|  |
| --- |
| # Get API key from here: https://ipgeolocation.io/ |
|  |  | $API\_KEY = "d4600b4efdef42b39828f5155041a457" |
|  |  | $LOGFILE\_NAME = "failed\_rdp.log" |
|  |  | $LOGFILE\_PATH = "C:\ProgramData\$($LOGFILE\_NAME)" |
|  |  |  |
|  |  | # This filter will be used to filter failed RDP events from Windows Event Viewer |
|  |  | $XMLFilter = @' |
|  |  | <QueryList> |
|  |  | <Query Id="0" Path="Security"> |
|  |  | <Select Path="Security"> |
|  |  | \*[System[(EventID='4625')]] |
|  |  | </Select> |
|  |  | </Query> |
|  |  | </QueryList> |
|  |  | '@ |
|  |  |  |
|  |  | <# |
|  |  | This function creates a bunch of sample log files that will be used to train the |
|  |  | Extract feature in Log Analytics workspace. If you don't have enough log files to |
|  |  | "train" it, it will fail to extract certain fields for some reason -\_-. |
|  |  | We can avoid including these fake records on our map by filtering out all logs with |
|  |  | a destination host of "samplehost" |
|  |  | #> |
|  |  | Function write-Sample-Log() { |
|  |  | "latitude:47.91542,longitude:-120.60306,destinationhost:samplehost,username:fakeuser,sourcehost:24.16.97.222,state:Washington,country:United States,label:United States - 24.16.97.222,timestamp:2021-10-26 03:28:29" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:-22.90906,longitude:-47.06455,destinationhost:samplehost,username:lnwbaq,sourcehost:20.195.228.49,state:Sao Paulo,country:Brazil,label:Brazil - 20.195.228.49,timestamp:2021-10-26 05:46:20" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:52.37022,longitude:4.89517,destinationhost:samplehost,username:CSNYDER,sourcehost:89.248.165.74,state:North Holland,country:Netherlands,label:Netherlands - 89.248.165.74,timestamp:2021-10-26 06:12:56" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:40.71455,longitude:-74.00714,destinationhost:samplehost,username:ADMINISTRATOR,sourcehost:72.45.247.218,state:New York,country:United States,label:United States - 72.45.247.218,timestamp:2021-10-26 10:44:07" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:33.99762,longitude:-6.84737,destinationhost:samplehost,username:AZUREUSER,sourcehost:102.50.242.216,state:Rabat-Salé-Kénitra,country:Morocco,label:Morocco - 102.50.242.216,timestamp:2021-10-26 11:03:13" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:-5.32558,longitude:100.28595,destinationhost:samplehost,username:Test,sourcehost:42.1.62.34,state:Penang,country:Malaysia,label:Malaysia - 42.1.62.34,timestamp:2021-10-26 11:04:45" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:41.05722,longitude:28.84926,destinationhost:samplehost,username:AZUREUSER,sourcehost:176.235.196.111,state:Istanbul,country:Turkey,label:Turkey - 176.235.196.111,timestamp:2021-10-26 11:50:47" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:55.87925,longitude:37.54691,destinationhost:samplehost,username:Test,sourcehost:87.251.67.98,state:null,country:Russia,label:Russia - 87.251.67.98,timestamp:2021-10-26 12:13:45" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:52.37018,longitude:4.87324,destinationhost:samplehost,username:AZUREUSER,sourcehost:20.86.161.127,state:North Holland,country:Netherlands,label:Netherlands - 20.86.161.127,timestamp:2021-10-26 12:33:46" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:17.49163,longitude:-88.18704,destinationhost:samplehost,username:Test,sourcehost:45.227.254.8,state:null,country:Belize,label:Belize - 45.227.254.8,timestamp:2021-10-26 13:13:25" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:-55.88802,longitude:37.65136,destinationhost:samplehost,username:Test,sourcehost:94.232.47.130,state:Central Federal District,country:Russia,label:Russia - 94.232.47.130,timestamp:2021-10-26 14:25:33" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | } |
|  |  |  |
|  |  | # This block of code will create the log file if it doesn't already exist |
|  |  | if ((Test-Path $LOGFILE\_PATH) -eq $false) { |
|  |  | New-Item -ItemType File -Path $LOGFILE\_PATH |
|  |  | write-Sample-Log |
|  |  | } |
|  |  |  |
|  |  | # Infinite Loop that keeps checking the Event Viewer logs. |
|  |  | while ($true) |
|  |  | { |
|  |  |  |
|  |  | Start-Sleep -Seconds 1 |
|  |  | # This retrieves events from Windows EVent Viewer based on the filter |
|  |  | $events = Get-WinEvent -FilterXml $XMLFilter -ErrorAction SilentlyContinue |
|  |  | if ($Error) { |
|  |  | #Write-Host "No Failed Logons found. Re-run script when a login has failed." |
|  |  | } |
|  |  |  |
|  |  | # Step through each event collected, get geolocation |
|  |  | # for the IP Address, and add new events to the custom log |
|  |  | foreach ($event in $events) { |
|  |  |  |
|  |  |  |
|  |  | # $event.properties[19] is the source IP address of the failed logon |
|  |  | # This if-statement will proceed if the IP address exists (>= 5 is arbitrary, just saying if it's not empty) |
|  |  | if ($event.properties[19].Value.Length -ge 5) { |
|  |  |  |
|  |  | # Pick out fields from the event. These will be inserted into our new custom log |
|  |  | $timestamp = $event.TimeCreated |
|  |  | $year = $event.TimeCreated.Year |
|  |  |  |
|  |  | $month = $event.TimeCreated.Month |
|  |  | if ("$($event.TimeCreated.Month)".Length -eq 1) { |
|  |  | $month = "0$($event.TimeCreated.Month)" |
|  |  | } |
|  |  |  |
|  |  | $day = $event.TimeCreated.Day |
|  |  | if ("$($event.TimeCreated.Day)".Length -eq 1) { |
|  |  | $day = "0$($event.TimeCreated.Day)" |
|  |  | } |
|  |  |  |
|  |  | $hour = $event.TimeCreated.Hour |
|  |  | if ("$($event.TimeCreated.Hour)".Length -eq 1) { |
|  |  | $hour = "0$($event.TimeCreated.Hour)" |
|  |  | } |
|  |  |  |
|  |  | $minute = $event.TimeCreated.Minute |
|  |  | if ("$($event.TimeCreated.Minute)".Length -eq 1) { |
|  |  | $minute = "0$($event.TimeCreated.Minute)" |
|  |  | } |
|  |  |  |
|  |  |  |
|  |  | $second = $event.TimeCreated.Second |
|  |  | if ("$($event.TimeCreated.Second)".Length -eq 1) { |
|  |  | $second = "0$($event.TimeCreated.Second)" |
|  |  | } |
|  |  |  |
|  |  | $timestamp = "$($year)-$($month)-$($day) $($hour):$($minute):$($second)" |
|  |  | $eventId = $event.Id |
|  |  | $destinationHost = $event.MachineName# Workstation Name (Destination) |
|  |  | $username = $event.properties[5].Value # Account Name (Attempted Logon) |
|  |  | $sourceHost = $event.properties[11].Value # Workstation Name (Source) |
|  |  | $sourceIp = $event.properties[19].Value # IP Address |
|  |  |  |
|  |  |  |
|  |  | # Get the current contents of the Log file! |
|  |  | $log\_contents = Get-Content -Path $LOGFILE\_PATH |
|  |  |  |
|  |  | # Do not write to the log file if the log already exists. |
|  |  | if (-Not ($log\_contents -match "$($timestamp)") -or ($log\_contents.Length -eq 0)) { |
|  |  |  |
|  |  | # Announce the gathering of geolocation data and pause for a second as to not rate-limit the API |
|  |  | #Write-Host "Getting Latitude and Longitude from IP Address and writing to log" -ForegroundColor Yellow -BackgroundColor Black |
|  |  | Start-Sleep -Seconds 1 |
|  |  |  |
|  |  | # Make web request to the geolocation API |
|  |  | # For more info: https://ipgeolocation.io/documentation/ip-geolocation-api.html |
|  |  | $API\_ENDPOINT = "https://api.ipgeolocation.io/ipgeo?apiKey=$($API\_KEY)&ip=$($sourceIp)" |
|  |  | $response = Invoke-WebRequest -UseBasicParsing -Uri $API\_ENDPOINT |
|  |  |  |
|  |  | # Pull Data from the API response, and store them in variables |
|  |  | $responseData = $response.Content | ConvertFrom-Json |
|  |  | $latitude = $responseData.latitude |
|  |  | $longitude = $responseData.longitude |
|  |  | $state\_prov = $responseData.state\_prov |
|  |  | if ($state\_prov -eq "") { $state\_prov = "null" } |
|  |  | $country = $responseData.country\_name |
|  |  | if ($country -eq "") {$country -eq "null"} |
|  |  |  |
|  |  | # Write all gathered data to the custom log file. It will look something like this: |
|  |  | # |
|  |  | "latitude:$($latitude),longitude:$($longitude),destinationhost:$($destinationHost),username:$($username),sourcehost:$($sourceIp),state:$($state\_prov),label:$($country) - $($sourceIp),timestamp:$($timestamp)" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  | "latitude:$($latitude),longitude:$($longitude),destinationhost:$($destinationHost),username:$($username),sourcehost:$($sourceIp),state:$($state\_prov), country:$($country),label:$($country) - $($sourceIp),timestamp:$($timestamp)" | Out-File $LOGFILE\_PATH -Append -Encoding utf8 |
|  |  |  |
|  |  | Write-Host -BackgroundColor Black -ForegroundColor Magenta "latitude:$($latitude),longitude:$($longitude),destinationhost:$($destinationHost),username:$($username),sourcehost:$($sourceIp),state:$($state\_prov),label:$($country) - $($sourceIp),timestamp:$($timestamp)" |
|  |  | } |
|  |  | else { |
|  |  | # Entry already exists in custom log file. Do nothing, optionally, remove the # from the line below for output |
|  |  | # Write-Host "Event already exists in the custom log. Skipping." -ForegroundColor Gray -BackgroundColor Black |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  | } |