Hunting for Malicious Infrastructure Using Big data

Shadi Al-Hakimi
Freek Bax
University of Amsterdam

Jop van der Lelie
Daniel Sierat
NCSC
Content

- Introduction
- Problem statement
- AI and Machine learning
- Implementation/Data gathering
- Results/discussion
- Conclusion
- Future work
Introduction

- Malicious infrastructure: Malicious Command and Control Web Servers such as (Cobalt strike).
- They try to hide their messages in HTTP because it is not blocked by firewalls and it is hard to analyze all HTTP traffic.
- Fingerprinting is one of the techniques used to identify malicious C&C servers.
- Most research on anomaly detection focused on outgoing and incoming HTTP traffic of a specific host or network. (needs active beacon)
- Passive fingerprinting of HTTP GET root (/) responses could identify active C&C servers before beacon deployment.
Problem statement

How can machine learning be used on a data set of HTTP responses to identify malicious webservers?

- What features can we extract from the HTTP responses? (Empirical Approach)
- Which machine learning algorithm are best suited for anomaly detection on a data set of HTTP responses? (Theoretical analysis)
AI and Machine learning

- Supervised versus unsupervised learning
- Creating a labeled data set
- Clustering algorithms
Feature extraction

- Transforming text
- Natural language processing

- Header ordering

Server: Varnish
Retry-After: 0
content-type: text/html
Cache-Control: private, no-cache
X-Served-By: cache-pao17442-PAO
Content-Length: 247
Accept-Ranges: bytes
Date: Mon, 11 Jan 2021 05:03:15 GMT
Via: 1.1 varnish
Connection: close
Uniqueness

- Counting the frequency of all header fields
- Computing the uniqueness for each header field
- Uniqueness of a header = 1 - frequency header / total responses

- For each HTTP response computing their uniqueness
- Adding the uniqueness value of all the headers present
- Adding 1 - uniqueness for all headers absent
Evaluation

- Looking at the distribution of the features
- Finding anomalies with the features
Implementation

- Dataset: Part of project sonar from Rapid7 specifically IPv4 space scan of port 80 HTTP GET root (/).
- Technologies: Dask a python library for parallel processing of large-scale datasets.
- Hardware: Group (cluster) of 3 machines (160GiB RAM, 60 core CPU)
- Data processing:
  - Preprocessing.
  - Dask processing.
- Uniqueness Algorithm
- Header order Algorithm
Implementation: Data processing ... con’t 2

- Preprocessing
  - Each raw of the original dataset contain the ip and the base64 of the response.
  - We wrote a python script to decode the base64 encoding into json object
    
    \{'ip': '<IPv4adress>', 'headers': '<HTTPHeaders>'\}

- Dask processing
  - We changed data file format from json to parquet.
  - Loaded the data into Dask.
  - Proper partitioning of the data is really important!
  - Started to run queries. \( df1 = df[\text{'headers'}].str.extractall(\'(\n[\s\-]*:[^\s\-]*:\n[\s\-]*\')\) \)
## Uniqueness Algorithm

For each response in dataset:

For each header_name in uniqueness_values:

If header_name inside response:

Add uniqueness_value[header_name]

Else:

Add (1.0 - uniqueness_value[header_name])

For each header_name in response:

If header_name not in uniqueness_values:

Add 0.99

## Header order Algorithm

For each response in dataset:

`row = get_index.header_names() # [160,12,30,190,0]`

`row = row.sort_values() # [0,12,30,160,190]`

For value in row.index:

If index > -1:  # header_name not exists

    result.append(value)

response['order'] = result  # [2,1,3,4,5]
Results
Ordering

- Not distributed evenly
- Majority of orderings uncommon

<table>
<thead>
<tr>
<th>% the data set</th>
<th>Number of unique orderings</th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>4</td>
</tr>
<tr>
<td>75%</td>
<td>10</td>
</tr>
<tr>
<td>90%</td>
<td>32</td>
</tr>
<tr>
<td>95%</td>
<td>55</td>
</tr>
<tr>
<td>99%</td>
<td>87</td>
</tr>
<tr>
<td>99.9%</td>
<td>168</td>
</tr>
<tr>
<td>99.99%</td>
<td>230</td>
</tr>
<tr>
<td>100%</td>
<td>301</td>
</tr>
</tbody>
</table>
## Ordering

<table>
<thead>
<tr>
<th>Ordering</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server, Content-type, Content-length, Date, Connection</td>
<td>9824687</td>
</tr>
<tr>
<td>Server, Date, Content-length, Content-type, Connection</td>
<td>8099211</td>
</tr>
<tr>
<td>Date, Server, Content-length, Connection, Content-type</td>
<td>7875328</td>
</tr>
<tr>
<td>Server, Date, Content-type, Connection</td>
<td>4528994</td>
</tr>
<tr>
<td>Content-type, Server, Date, Connection, Content-length</td>
<td>3612256</td>
</tr>
<tr>
<td>Date, Content-type, Content-length, Connection, Server</td>
<td>1469608</td>
</tr>
<tr>
<td>Date, Server, Content-length, Content-type, Connection</td>
<td>1239995</td>
</tr>
<tr>
<td>Date, Server, Connection, Content-type</td>
<td>1041208</td>
</tr>
<tr>
<td>Date, Server, Connection, Content-length Content-type</td>
<td>980881</td>
</tr>
<tr>
<td>Date, Content-type, Content-length, Connection</td>
<td>571911</td>
</tr>
</tbody>
</table>
## Ordering

<table>
<thead>
<tr>
<th></th>
<th>Connection</th>
<th>Date</th>
<th>Content-type</th>
<th>Server</th>
<th>Content-length</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>4.62%</td>
<td></td>
<td>25.49%</td>
<td>7.68%</td>
<td>17.97%</td>
<td>13.93739275%</td>
</tr>
<tr>
<td>Date</td>
<td>95.38%</td>
<td></td>
<td>61.66%</td>
<td>35.74%</td>
<td>67.55%</td>
<td>65.08130996%</td>
</tr>
<tr>
<td>Content-type</td>
<td>74.51%</td>
<td>38.34%</td>
<td>18.13%</td>
<td>70.49%</td>
<td>50.36916664%</td>
<td>50.36916664%</td>
</tr>
<tr>
<td>Server</td>
<td>92.32%</td>
<td>64.26%</td>
<td>81.87%</td>
<td></td>
<td>91.09%</td>
<td>82.38651886%</td>
</tr>
<tr>
<td>Content-length</td>
<td>82.03%</td>
<td>32.45%</td>
<td>29.51%</td>
<td>8.90%</td>
<td></td>
<td>38.22561181%</td>
</tr>
</tbody>
</table>
Conclusion

- Both uniqueness and ordering feature are promising
- Clustering looks best suited to use on HTTP responses
Future research

- Using the features with machine learning
- Comparing different clustering algorithms
- Other features
Questions?

https://gitlab.com/shadialhakimi/rp1

Thank you