Dive into DSL: Digital Response Analysis with Elasticsearch

Brian Marks and Andrea Sancho Silgado

June 23, 2016
http://kenjohnson.mydagsite.com/home
Assumptions

- Elasticsearch deployed
- Ingested Forensic artifact(s) data into Elasticsearch with the “field: values”
  - Logstash
  - Elasticsearch API (bulk insert)
  - Third party application
- You have installed a python library
  $ pip install elasticsearch_dsl
Elasticsearch DSL

- Libraries available: Python, Ruby, Java, .NET, PHP
- DSL Objects: Query, Filter, Aggregation, Mapping, Search
- Ability to combine Query objects with Boolean operators
  - &: AND
  - |: OR
  - ~: NOT
- DSL objects can be de-serialized to a python dictionary
- Supports various types of queries (we’ll focus on query_string)
  - Field: value
  - Full featured search
  - Kibana search syntax
Sample Query

from elasticsearch_dsl import Q

q = Q("query_string",
    query="_type:prefetch")

print q.to_dict()
from elasticsearch_dsl import Q

q0 = Q("query_string",
    query="_type:prefetch")
q1 = Q("query_string",
    query="_type:amcache")

q = q0 | q1

print q.to_dict()
Sample Query 3.0

```python
from elasticsearch_dsl import Q

q0 = Q("query_string", query="_type:prefetch")
q1 = Q("query_string", query="_type:amcache")
q2 = ~Q("query_string", query="path:*\.dll")

q = q0 | (q1 & q2)

print q.to_dict()
```
“Signature of Forensics”
"Signature of Forensics"

• **Features of Apache Lucene syntax**\(^1\)
  - Boolean operators – this AND that OR NOT some other thing
  - Wildcard queries – “*.evtx” matches files with an event log extension
  - Phrase queries – “this is my secret text file”
  - Proximity queries – “secret file”~1 matches the phrase above
  - Fuzzy queries – svchost.exe~ matches scvhost.exe
  - Range queries – EventID:[4624 TO 4648]
  - Regular expression - /.*Program Files( \(x86\))?.*/ matches either folder

• **Search multiple artifacts and data types**
  - Combine queries objects
  - Boolean operators: & (AND), | (OR), ~ (NOT)

• **Convert queries to Kibana search syntax**

Aggregations
Aggregations

• Analytics on a search
• Build buckets based on matching criteria
• Types of aggregations
  • Terms
  • Date histogram
  • Metrics: (extended_stats)
  • Geo Bounds
  • And more…
• Best part: Nesting Aggregations!
Aggregations: How to?

1. Create query
   ```python
   from elasticsearch_dsl import Q
   q = Q("query_string", query="_type:mft")
   ```

2. Create aggregation (top level bucket)
   ```python
   from elasticsearch_dsl import A
   a = A("terms", field="extension")
   ```

3. Add to Search object
   ```python
   from elasticsearch_dsl import Search
   s = Search(using=es, index="my_index")
   s.query(q)
   s.aggs("extensions", a)
   ```

4. Execute
   ```python
   result = s.execute()
   ```
How many different files are on the file system per extension?
Single Aggregation: Accessing the result

```python
for item in result.aggregations.extensions.buckets:
    print item.key, item.doc_count
```

Name of the top level aggregation

```
htm 120833
png 96346
dll 83216
docx 94
...```

MFT entries
How many files are created on a daily basis per file extension?
Nesting Aggregations

```python
from elasticsearch_dsl import Search, Q, A
q = Q("query_string", query="_type:mft")

a = A('date_histogram', field='create_date', interval='day',
    format='yyyy-MM-dd')

a.bucket('extensions', 'terms', field='extension')

s = s.query(q)
s.aggs("dates_mft", a)

result = s.execute()
```
Nesting Aggregations

Date: 2015-12-31, Items created: 1
File Extensions: ["zip"= 1]

Date: 2016-01-01, Items created : 79
File Extensions: ["dll"= 57] [""= 11]
["exe"= 2] ["ad3"= 1] ["ad1"= 1]
["sys"= 1] ["ad5"= 1] ["ad2"= 1]
["ad4"= 1] ["mem"= 1] ["pdf"= 1]

Date: 2016-01-02, Items created : 19
File Extensions: ["log"= 9] ["etl"= 8]
["wer"= 1] [" "= 1]

Date: 2016-01-03, Items created : 73
File Extensions: [" "= 21] ["etl"= 19]
How many users logged into the system per day?
Which user accounts?
How did they log in?
Nesting Nested Aggregations

\[
q = Q("query_string",
    query="\_type:winevt_logs AND event\_ID:4624")
\]

\[
a = A('date\_histogram', field='event\_datetime',
    interval='day', format='yyyy-MM-dd')
\]

\[
a.bucket("users", "terms", field="username").
    bucket("logon", "terms", field="logon\_type")
\]

\[
s = s.query(q)
\]

\[
s.aggs("logons\_dates", a)
\]

\[
result = s.execute()
\]
Nesting Nested Aggregations (the result)

2012-03-29  Total Logons:18
User Name: wks-win732bits, Logons:17
  Logon Type 3, Count: 17
User Name: system, Logons:1
  Logon Type 5, Count: 1

2012-03-30  Total Logons:25
User Name: wks-win732bits, Logons:19
  Logon Type 3, Count: 19
User Name: system, Logons:4
  Logon Type 5, Count: 4
User Name: tdungan, Logons:2
  Logon Type 2, Count: 2

2012-03-31  Total Logons:30
User Name: wks-win732bits, Logons:20
  Logon Type 3, Count: 20
User Name: system, Logons:5
  Logon Type 5, Count: 5
User Name: nromanoff, Logons:3
  Logon Type 10, Count: 1
User Name: tdungan, Logons:2
  Logon Type 2, Count: 2

...
Aggregations in Kibana
So what does this all mean?
Thank you!

Questions?

@brianDFIR
@andreasanchos
Appendix: Accessing Nested Aggregations

for item in result.aggregations.dates_mft.buckets:
    print "Date: ", item.key_as_string,
    print "Items Created: ", item.doc_count
    print "File Extensions:",

    for ext in item.extensions.buckets:
        print "[%s = %d] " %
        (ext.key, ext.doc_count)