Impossible Travel Detection

Analytical Rule

ANALYZING ANOMALOUS SIGN-IN ACTIVITIES

WITH 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](#_Toc176975240)

[2 Business Requirements 4](#_Toc176975241)

[3 Prerequisites 4](#_Toc176975242)

[4 MITRE ATT&CK 4](#_Toc176975243)

[5 Query Scheduling 5](#_Toc176975244)

[6 Query 5](#_Toc176975245)

[7 Step-by-Step Breakdown of The Query 6](#_Toc176975246)

[8 Summary 9](#_Toc176975247)

[9 Appendix 9](#_Toc176975248)

1. Description

Impossible Travel Kusto Query Language (KQL) script is designed to analyze sign-in logs to detect potential anomalous activity by calculating the speed of travel between login locations. It filters out logs at excessively high speeds, which may indicate suspicious behavior.

1. Business Requirements

* **Identify Potential Compromises:** Detect anomalous user sign-ins that may indicate an account compromise, such as logins from geographically distant locations within an unfeasibly short time frame.
* **Enhance Incident Response:** Provide security teams with actionable insights to quickly investigate and respond to potential threats. Detecting impossible travel can help in initiating prompt investigations into compromised accounts.
* **Mitigate Insider and External Threats:** Protect against both internal (e.g., account misuse) and external threats (e.g., credential theft) by flagging suspicious activities that could indicate unauthorized access to corporate resources.

1. Prerequisites

* An active Azure subscription
* Sentinel Contributor RBAC role assigned to a resource group
* An active Sentinel workspace
* A Log Analytics workspace linked to Sentinel
* Azure Active Directory (Azure AD) Sign-In Logs.
* Adequate log retention policies must be in place to store sign-in data for a period sufficient to perform meaningful analysis.
* Determine appropriate threshold values for travel distance (in kilometers) and timeframe (in hours) based on the organization’s risk tolerance and normal user behavior.

1. MITRE ATT&CK

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

**Initial Access**

* T1078 - Valid Accounts

If an account is used from multiple geographically distant locations in a short period, it may indicate that an adversary has obtained valid credentials and is using them from a different location.

**Command and Control**

* T1071 - Application Layer Protocol

Adversaries might use remote access tools that leverage common application layer protocols to hide their activities, which could be indicated by logins from unusual locations.

**Defense Evasion**

* T1036 – Masquerading

An adversary might use techniques to make their activity appear normal, such as using VPNs or proxies to mimic legitimate user access patterns, which could involve impossible travel scenarios.

1. Query Scheduling

* Query Frequency: - Run query every 6 hours
* Query Lookup data: - Lookup data from the last 24 hours.

1. Query

let maxSpeed = 1000;

SigninLogs

| where ResultType == "0"

| extend latitude\_ = todouble(parse\_json(tostring(LocationDetails.geoCoordinates)).latitude)

| extend longitude\_ = todouble(parse\_json(tostring(LocationDetails.geoCoordinates)).longitude)

| extend countryOrRegion = tostring(LocationDetails.countryOrRegion)

| extend state = tostring(LocationDetails.state)

| extend location = strcat(state,' - ', countryOrRegion)

| where location <> ' - '

| extend browser = tostring(DeviceDetail.browser)

| summarize Count=count(),IP=any(IPAddress),Last=max(TimeGenerated) by UserDisplayName, latitude\_, longitude\_, Locations=tostring(location), browser,AppDisplayName, UserPrincipalName, Location,state

| extend coordinates = pack\_array(latitude\_,longitude\_)

| summarize coordinates=any(coordinates),StateCountries=makeset(Locations),Last=max(Last),IP=any(IP),Apps=makeset(AppDisplayName),Browsers=makeset(browser), Locations=makeset(Locations) by UserDisplayName, UserPrincipalName, Location,state

| summarize Coordinates=makeset(coordinates),NumberOfCountries=dcount(Location), Country = make\_set(Location) ,NumberOfState=dcount(state),StateCountries=make\_set(Locations),Timestamps=makeset(Last),IPs=makeset(IP),Apps=makeset(Apps),Browsers=makeset(Browsers) by UserDisplayName, UserPrincipalName

| where NumberOfCountries > 1 or NumberOfState >2

| extend distance = round(geo\_distance\_2points(todouble(Coordinates[1]),todouble(Coordinates[0]),todouble(Coordinates[3]),todouble(Coordinates[2]))/1000,0)

| extend hours = abs(datetime\_diff('hour', todatetime(Timestamps[1]),todatetime(Timestamps[0])))

| where hours > 0

| extend speedKmPerHour = round(distance/hours,0)

| where speedKmPerHour > maxSpeed

| sort by UserPrincipalName asc

| extend Speed = round(0,0)

| extend IPs = replace('[\\[|\\|\\\\|"\\]]','',tostring(IPs))

| extend if\_3\_countries\_or\_3\_state = iff((NumberOfCountries == 3 or NumberOfState == 3), true, false)

| extend if\_4\_countries\_or\_4\_state = iff((NumberOfCountries == 4 or NumberOfState == 4), true, false)

| extend if\_5\_countries\_or\_5\_state = iff((NumberOfCountries == 5 or NumberOfState == 5), true, false)

// if 3 countries or 3 states are present then

| extend distance2 = iff(if\_3\_countries\_or\_3\_state == true, (round(geo\_distance\_2points(todouble(Coordinates[3]),todouble(Coordinates[2]),todouble(Coordinates[5]),todouble(Coordinates[4]))/1000,0)), Speed)

| extend hours2 = iff(if\_3\_countries\_or\_3\_state == true,(abs(datetime\_diff('hour', todatetime(Timestamps[2]),todatetime(Timestamps[1])))),0)

| extend speedKmPerHour2 = iff(if\_3\_countries\_or\_3\_state == true,round(distance2/hours2,0),Speed)

// if 4 countries or 4 states are present then

| extend distance3 = iff(if\_4\_countries\_or\_4\_state == true, (round(geo\_distance\_2points(todouble(Coordinates[5]),todouble(Coordinates[4]),todouble(Coordinates[7]),todouble(Coordinates[6]))/1000,0)), Speed)

| extend hours3 = iff(if\_4\_countries\_or\_4\_state == true,(abs(datetime\_diff('hour', todatetime(Timestamps[3]),todatetime(Timestamps[2])))),0)

| extend speedKmPerHour3 = iff(if\_3\_countries\_or\_3\_state == true,(round(distance3/hours3,0)),Speed)

// if 5 countries or 5 states are present then

| extend distance4 = iff(if\_5\_countries\_or\_5\_state == true, (round(geo\_distance\_2points(todouble(Coordinates[7]),todouble(Coordinates[6]),todouble(Coordinates[9]),todouble(Coordinates[8]))/1000,0)), Speed)

| extend hours4 = iff(if\_5\_countries\_or\_5\_state == true,(abs(datetime\_diff('hour', todatetime(Timestamps[4]),todatetime(Timestamps[3])))),0)

| extend speedKmPerHour4 = iff(if\_5\_countries\_or\_5\_state == true,(round(distance4/hours4,0)),Speed)

// calvulating total time, and distance

| extend ["Total Hours"] = ( hours + hours2 + hours3 + hours4 )

| extend ["Total Distance"] = distance + distance2 + distance3 + distance4

| extend IN\_US = iff(((NumberOfCountries == 1 or NumberOfCountries == 2) and (Country has "IN" and Country has "US")), true , false )

| project-away if\_3\_countries\_or\_3\_state,if\_4\_countries\_or\_4\_state,if\_5\_countries\_or\_5\_state,Speed

1. Step-by-Step Breakdown of The Query

Constants and Initial Data Retrieval

**1.Set Maximum Speed**

let maxSpeed = 1000;

**2.Retrieve Sign-in Logs**

SigninLogs

| where TimeGenerated >= ago(1d)

| where ResultType == "0"

Filters logs from the last day (ago(1d)) where the result type is "0" (indicating successful sign-ins).

Data Extraction and Transformation.

**3.Extract and Parse Geolocation Details**

| extend latitude\_ = todouble(parse\_json(tostring(LocationDetails.geoCoordinates)).latitude)

| extend longitude\_ = todouble(parse\_json(tostring(LocationDetails.geoCoordinates)).longitude)

| extend countryOrRegion = tostring(LocationDetails.countryOrRegion)

| extend state = tostring(LocationDetails.state)

| extend location = strcat(state,' - ', countryOrRegion)

| where location <> ' - '

•Extracts latitude, longitude, country, and state information from the geolocation details.

•Constructs a location string from the state and country.

**4.Extract Browser and Other Details**

| extend browser = tostring(DeviceDetail.browser)

| summarize Count=count(),IP=any(IPAddress),Last=max(TimeGenerated) by UserDisplayName, latitude\_, longitude\_, Locations=tostring(location), browser,AppDisplayName, UserPrincipalName, Location,state

Extracts and summarizes data such as browser type, IP address, last sign-in time, and application display name.

Summarization and Grouping

**5.Summarize by User**

| extend coordinates = pack\_array(latitude\_,longitude\_)

| summarize coordinates=any(coordinates),StateCountries=makeset(Locations),Last=max(Last),IP=any(IP),Apps=makeset(AppDisplayName),Browsers=makeset(browser), Locations=makeset(Locations) by UserDisplayName, UserPrincipalName, Location,state

Summarizes sign-in data for each user, grouping by user display name and principal name.

**6.Calculate Number of Distinct Locations**

| summarize Coordinates=makeset(coordinates),NumberOfCountries=dcount(Location), Country = make\_set(Location) ,NumberOfState=dcount(state),StateCountries=make\_set(Locations),Timestamps=makeset(Last),IPs=makeset(IP),Apps=makeset(Apps),Browsers=makeset(Browsers) by UserDisplayName, UserPrincipalName

Calculates the number of distinct countries and states a user has signed in from.

Filtering and Distance Calculation

**7.Filter on Number of Countries/States**

| where NumberOfCountries >= 1 and NumberOfState >= 1

Filters users who have logged in from at least one country and one state.

**8.Calculate Distance and Time Differences**

| extend distance = round(geo\_distance\_2points(todouble(Coordinates[1]),todouble(Coordinates[0]),todouble(Coordinates[3]),todouble(Coordinates[2]))/1000,0)

| extend hours = abs(datetime\_diff('hour', todatetime(Timestamps[1]),todatetime(Timestamps[0])))

| where hours > 0

| extend speedKmPerHour = round(distance/hours,0)

| where speedKmPerHour > maxSpeed

Calculates the distance between the first two coordinates and the time difference between their timestamps.

Filters out entries where the calculated speed exceeds maxSpeed.

Sorting and Speed Calculation for Additional Locations

**9.Sort and Extend Calculations for Additional Locations**

| sort by UserPrincipalName asc

| extend Speed = round(0,0)

| extend IPs = replace('[\\[|\\|\\\\|"\\]]','',tostring(IPs))

| extend if\_3\_countries\_or\_3\_state = iff((NumberOfCountries >= 3 or NumberOfState >= 3), true, false)

| extend if\_4\_countries\_or\_4\_state = iff((NumberOfCountries >= 4 or NumberOfState >= 4), true, false)

| extend if\_5\_countries\_or\_5\_state = iff((NumberOfCountries >= 5 or NumberOfState >= 5), true, false)

Extends the query to check if the user has logged in from three, four, or five different countries or states.

**10.Calculate Speeds for Additional Locations**

| extend distance2 = iff(if\_3\_countries\_or\_3\_state == true, (round(geo\_distance\_2points(todouble(Coordinates[3]),todouble(Coordinates[2]),todouble(Coordinates[5]),todouble(Coordinates[4]))/1000,0)), Speed)

| extend hours2 = iff(if\_3\_countries\_or\_3\_state == true,(abs(datetime\_diff('hour', todatetime(Timestamps[2]),todatetime(Timestamps[1])))),0)

| extend speedKmPerHour2 = iff(if\_3\_countries\_or\_3\_state == true,round(distance2/hours2,0),Speed)

| extend distance3 = iff(if\_4\_countries\_or\_4\_state == true, (round(geo\_distance\_2points(todouble(Coordinates[5]),todouble(Coordinates[4]),todouble(Coordinates[7]),todouble(Coordinates[6]))/1000,0)), Speed)

| extend hours3 = iff(if\_4\_countries\_or\_4\_state == true,(abs(datetime\_diff('hour', todatetime(Timestamps[3]),todatetime(Timestamps[2])))),0)

| extend speedKmPerHour3 = iff(if\_3\_countries\_or\_3\_state == true,(round(distance3/hours3,0)),Speed)

| extend distance4 = iff(if\_5\_countries\_or\_5\_state == true, (round(geo\_distance\_2points(todouble(Coordinates[7]),todouble(Coordinates[6]),todouble(Coordinates[9]),todouble(Coordinates[8]))/1000,0)), Speed)

| extend hours4 = iff(if\_5\_countries\_or\_5\_state == true,(abs(datetime\_diff('hour', todatetime(Timestamps[4]),todatetime(Timestamps[3])))),0)

| extend speedKmPerHour4 = iff(if\_5\_countries\_or\_5\_state == true,(round(distance4/hours4,0)),Speed)

Calculates distances, times, and speeds for additional sign-in locations if the user has logged in from multiple countries or states.

Summarizing Total Distance and Time

**11.Calculate Total Time and Distance**

| extend ["Total Hours"] = ( hours + hours2 + hours3 + hours4 )

| extend ["Total Distance"] = distance + distance2 + distance3 + distance4

Calculates the total travel time and distance for each user.

Excluding Certain Countries

**12.Exclude Specific Country Combinations**

| extend IN\_US\_and\_US\_PH = iff((( NumberOfCountries == 2) and ((Country has "IN" and Country has "US") or( Country has "US" and Country has "PH" ))), true , false )

| extend within\_US = iff(NumberOfCountries == 1 and Country has "US", true, false)

| where IN\_US\_and\_US\_PH == false

| where within\_US == false

Excludes users who logged in from specific combinations of countries (India and US, or US and Philippines), or only within the US.

Final Projection

**13.Project Final Output**

| project-away if\_3\_countries\_or\_3\_state,if\_4\_countries\_or\_4\_state,if\_5\_countries\_or\_5\_state,Speed, IN\_US\_and\_US\_PH, within\_US, Country

Projects the final set of fields excluding intermediate calculation fields.

1. Summary

This query is useful for identifying potential security risks by detecting impossible travel speeds between sign-in locations.

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/Impossible-Travel-Custom-Query)

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