FALCON 302

CROWDSTRIKE QUERY LANGUAGE

VERSION 02.2024

CROWDSTRIKE

CROWDSTRIKE UNIVERSITY

TABLE OF CONTENTS

CROWDSTRIKE

OVERVIEW

The following queries from CrowdStrike's Query Language (CQL) are provided as takeaway material for the Falcon 202 instructor-led training. This guide includes some of the simple queries we performed in class as well as a collection of useful queries to get you started with hunting in your own environment.

These queries can be used to fit your environment in the following manner.

Change use case:

Modify "**FileName**" or other parameters. Many of these queries are designed to show a concept, however, they can be easily modified to use in production.

Filter:

Include "**AND NOT**" or "**!=**" (not equal to) statements to filter out unwanted events.

Change scope:

Search across smaller datasets by using search modifiers like:

- Reducing the scope of the search:
  - aid=Some_AgentID
  - ComputerName=Some_Computer
  - LocalAddressIP4=Some_LocalIP
- Reducing the search timeframe:
  - you may reduce the date range of the search to recue the total events searched

KEY FIELDS AND THINGS TO REMEMBER

Query	Description
@id, @timestamp, @timezone, @rawstring	Metadata Fields • FieldNames starting with an @ • Contains the metadata attached to events on ingestion
#repo, #type, #hosts, #source	Tag Fields • FieldNames starting with a # • Defines how events are stored and indexed • Used to speed up queries
[ComputerName, "Host Name"]	Arrays • Use the [ ] syntax to use an Array • Listed members of the array are separated by commas, similar to JSON
| groupBy(ComputerName, function={avgStartTime:=avg(ProcessStartTime)})	Functions • Use the { } syntax to use a Function • Allows for further calculations or parsing of referenced event data
CommandLine="*Program Files\\Some_Folder*"	Meta Characters • These characters have multiple functions within our queries • Escape meta-characters in your syntax with a “\”
#event_simpleName	The name of the event
aid	The sensor ID. This value is unique to each installation of a Falcon sensor. When a sensor is updated or reinstalled, the host gets a new aid. In those situations, a single host could have multiple aid values over time.
#cid	Customer ID (only one in most cases)
@timestamp	• Timestamp when the event was received by the CrowdStrike cloud • Not to be confused with the time the event was generated locally on the system (context time)
| EventTime:=**formatTime ("%Y-%m-%dT%H:%M:%S.%L", field=timestamp, timezone="Zulu")**	Command to convert a timestamp field to a human readable UTC format

KEY FIELDS AND THINGS TO REMEMBER CONTINUED

Query	Description
ComputerName	The name of the host
ContextTimeStamp	• The time at which an event occurred on the system, as seen by the sensor (in decimal, non-hex format). Not to be confused with timestamp which is the time the event was received by the cloud. • Applies to non-process events like DnsRequest, NetworkConnectIP4, ImageHash, AsepValueUpdate, etc. (must be converted)
ProcessStartTime	• The time the process began in UNIX epoch time (in decimal, non-hex format) • Applies only to ProcessRollup2 and SyntheticProcessRollup2 (must be converted)
DeviceTimeStamp	• Timestamp when the event occurred on the endpoint • Applies to Device Control events
TargetProcessID	• The unique ID of a target process (in decimal, non-hex format) • This field exists in almost all events, and it represents the ID of the process that is responsible for the activity of the event in focus
Parent_ProcessId	Field that identifies the TargetProcessId of the parent process
ContextProcessId	The unique ID of an event that was spawned by another process (in decimal, non-hex format) that identifies the TargetProcessId of the responsible process
RpcClientProcessId	Field in an RPC related event that identifies the TargetProcessId for the responsible process

SAMPLE LANGUAGE SNIPPETS

Query	Description
ContextTime := formatTime("%Y/%m/%d %H:%M:%S", field=ContextTimeStmp, locale=en_US, timezone=Z)	Convert ContextTimeStamp to readable UTC format (new field called ContextTime)
ProcessTime := **formatTime**("%Y/%m/%d %H:%M:%S", field=ProcessTimeStamp, locale=en_US, timezone=Zulu)	Convert ProcessStartTime to readable UTC format (new field called ProcessTime)
formattime("%A %d %B %Y, %R", as=fmttime, field=@timestamp, timezone=PST)	Format time as Thursday 18 November 2021, 22:59 using US locale and PST timezone using the ”as” parameter to fmttime
groupBy(ComputerName) | top(ComputerName, limit=10)	GroupBy ComputerName field and limit the output to the top 10 results.
| **rename**([[ComputerName, "Host Name"], [@timestamp,"Time"]])	Rename one or more fields, Comma separated Array [Brackets] for multiple fields. Use quotes for spaces. Can affect later syntax in the query
| **sort**([_count], order=desc, limit=10)	Sort on a [field] Order = **desc** is descending and **asc** is ascending Limit to the top 10 entries
| **FileName**=?FileSearchField Syntax: <FieldName> =? <Input field to create>	? Parameter function creates an entry field for the user to enter the filename without needing to modify the Advanced Event Search

SAMPLE LANGUAGE SNIPPETS CONTINUED

Query	Description
| join ({#event_simpleName=AgentOnline}, field=[aid], include=[aid, ComputerName])	Used to **join** two data sets together using common fields from both sets
| groupBy (ComputerName, **function=[ ])**	Remove duplicate values of a specific field with the function= [ ]
| iplocation aip	Perform geolocation on the AIP field for the events and display Country, City, Region (state), Lat/Long, etc.
CommandLine!="*Program Files\\Some_Folder*"	Omitting entries found in a FieldName
| case **{ TreeId = *** **| EventDetails:= "Detections Found";** ***;** **}**	Case Expressions allow you to write if-then-else constructs that work on events streams

BASIC PROCESS QUERIES FROM CLASS

Query	Description
#event_simpleName=ProcessRollup2 OR SyntheticProcessRollup2 FileName=cmd.exe	Simple Process Search (modify the filename for your needs)
#event_simpleName=ProcessRollup2 OR SyntheticProcessRollup2 FileName=cmd.exe **| table**([FileName, ComputerName, CommandLine])	Table command outputs specified fields in tabular format
#event_simpleName=ProcessRollup2 OR SyntheticProcessRollup2 FileName=cmd.exe **| groupBy**([ComputerName, CommandLine])	groupBy command provides a count of events specified field(s)

SUB-QUERIES AND JOINS

Query	Description
#event_simpleName=DnsRequest | **in**("DomainName", values=["falconsplayingpoker.com", "winteriscoming.com"]) | **join**({#event_simpleName=ProcessRollup2}, field=[ContextProcessId], key=TargetProcessId, include=[FileName, UserName, ImageFileName, CommandLine], mode=left) | **table**([@timestamp, ComputerName, UserName, DomainName, FileName, ImageFileName, CommandLine])	Joining the ProcessRollup2 event data to connect the DnsRequest records with their associated processes.
#event_simpleName=DnsRequest | **in**("DomainName", values=["falconsplayingpoker.com", "winteriscoming.com"]) | **join**({#event_simpleName=ProcessRollup2}, field=[ContextProcessId], key=TargetProcessId, include=[FileName, UserName, ImageFileName, CommandLine], mode=left) | **table**([@timestamp, ComputerName, UserName, DomainName, FileName, ImageFileName, CommandLine])	Show a responsible process from a "Context Process" item such as DnsRequest, NetworkConnectIP4, etc.
#event_simpleName=NetworkConnectIP4 | **!cidr**(RemoteAddressIP4, subnet=["10.0.0.0/8", "127.0.0.0/8"]) | **join**({ProcessRollup2}, field=[ContextProcessId], key=TargetProcessId, include=[FileName, UserName, ImageFileName, RemoteAddressIP4, RemotePort, CommandLine], mode=left) | **groupBy**(UserName, function=collect([FileName, UserName, ImageFileName, RemoteIP, RPort, CommandLine])) | **sort**([_count], order=asc)	Rare external connections per user
#event_simpleName=NetworkConnectIP4 |**in**(field="RemotePort", values=[137, 139, 389, 3389, 445]) | **join**({#event_simpleName=ProcessRollup2}, field=[ContextProcessId], key=TargetProcessId, include=[UserName, ComputerName, ImageFileName, RemoteAddressIP4, RemotePort], mode=left) | **groupBy**(ComputerName, function=collect([FileName, UserName, ImageFileName, RemoteAddressIP4, RemotePort])) | **sort**([_count], order=asc)	Rare internal connections (common ports)

OTHER QUERIES OF NOTE

<table>
<tr><th>Query</th><th>Description</th></tr>
<tr><td><b>#event_simpleName</b>=DcUsbDeviceConnected</td><td>USB Device Insertions</td></tr>
<tr><td><b>#event_simpleName</b>=DcUsbDeviceConnected
<b>| DeviceTimeStamp :=</b> parseTimeStamp(field=DeviceTimeStamp, format=seconds)
<b>| "Time Inserted" :=</b> formatTime("%Y-%m-%dT%H:%M:%S.%L", field=DeviceTimeStamp, timezone="Zulu")
<b>| rename</b>([[ComputerName,"Host Name"], [DevicePropertyClassName,"Connection Type"], [DeviceManufacturer,Manufacturer],[DeviceProduct, "Product Name"], [DevicePropertyDeviceDescription, Description], [DevicePropertyClassGuid,GUID], [DeviceInstanceId,"Device ID"]])
<b>| groupBy</b>([aid, "Device ID"], function=([collect(["Time Inserted", ComputerName, "Connection Type", Manufacturer, "Product Name", Description, GUID])]))</td><td>USB Device Insertions with time conversion on the timestamp field and renaming of fields</td></tr>
<tr><td><b>(#event_simpleName=</b>DcUsbDeviceConnected) OR <b>(#event_simpleName=</b>*FileWritten IsOnRemovableDisk=1)
<b>| DeviceId:=</b>DeviceInstanceId
<b>| DeviceId:=</b>DiskParentDeviceInstanceId
<b>| selfJoinFilter</b>(field=[DeviceId, aid], where=[{*FileWritten OR #event_simpleName=DcUsbDeviceConnected}, {IsOnRemovableDisk=1}])
<b>| ContextTimeStamp :=</b> parseTimeStamp(field=ContextTimeStamp, format=seconds)
<b>| ReadableEventTime :=</b> formatTime("%Y-%m-%dT%H:%M:%S.%L", field=ContextTimeStamp, timezone="Zulu")
<b>| groupBy</b>([aid, DeviceId], function=([collect([ComputerName, DevicePropertyClassName, DeviceManufacturer, DeviceProduct, ReadableEventTime, ContextTimeStamp, FileName])]))</td><td>Show files written to removable USB disk</td></tr>
<tr><td><b>#event_simpleName=</b>RemovableMediaVolumeMounted
<b>| table</b>([ComputerName, VolumeDriveLetter, VolumeFileSystemDevice, VolumeFileSystemDriver])</td><td>Show volume mounts for removable media</td></tr>
</table>

OTHER QUERIES OF NOTE CONTINUED

Query	Description
#event_simpleName=ProcessRollup2 ImageFileName!="\\Device\\HarddiskVolume*" **| rename** ([[ComputerName,"Endpoint"],[FileName,"File"],[FilePath,"Path"],[SHA256HashData, "SHA256"]]) **| groupBy**([aid, Endpoint], function=[collect([File, Path, SHA256])])	Show processes being run from a removable USB drive
ComputerName= SomeComputer RpcClientProcessId=* **| groupBy**([#event_simpleName])	List Enhanced Attacker Execution Profiling (EAEP) events on specified host
ServiceStarted HostedServiceStarted ServiceStopped HostedServiceStopped FirewallSetRule UserAccountCreated FirewallDeleteRule UserAccountDeleted FirewallChangeOption UserAccountAddedToGroup ScheduledTaskDeleted ScheduledTaskRegistered Sample Query: **#event_simpleName=**UserAccountCreated	Available EAEP event types
ComputerName=DRAGONSTONE **#event_simpleName=**"DnsRequest" **| groupBy**([DomainName]) **| sort**([_count], order=desc) **| top**(_count, limit=10)	Top 10 domain requests
ComputerName=DRAGONSTONE CommandLine!="" **| table**([UserName, FileName, ComputerName, CommandLine, @timestamp], sortby=CommandLine, limit=20000)	Table command line activity for a host and sort by CommandLine field
#event_simpleName=ProcessRollup2 FileName=cmd.exe **| groupBy**(ParentBaseFileName, function=collect([ComputerName, UserName, DomainName, FileName, ImageFileName, CommandLine]))	Show all Parents of a process (Example: cmd.exe)
#event_simpleName=ProcessRollup2 ParentBaseFileName=winlogon.exe **| groupBy**([FileName, CommandLine])	Show all children of a process (Example: winlogon.exe)

GENERAL QUERIES

<table>
<tr><th>Query</th><th>Description</th></tr>
<tr><td>**ComputerName=**DRAGONSTONE
#event_simpleName=ProcessRollup2
| in(CommandLine, values=["*\\arp.exe*", "at *", "*\\at.exe*", "bitsadmin *", "*\\bitsadmin.exe*", "*csvde.exe *", "dsquery *", "*\\dsquery.exe*", "*\\ftp.exe*", "*\\makecab.exe*", "*nbtstat *", "net *", "*\\net.exe*", "*\\net1.exe*", "netsh *", "*\\netsh.exe*", "netstat *", "*\\netstat.exe*", "*nslookup*", "ping *", "*\\ping.exe*", "*quser.exe*", "*\\reg.exe*", "*\\regsvr32.exe*", "route *", "*\\route.exe*", "sc *", "*\\sc.exe*", "schtasks *", "*\\schtasks.exe*", "systeminfo*", "taskkill *", "*\\taskkill.exe*", "tasklist *", "*\\tasklist.exe*", "*wevtutil*", "whoami*", "*\\whoami.exe*", "*\\xcopy.exe", "wmic *", "*\\wmic.exe*", "psexec *", "*\\psexec.exe*", "*\\psexesvc.exe*", "powershell.exe *", "*\\powershell.exe*", "*\\cmd.exe*", "cmd *", "cmd.exe*"])
| !in(field="CommandLine", values=["*powershell.exe*\\CCM\\*", "\"C:\\Windows\\system32\\SearchFilterHost.exe\"*", "\"C:\Windows\system32\SearchProtocolHost.exe\"*", "\"C:\\Windows\\system32\\makecab.exe\" C:\\Windows\\Logs\\CBS\\*", "\"C:\\Windows\\CCM\\*", "\\??\\C:\\Windows\\system32\\conhost.exe \"*", "rundll32 C:\\Windows\\system32\\spool\\DRIVERS\\*", "taskeng.exe \{*", "microsoft monitoring agent", "system center operations manager"])
| groupBy([UserName, ComputerName], function=([collect([CommandLine])]))
| sort([_count], order=desc)</td><td>Stack command line activity for common attacker binaries</td></tr>
</table>

RESOURCE

This resource contains information about how to hunt using Falcon and is tailored specifically to users running the Falcon sensor on Windows hosts. However, many of the ideas and concepts also apply to users running the Falcon sensor on Mac or Linux. Depending on the sensor platform, however, the names and descriptions of certain events as well as custom query syntax will vary. We recommend that you read and refer to the Events Data Dictionary to learn more about specific events, the fields within those events, and the variations in events across platforms.

CrowdStrike Query Language Hunting Queries