Account Lockout Analysis Analytical Rule

IDENTIFYING ANOMALOUS LOCKOUT EVENTS

USING KUSTO QUERY LANGUAGE (KQL)

About This Document

This document showcases the custom solutions developed by Inspira for Microsoft Sentinel. Our team has tailored various enhancements and automation to meet specific business needs, improving the overall security posture of our clients. Each solution was designed to address unique challenges, streamline processes, and enhance threat detection and response capabilities within Microsoft Sentinel.

Through these customizations, we have enabled organizations to better protect their environments, automate manual tasks, and respond more effectively to security incidents. The following sections detail each solution, including their descriptions, business requirements, prerequisites, and final outputs.

Disclaimer

This document contains information considered confidential and proprietary to Inspira Enterprise; Reader is not permitted to divulge this information without explicit permission from respective stake holders who all had signed off on this document This is an internal, restricted-release document. All brand names used herein are trademarks of respective owners. For all clarifications and discussions please contact

* ***Inspira Microsoft Security Practice Team.***

Document Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No | Version Number | Release Date | Updated By, Company Name | Summary of Changes |
| 1 | 1.0 | Sep 2024 | Inspira Enterprise |  |
|  |  |  |  |  |

Contents

[1 Description 4](#_Toc176975368)

[2 Business Requirements 4](#_Toc176975369)

[3 Prerequisites 4](#_Toc176975370)

[4 MITRE ATT&CK 4](#_Toc176975371)

[5 Query Scheduling 5](#_Toc176975372)

[6 Query 5](#_Toc176975373)

[7 Step-by-Step Breakdown of the Query 6](#_Toc176975374)

[8 Summary 8](#_Toc176975375)

[9 Appendix 8](#_Toc176975376)

1. Description

This Kusto Query Language (KQL) script is designed to analyze account lockout events (EventID 4740) over the last 5 working days, focusing on specific working hours (8 PM to 8 AM) in the US/Pacific time zone. The objective is to identify and highlight potential security anomalies where the frequency of lockouts exceeds the normal rate.

1. Business Requirements

* **Reduce False Positives:** By focusing on non-standard working hours and using a rolling average for anomaly detection, the system should minimize false positives and reduce alert fatigue for security teams.
* **Timely Alerts:** The solution must ensure that security teams are alerted to abnormal lockout activities in near real-time, allowing for swift investigation and response.
* **Enhance Security Posture:** The system should help the organization proactively detect and respond to potential security incidents, reducing the risk of unauthorized access.

1. Prerequisites

* An active Azure subscription
* Contributor RBAC role assigned to a resource group
* An active Sentinel workspace
* A Log Analytics workspace linked to Sentinel
* The SecurityEvent table must be populated with relevant log data, including EventID 4740 (Account Lockout events).
* Ensure that at least 30 days of historical log data is available in the SecurityEvent table.
* An alerting mechanism should be set up to notify the security team when the query detects abnormal lockout activities.

1. MITRE ATT&CK

The following are the MITRE ATT&CK tactics and techniques associated with the analytical rule:

**Credential Access**

* T1110 - Brute Force

Account lockouts are often a result of brute force attacks, where an adversary repeatedly attempts to guess a user's password until the account is locked.

**Defense Evasion**

* T1098 - Account Manipulation

Adversaries may intentionally lock user accounts as a means of disrupting normal operations or as part of a broader strategy to evade detection.

**Persistence**

* T1078 – Valid Accounts

An adversary who has successfully obtained valid credentials may attempt to trigger account lockouts to mislead defenders or hinder the user's ability to access their account.

1. Query Scheduling

* Query Frequency: - Run query every 15 minutes
* Query Lookup data: - Lookup data from the last 5 days.

1. Query

let startHour = 20; // Start of the working hours (8 PM)

let endHour = 8; // End of the working hours (8 AM)

let workingDays = 5;

let endTime = now();

let startTime = endTime - 30d; // A sufficiently large range to capture the last 5 working days plus today

// Get all days in the range and filter out weekends

let allDays = range x from startofday(startTime) to endofday(endTime) step 1d

| extend DayOfWeek = dayofweek(x)

| where DayOfWeek != 6d and DayOfWeek != 0d; // Exclude weekends

// Get the last 5 working days including today

let last5WorkingDays = allDays

| sort by x desc

| take workingDays;

// Determine the exact start time for the last 5 working days including today

let workingDaysStart = toscalar(last5WorkingDays | summarize min(x));

let workingDaysEnd = endTime;

// Retrieve and process lockout events

let lockouts =

SecurityEvent

| extend pacific\_dt = datetime\_utc\_to\_local(TimeGenerated, 'US/Pacific') // converting the time zone to US/Pacific Time Zone

| extend DayOfWeek = dayofweek(pacific\_dt)

| where DayOfWeek != 6d and DayOfWeek != 0d // Exclude weekends

| where EventID == 4740 // Account lockout event

| where pacific\_dt between (workingDaysStart .. workingDaysEnd)

| extend Time = format\_datetime(pacific\_dt, "HH")

// Handle the time range that crosses midnight

| extend hourOfDay = datetime\_part("hour", pacific\_dt)

| where (hourOfDay >= startHour or hourOfDay < endHour)

| summarize lockoutCount = count(),TargetUserName = make\_set(TargetUserName) by bin(pacific\_dt, 1d), hourOfDay, Time;

// Calculate the average lockouts per hour over the last 5 working days

let rollingAverageLockoutsPerHour =

lockouts

| summarize avgLockouts = avg(lockoutCount) by hourOfDay,Time;

// Get lockouts by hour for each of the last 5 working days including today

let workingDayLockoutsPerHour =

lockouts

| summarize dailyLockoutCount = sum(lockoutCount),TargetUserName = make\_set(TargetUserName) by bin(pacific\_dt, 1d), hourOfDay, Time

| project pacific\_dt, hourOfDay, dailyLockoutCount, TargetUserName;

// Join with the rolling average lockouts per hour

workingDayLockoutsPerHour

| join kind=inner (rollingAverageLockoutsPerHour) on hourOfDay

| extend Time = strcat(format\_datetime(pacific\_dt, "yyyy-MM-dd"), " ", tostring(Time), ":00")

| project Time, dailyLockoutCount, avgLockouts, TargetUserName

| order by Time asc

| where dailyLockoutCount > avgLockouts \* 1.5

| render timechart with(title="Account Lock Events")

1. Step-by-Step Breakdown of the Query

Constants and Initial Data Retrieval

**1.** **Set Time Range and Working Days Parameters:**

let startHour = 20; // Start of the working hours (8 PM)

let endHour = 8; // End of the working hours (8 AM)

let workingDays = 5;

These constants define the working hours as 8 PM to 8 AM, and specify that the analysis should cover the last 5 working days.

**2. Define the Time Range for Analysis:**

let endTime = now();

let startTime = endTime - 30d; // A sufficiently large range to capture the last 5 working days plus today.

The query sets the time range from 30 days ago until the current time. This range is chosen to ensure that it captures the last 5 working days, including the current day.

Generate and Filter Working Days.

**3.** **Generate All Days in the Specified Range**

let allDays = range x from startofday(startTime) to endofday(endTime) step 1d.

This step creates a sequence of days within the specified time range.

**4.** **Filter Out Weekends:**

| extend DayOfWeek = dayofweek(x)

| where DayOfWeek != 6d and DayOfWeek != 0d; // Exclude weekends.

The query filters out Saturdays (6d) and Sundays (0d) from the list of days, leaving only working days.

**5.** **Select the Last 5 Working Days**

let last5WorkingDays = allDays | sort by x desc | take workingDays;

This step sorts the remaining working days in descending order and selects the most recent 5.

**6.** **Determine Start Time for the Last 5 Working Days**

let workingDaysStart = toscalar(last5WorkingDays | summarize min(x));

let workingDaysEnd = endTime;

The query determines the earliest start time for the last 5 working days and sets the current time as the end time for analysis.

Process and Analyze Lockout Events

**7.** **Retrieve and Process Lockout Events**

let lockouts = SecurityEvent

| extend pacific\_dt = datetime\_utc\_to\_local(TimeGenerated, 'US/Pacific') // converting the time zone to US/Pacific Time Zone

| extend DayOfWeek = dayofweek(pacific\_dt)

| where DayOfWeek != 6d and DayOfWeek != 0d // Exclude weekends

| where EventID == 4740 // Account lockout event

| where pacific\_dt between (workingDaysStart .. workingDaysEnd)

The query retrieves account lockout events (EventID == 4740) from the SecurityEvent table, converting the timestamps to US/Pacific time. It filters out events that occurred on weekends and selects only those within the last 5 working days.

**8.** **Filter Lockouts by Working Hours**

| extend Time = format\_datetime(pacific\_dt, "HH")

| extend hourOfDay = datetime\_part("hour", pacific\_dt)

| where (hourOfDay >= startHour or hourOfDay < endHour)

This step filters the lockout events to include only those that occurred during the defined working hours (8 PM to 8 AM). It accounts for the time range crossing midnight.

**9.** **Summarize Lockouts by Hour and Date**

| summarize lockoutCount = count(), TargetUserName = make\_set(TargetUserName) by bin(pacific\_dt, 1d), hourOfDay, Time;

The query summarizes the lockout events, grouping them by hour and date, and counting the occurrences. It also collects the target usernames into a set.

Calculate Rolling Averages and Compare

**10.** **Calculate Average Lockouts per Hour**

let rollingAverageLockoutsPerHour = lockouts | summarize avgLockouts = avg(lockoutCount) by hourOfDay,Time;

This step calculates the average number of lockouts per hour over the last 5 working days.

**11.** **Summarize Lockouts per Hour for Each Day**

let workingDayLockoutsPerHour = lockouts | summarize dailyLockoutCount = sum(lockoutCount), TargetUserName = make\_set(TargetUserName) by bin(pacific\_dt, 1d), hourOfDay, Time | project pacific\_dt, hourOfDay, dailyLockoutCount, TargetUserName;

The query summarizes the lockouts by hour for each of the last 5 working days, providing a daily count of lockouts.

**12.** **Join with Rolling Averages**

workingDayLockoutsPerHour | join kind=inner (rollingAverageLockoutsPerHour) on hourOfDay

The query joins the daily lockout counts with the rolling average lockouts per hour.

**13.** **Identify Anomalies**

| extend Time = strcat(format\_datetime(pacific\_dt, "yyyy-MM-dd"), " ", tostring(Time), ":00")

| project Time, dailyLockoutCount, avgLockouts, TargetUserName

| order by Time asc

| where dailyLockoutCount > avgLockouts \* 1.5

This step identifies potential anomalies by selecting hours where the number of lockouts exceeds 1.5 times the average. It orders the results by time.

Final Visualization

**14.** **Render the Timechart**

| render timechart with(title="Account Lock Events")

The query ends by rendering a timechart to visually display the account lockout events, highlighting times where the lockout count significantly exceeded the rolling average.

1. Summary

This query monitors account lockout events during specific working hours (8 PM to 8 AM) over the last 5 working days, comparing the current lockouts to a rolling average to identify anomalies. It is best scheduled to run every 15 minutes for timely detection of suspicious activities.

1. Appendix

**GitHub Repository**

For detailed code and additional resources related to the custom solutions presented in this document, please refer to our GitHub repository:

* **GitHub Repository Link**: : [Inspira Custom Sentinel Solutions](https://github.com/sneha-joy/Account-Lockout-Analysis-Custom-Analytical-Rule)

The repository contains all relevant scripts, configuration files, and documentation necessary to deploy and manage the custom solutions described.