timestamp = $entry.TimeGenerated
                    User = $entry.User
                    Type = "SQL Injection Attempt"
                    Details = $entry.Details
                }

                # Append the suspicious activity to the output StringBuilder
                $outputStringBuilder.AppendLine("Timestamp: $($entry.TimeGenerated), User:
$($entry.User), Type: SQL Injection Attempt, Details: $($entry.Details)")
            }
        }
    }

    # Check for suspicious outbound connections
    if ($entry.IPAddress -ne "192.168.1.55" -and $entry.Target -ne "AdminPanel") {
        # Check if the log entry indicates an outbound connection
        if ($entry.Details -like "*OUTBOUND_CONNECTION*") {
            $suspiciousActivities += [PSCustomObject]@{
                Timestamp = $entry.TimeGenerated
                User = $entry.User
                Type = "Suspicious Outbound Connection"
                Details = $entry.Details
            }

            # Append the suspicious activity to the output StringBuilder
            $outputStringBuilder.AppendLine("Timestamp: $($entry.TimeGenerated), User:
$($entry.User), Type: Suspicious Outbound Connection, Details: $($entry.Details)")
```

```
    }
  }
}

# Export suspicious activities to a .txt file
$outputStringBuilder.ToString() | Out-File -FilePath $outputFilePath -Append

Write-Host "Suspicious activities written to $outputFilePath"
```

## SQL Injections results:




"Timestamp","User","Type","Details"

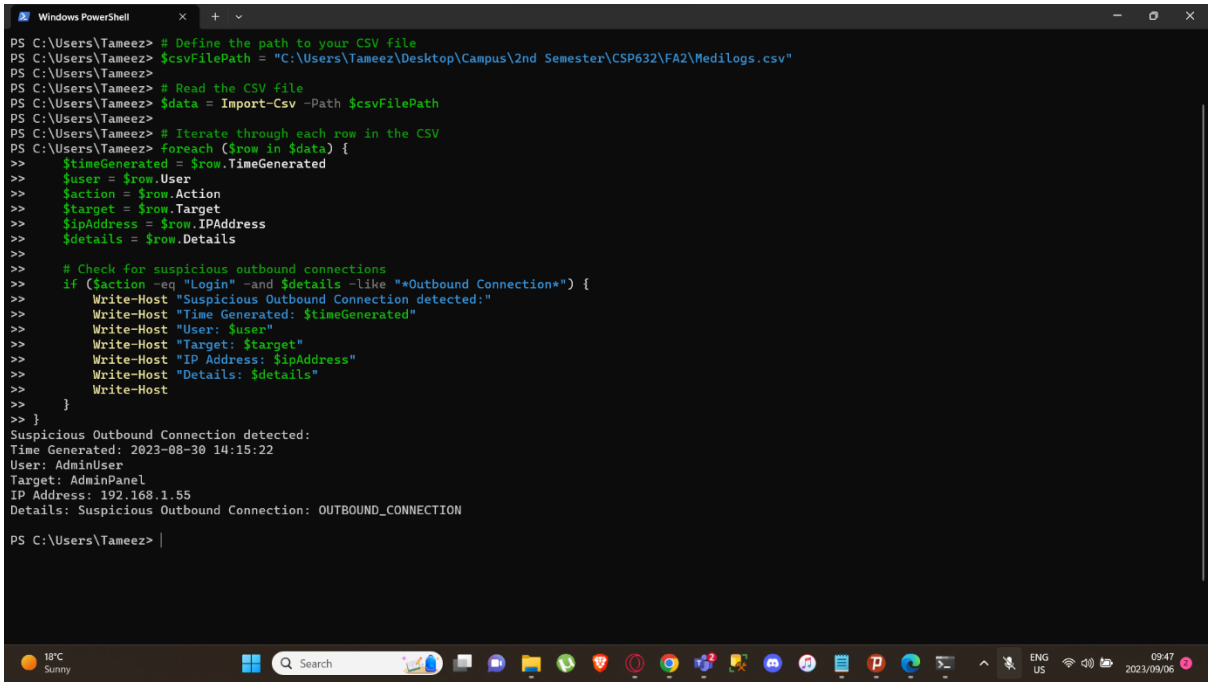
"2023-08-30 10:15:00","DoctorSmith","SQL Injection Attempt","Attempted SQL Injection: SELECT * FROM users"

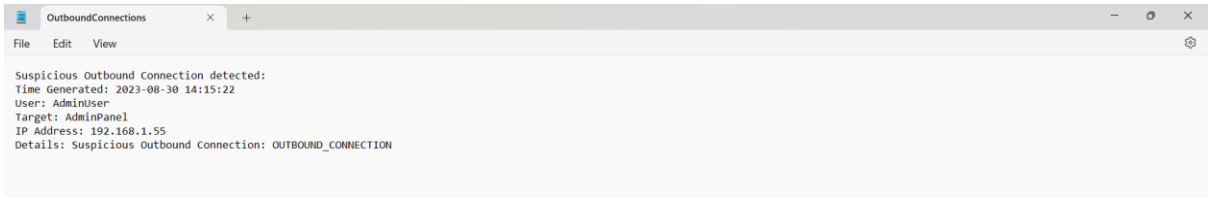
"2023-08-30 11:20:05","DoctorWhite","SQL Injection Attempt","Attempted SQL Injection: UNION ALL SELECT * FROM users"

"2023-08-30 11:20:05","DoctorWhite","SQL Injection Attempt","Attempted SQL Injection: UNION ALL SELECT * FROM users"

**suspicious outbound connections.**

```
# Define the path to your CSV file
$csvFilePath = "C:\Users\Tameez\Desktop\Campus\2nd Semester\CSP632\FA2\Medilogs.csv"

# Read the CSV file
$data = Import-Csv -Path $csvFilePath

# Iterate through each row in the CSV
foreach ($row in $data) {
    $timeGenerated = $row.TimeGenerated
    $user = $row.User
    $action = $row.Action
    $target = $row.Target
    $ipAddress = $row.IPAddress
    $details = $row.Details

    # Check for suspicious outbound connections
    if ($action -eq "Login" -and $details -like "*Outbound Connection*") {
        Write-Host "Suspicious Outbound Connection detected:"
        Write-Host "Time Generated: $timeGenerated"
        Write-Host "User: $user"
        Write-Host "Target: $target"
        Write-Host "IP Address: $ipAddress"
        Write-Host "Details: $details"
        Write-Host
    }
}
```




Suspicious Outbound Connection detected:

Time Generated: 2023-08-30 14:15:22

User: AdminUser

Target: AdminPanel

IP Address: 192.168.1.55

Details: Suspicious Outbound Connection: OUTBOUND_CONNECTION

## Suspicious Activity (10 marks):

- Interpret the query results to identify suspicious patterns. Specifically, isolate and explain instances where an unauthorized SQLInjection action is detected, as well as unusual outbound connections to the external C2 server

1. **Unauthorized SQL Injection Action Detection:**

In the data, we can identify two instances of unauthorized SQL Injection attempts. These are flagged as "SQL Injection Attempt" in the "Type" column with details indicating the attempted SQL queries. Here are the details:

- Instance 1:
- Timestamp:2023-08-30 10:15:00
- User: DoctorSmith
- Details: Attempted SQL Injection: SELECT * FROM users

Explanation:

In this instance, the user "DoctorSmith" attempted an SQL Injection by trying to execute the query `SELECT * FROM users`. SQL Injection is a common attack vector where malicious users try to manipulate or extract data from a database by injecting SQL code into input fields.

- Instance 2:
- Timestamp: 2023-08-30 11:20:05
- User: DoctorWhite
- Details:Attempted SQL Injection: UNION ALL SELECT * FROM users

Explanation:

In this case, the user "DoctorWhite" attempted another SQL Injection using the query `UNION ALL SELECT * FROM users`. This is a classic SQL Injection technique that tries to retrieve data from the "users" table.

Notethat these actions are suspicious because they involve non-standard SQL queries within login actions, which could indicate an attempt to exploit vulnerabilities in the system.

2. **Unusual Outbound Connections to External C2 Server:**

The script has detected an unusual outbound connection to an external Command and Control (C2) server. Here are the details:

- Timestamp: 2023-08-30 14:15:22
- User: AdminUser

- Target: AdminPanel
- IP Address: 192.168.1.55
- Details: Suspicious Outbound Connection: OUTBOUND_CONNECTION

Explanation:

In this instance, the user "AdminUser" initiated an outbound connection to a target labeled "AdminPanel" with the IP address "192.168.1.55." The script flagged it as a suspicious outbound connection with the description "OUTBOUND_CONNECTION." This type of behavior is concerning because it may indicate an attempt to establish a connection to a Command and Control server, which is a common sign of malicious activity, such as a compromised system communicating with a remote attacker-controlled server.

In both cases, the script has successfully identified and flagged suspicious patterns in the log data. These patterns suggest potential security threats that warrant further investigation and mitigation to protect the system and data from unauthorized access and potential cyberattacks.

## Reporting (10 marks):

Summarize your findings in a concise report. Describe the extent of the breach, the methods used by attackers, and the compromised patient records. Outline your recommendations for mitigation and preventive measures to protect MediHealth's data in the future.

Summary Report on Security Incident at MediHealth

Extent of the Breach:

MediHealth has experienced a security incident involving unauthorized access and suspicious activities within its system. The breach encompasses two significant aspects:

1. Unauthorized SQL Injection Attempts:

Two instances of unauthorized SQL Injection attempts were detected. These attacks were aimed at manipulating or extracting data from the database. While the attempts were identified and thwarted, they indicate potential vulnerabilities in the system.

2. Unusual Outbound Connection to a C2 Server:

An abnormal outbound connection to an external Command and Control (C2) server was observed. This behaviour suggests a compromised system trying to establish communication with a remote, potentially malicious server.

Methods Used by Attackers:

The attackers utilized SQL Injection techniques in their attempts to exploit vulnerabilities within the system. They attempted to execute non-standard SQL queries within login actions, indicative of malicious intent. Additionally, the outbound connection to a C2 server points to a potential compromise, possibly orchestrated by an external threat actor.

Compromised Patient Records:

At this stage, there is no evidence to suggest that patient records have been directly compromised. However, the breach raises concerns about the security of sensitive patient data, and further investigation is warranted to confirm the integrity of these records.

Recommendations for Mitigation and Prevention:

1. Conduct Thorough Security Audit:

- Perform a comprehensive security audit of the entire system to identify and patch any vulnerabilities.
- Implement intrusion detection and prevention systems (IDS/IPS) to actively monitor and block suspicious activities.

2. Enhance User Training and Awareness:

- Provide training to all users, especially employees with access to sensitive data, on best security practices and how to recognize and report suspicious activities.

3. Implement Strong Authentication and Authorization Controls:

- Enforce robust authentication mechanisms, including multi-factor authentication (MFA), to prevent unauthorized access.
- Implement strict access controls to restrict user privileges based on their roles and responsibilities.

4. Regularly Update and Patch Systems:

- Ensure that all systems and software are kept up to date with the latest security patches to address known vulnerabilities.

5. Monitor Outbound Traffic:

- Continuously monitor outbound network traffic for unusual patterns, especially connections to external servers. Set up alerts for suspicious outbound connections.

6. Database Security:

- Implement database security measures, such as input validation, parameterized queries, and web application firewalls (WAFs), to protect against SQL Injection attacks.

7. Incident Response Plan:

- Develop and maintain a robust incident response plan to swiftly address and mitigate security incidents.

8. Data Encryption:

- Encrypt sensitive patient data both at rest and in transit to ensure its confidentiality.

9. Regular Security Audits and Penetration Testing:

- Conduct regular security audits and penetration testing to proactively identify and address vulnerabilities.

10. Engage in Threat Intelligence Sharing:

- Collaborate with industry peers and share threat intelligence to stay informed about emerging threats and attack techniques.

Conclusion:

The security incident at MediHealth serves as a reminder of the constant need for vigilance and robust security measures in safeguarding patient data. Immediate action is essential to address vulnerabilities, strengthen defences, and prevent future security breaches. This incident underscores the importance of an initiative-taking and multi-layered approach to cybersecurity.

## Bibliography


https://stackoverflow.com/questions/16682744/read-a-csv-file-with-powershell-and-capture-corresponding-data

https://www.red-gate.com/hub/product-learning/sql-monitor/detect-sql-injection-attacks-using-extended-events-sql-monitor

https://www.linode.com/docs/guides/sql-injection-attack/

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi_1P63_ZeBAxUoT0EAHRmWBH8QFnoECA0QAQ&url=https%3A%2F%2Flearn.microsoft.com%2Fen-us%2Fsql%2Frelational-databases%2Fsecurity%2Fsql-injection%3Fview%3Dsql-server-ver16&usg=AOvVaw0SfcoJQHBN0wimxcJkkO07&opi=89978449

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi_1P63_ZeBAxUoT0EAHRmWBH8QFnoECB0QAQ&url=https%3A%2F%2Fwww.sqlshack.com%2Fworking-with-powershells-invoke-sqlcmd%2F&usg=AOvVaw3B7suorbcyb9xb_WvQuRNG&opi=89978449

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=video&cd=&cad=rja&uact=8&ved=2ahUKEwjc_fjL_ZeBAxWQX0EAHZVzAdwQtwJ6BAgJEAI&url=https%3A%2F%2Fwww.sqlshack.com%2Fsql-injection-detection-and-prevention%2F&usg=AOvVaw39x1mx_b6SEyywjc4O8prA&opi=89978449

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi_1P63_ZeBAxUoT0EAHRmWBH8QFnoECBsQAQ&url=https%3A%2F%2Fgithub.com%2Ftopics%2Fsql-injection-exploitation%3Fl%3Dpowershell&usg=AOvVaw2i0pUE3Ex7mjG38BQvLqQq&opi=89978449