



Blue Teaming with KQL – 2022 KQL|Café Edition

Ashwin Patil, @[ashwinpatil](https://twitter.com/ashwinpatil)
Senior Security Researcher, Microsoft



GitHub: <https://github.com/ashwin-patil/blue-teaming-with-kql>

Agenda

- 2020 Gray Hat Talk Recap
- Mastered KQL syntax. Next Steps ?
- Practical Detection Engineering/Hunting with KQL
- Extending KQL
- Conclusion

GrayHat 2020 Talk Recap

- Why learn query language ?
- Intro to KQL
- Structure of Basic KQL Query
- KQL Basic Searches
- Exploring Tables, Schemas
- Asset/Device Details
- Query Parameterization
- Dynamic datatypes
- Datetime
- Regex Extraction
- Functions (User-defined ,Built-in)
- Externaldata
- Time Series Analysis
- Threat Hunting Use-Cases
 - Time Series Anomaly
 - Network Beacons
- KQL Programmatic Interfaces
 - Msticpy query provider
- KQL Playground, Trainings, Resources
- Conclusion

[Slides and Video: GrayHat- BlueTeamingwithKQL.pdf · ashwin-patil/blue-teaming-with-kql \(github.com\)](#)

Mastered KQL Syntax. Next Steps ?

- ❑ Most of the real world KQL detection queries are **much larger and complex**.
- ❑ Effective query writing can still be **overwhelming for defenders** who are new to KQL and Tables.
- ❑ Analytical questions do not translate directly to language syntax and **requires prior preparation of the data** - data wrangling with interjoining tables , parsing and extracting etc.

Practical Detection Engineering/Hunting with KQL

Practical real-world use-cases:

- summarize, bin: Simple aggregation and threshold-based query
- Multi-Joins: Bringing context from other data sources
- Leftanti: Rare events – Not historically seen events
- Pivot: To create heatmap like data structure to identify hourly spikes and more

Simple aggregation and threshold-based detection

Example: [Azure-Sentinel/Suspicious enumeration using adfind.yaml \(github.com\)](#)

- Looking for suspicious command line tokens crossing threshold (3) within lookup window (2 min)

```
1 let lookupwindow = 2m;
2 let threshold = 3; //number of commandlines in the set below
3 let DCADFServersList = dynamic (["DCServer01", "DCServer02", "ADFSserver01"]); // Enter a reference list of hostnames for your DC/ADFS servers
4 let tokens = dynamic(["objectcategory","domainlist","dcmodes","adinfo",
5 "trustdmp","computers_pwdnotreqd","Domain Admins", "objectcategory=person", "objectcategory=computer", "objectcategory=*"]);
6 SecurityEvent
7 //| where Computer in (DCADFServersList) // Uncomment to limit it to your DC/ADFS servers list if specified above or any pattern in hostnames (startswith, matches regex, etc).
8 | where EventID == 4688
9 | where CommandLine has_any (tokens)
10 | where CommandLine matches regex "(.*)>(.*)"
11 | summarize Commandlines = make_set(CommandLine), LastObserved=max(TimeGenerated) by bin(TimeGenerated, lookupwindow), Account, Computer, ParentProcessName, NewProcessName
12 | extend Count = array_length(Commandlines)
13 | where Count > threshold
```

```
13 | where Count > threshold
14 | extend Count = array_length(Commandlines)
15 | where Count > threshold
```

Bringing Context from other data sources

Example: [Azure-Sentinel/AADPrivilegedAccountsFailedMFA.yaml](#) ([github.com](#))

- Looking for failed MFA attempts from **Privileged accounts**. Privileged account list dynamically built using IdentityInfo UEBA tables

```
1 let starttime = 2d;
2 let endtime = 1d;
3 let aadFunc = (tableName:string){
4 IdentityInfo
5 | where AssignedRoles contains "Admin"
6 | mv-expand AssignedRoles
7 | extend Roles = tostring(AssignedRoles), AccountUPN = tolower(AccountUPN)
8 | where Roles contains "Admin"
9 | distinct Roles, AccountUPN
10 | join kind=inner (
11 // Failed Signins attempts with reasoning related to MFA.
12 table(tableName)
13 | where TimeGenerated between(starttime..endtime)
14 | where ResultDescription has_any ("MFA", "second factor", "multi-factor", "second factor") or ResultType in (50074, 50076, 50079, 50072, 53004, 500121)
15 ) on $left.AccountUPN == $right.UserPrincipalName
16 | extend timestamp = TimeGenerated, IPCustomEntity = IPAddress, AccountCustomEntity = UserPrincipalName
17 };
18 let aadSignin = aadFunc("SigninLogs");
19 let aadNonInt = aadFunc("AADNonInteractiveUserSignInLogs");
20 union isfuzzy=true aadSignin, aadNonInt
```

```
10 let aadSignin = aadFunc("SigninLogs");
11 let aadNonInt = aadFunc("AADNonInteractiveUserSignInLogs");
12 union isfuzzy=true aadSignin, aadNonInt
```

Rare events – Not historically seen events

Example: [Azure-Sentinel/PaloAlto-UnusualThreatSignatures.yaml \(github.com\)](https://github.com/Azure-Sentinel/PaloAlto-UnusualThreatSignatures.yaml)

- Looking for PAN threat sigs from unusual IP addresses which are not historically seen.

```
let starttime = 7d;
let endtime = 1d;
let timeframe = 1h;
let HistThreshold = 25;
let CurrThreshold = 10;
let HistoricalThreats = CommonSecurityLog
| where isnotempty(SourceIP)
| where TimeGenerated between (startofday(ago(starttime))..startofday(ago(endtime)))
| where DeviceVendor =~ "Palo Alto Networks"
| where Activity =~ "THREAT" and SimplifiedDeviceAction =~ "alert"
| where DeviceEventClassID in ('spyware', 'scan', 'file', 'vulnerability', 'flood', 'packet', 'virus', 'wildfire', 'wildfire-virus')
| summarize TotalEvents = count(), ThreatTypes = make_set(DeviceEventClassID), DestinationIpList = make_set(DestinationIP), FirstSeen = min(TimeGenerated), LastSeen = max(TimeGenerated) by SourceIP, DeviceAction, DeviceVendor;
let CurrentHourThreats = CommonSecurityLog
| where isnotempty(SourceIP)
| where TimeGenerated > ago(timeframe)
| where DeviceVendor =~ "Palo Alto Networks"
| where Activity =~ "THREAT" and SimplifiedDeviceAction =~ "alert"
| where DeviceEventClassID in ('spyware', 'scan', 'file', 'vulnerability', 'flood', 'packet', 'virus', 'wildfire', 'wildfire-virus')
| summarize TotalEvents = count(), ThreatTypes = make_set(DeviceEventClassID), DestinationIpList = make_set(DestinationIP), FirstSeen = min(TimeGenerated), LastSeen = max(TimeGenerated) by SourceIP, DeviceAction, DeviceProduct, DeviceVendor;
CurrentHourThreats
| where TotalEvents < CurrThreshold
| join kind = leftanti (HistoricalThreats
| where TotalEvents > HistThreshold) on SourceIP
| where TotalEvents > HistThreshold
```


Pivot – Create Heatmaps

Use case: Create weekly heatmap of failed logon per hour and identify spikes or plot heatmaps by exporting output. Pivot will create output of Days by Hour – count of failed logon and project-reorder with [granny-asc](#) will sort columns named with hours.

```
let end = datetime(2022-08-30T00:00:00Z);
let start = end - 7d;
SecurityEvent
| where EventID == 4625
| where TimeGenerated >= startofday(start)
| where TimeGenerated <= startofday(end)
| extend HourOfLogin = toint(hourofday(TimeGenerated)), DayNumberofWeek = dayofweek(TimeGenerated) ,
Date = format_datetime(TimeGenerated, "yyyy-MM-dd")
| extend DayofWeek = case( DayNumberofWeek == "00:00:00", "Sunday",
                            DayNumberofWeek == "1.00:00:00", "Monday",
                            DayNumberofWeek == "2.00:00:00", "Tuesday",
                            DayNumberofWeek == "3.00:00:00", "Wednesday",
                            DayNumberofWeek == "4.00:00:00", "Thursday",
                            DayNumberofWeek == "5.00:00:00", "Friday",
                            DayNumberofWeek == "6.00:00:00", "Saturday", "InvalidTimeStamp")
| evaluate pivot(HourOfLogin, count(), DayofWeek, Date)
| project-reorder Date, DayofWeek, * granny-asc
| sort by Date asc
```

```
sort by Date asc
project-reorder Date, DayofWeek, * granny-asc
```

Extending KQL – GitHub Action for dynamic TI feeds.

Externaldata :

- Allows to connect external data sources.
- Limited to static sites or blob storage data sources.

Use case : [Azure-Sentinel/Signins-from-NordVPN-Providers.yaml \(github.com\)](https://github.com/microsoft/mstic/blob/master/nordvpn-servers.csv)

Signins from Nord VPN Providers. Nord VPN API provides unauthenticated feed but is not accessible from externaldata. Solution create external feed via GitHub Actions.

```
let nord_vpn_feed = (externaldata(id:int,ip_address: string,search_keywords: dynamic,categories:dynamic,name: string,domain:string,price:int,flag:string,
country:string,location:dynamic ,load: int ,features:dynamic)
```

```
["@https://raw.githubusercontent.com/microsoft/mstic/master/nordvpn-servers.csv"] with (format="csv", ignoreFirstRecord=True));
```

SigninLogs

```
| where TimeGenerated > ago(4h)
```

```
| where ResultType == 0
```

```
| summarize TotalEvents = count(), AppList = make_set(AppDisplayName), StartTime = min(TimeGenerated), EndTime = max(TimeGenerated) by
IPAddress, UserPrincipalName, ClientAppUsed, ConditionalAccessStatus, AuthenticationRequirement, RiskDetail
```

```
| join kind= inner nord_vpn_feed on $left.IPAddress == $right.ip_address
```

```
| project StartTime , EndTime, IPAddress, UserPrincipalName, AppList, ClientAppUsed, ConditionalAccessStatus, AuthenticationRequirement,
RiskDetail, categories, domain, country
```

```
| extend timestamp = StartTime, AccountCustomEntity = UserPrincipalName, IPCustomEntity = IPAddress
```

```
| extend timestamp = StartTime, AccountCustomEntity = UserPrincipalName, IPCustomEntity = IPAddress
```

```
timestamp, categories, domain, country
```

Extending KQL – Github Action for dynamic TI feeds.

```
GitHub Workflow (github-workflow.json) | You, 3 months ago | 1 author (You)
1 name: Daily Nord VPN Servers Feed
2
3 on:
4   workflow_dispatch:
5   schedule:
6     - cron: "0 0 * * *" # Runs at 00:00 AM (UTC) everyday (Check https://crontab.tech/)
7
8 jobs:
9   download-nordvpnservers:
10    name: Download Nord VPN Servers via API
11    runs-on: ubuntu-latest
12    steps:
13      - name: Checkout the repo
14        uses: actions/checkout@v3
15        with:
16          path: master
17          fetch-depth: 0
18
19      - name: Set up Python 3.8
20        uses: actions/setup-python@v3
21        with:
22          python-version: "3.8"
23          architecture: "x64"
24
25      - name: Install dependencies
26        run: |
27          python -m pip install --upgrade pip
28          pip install requests pandas
29
30      - name: Run automation script
31        run: python master/.script/get-nordvpnservers.py You, 3 months ago • update
32
33      - name: Commit files
34        run: |
35          cd master
36          git config --local user.email "41898282+github-actions[bot]@users.noreply.github.com"
37          git config --local user.name "github-actions[bot]"
38          git add *
39          git commit -m "Adding new nord vpn server daily feed" -a
40
41      - name: Push changes
42        uses: ad-m/github-push-action@master
43        with:
44          directory: "master"
45
```

```
get-nordvpnservers.py
1 import requests
2 import json
3 import sys
4 import logging
5 import pandas as pd
6 from pathlib import Path
7
8 "https://api.nordvpn.com/server"
9
10
11 def download_nord_vpn_servers(url, output_file):
12     # Download data from the url
13     r = requests.get(url)
14
15     # Decode byte array into string
16     my_json = r.content.decode("utf8")
17
18     # Convert string to JSON
19     data = json.loads(my_json)
20
21     # Load list of JSON records into dataframe
22     df = pd.DataFrame(data)
23     # Convert to csv
24     df.to_csv(output_file, index=False)
25
26
27 def main():
28     logging.basicConfig(
29         stream=sys.stdout,
30         level=logging.DEBUG,
31         format="%(asctime)s: %(levelname)s: %(message)s",
32     )
33
34     api_url = "https://api.nordvpn.com/server"
35     logging.info("Python main function started")
36     logging.info(f"Downloading Nord VPN server list using API from {api_url}")
37
38     curr_path = Path.cwd()
39     out_path = (
40         curr_path / "master" / "PublicFeeds" / "NordVPNDaily" / "nordvpn-servers.csv"
41     )
42     try:
43         out_path.parents[0].mkdir(parents=True, exist_ok=False)
44     except FileExistsError:
45         logging.info("Folder is already present")
46     else:
47         logging.info(f"{out_path} Folder was created")
48
49     logging.info(f"Writing csv file to output directory : {out_path}")
50     download_nord_vpn_servers(api_url, out_path)
51
52 if __name__ == "__main__":
53     main()
```

✓ Daily Nord VPN Servers Feed Daily Nord VPN Servers Feed #162

Summary

Jobs

✓ Download Nord VPN Servers via API

Download Nord VPN Servers via API

succeeded 10 hours ago in 24s

> ✓ Set up job

> ✓ Checkout the repo

> ✓ Set up Python 3.8

> ✓ Install dependencies

> ✓ Run automation script

> ✓ Commit files

> ✓ Push changes

> ✓ Post Set up Python 3.8

> ✓ Post Checkout the repo

> ✓ Complete job

Extending KQL – ADX/LA Interoperability.

KQL has varying support in Azure Data Explorer (ADX) and Azure Log Analytics(LA)/Sentinel.

- You can connect both products from each other and can run native KQL against it.
- Connect additional data sources without duplicating data.
- Use Kusto explorer client with rich features on LA data.
- Extend support of missing KQL operators in LA/Sentinel.

Connect ADX via LA: [Cross-resource query Azure Data Explorer by using Azure Monitor - Azure Monitor | Microsoft Docs](#)

Connect LA via ADX: [Query data in Azure Monitor with Azure Data Explorer | Microsoft Docs](#)

Conclusion

- Once you mastered KQL language syntax, *get familiar with well-known detection methods* in KQL.
- *Practice and understand complex scenarios through Azure-Sentinel and Community Projects* (#365DaysofKQL, #50DaysofKQLforIntune)
- *Share your KQL queries and other innovative hunting methods* with the community.



Thank you