Copyright 2021 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.


DM21-0830
Motivation - Background & Collaboration

- Dustin and Tom are Cyber Security researchers at CERT/SEI/CMU
  - Architect and conduct realistic cyber warfare scenarios - high fidelity cyber range

- Collaboration - DoD Threat Hunters
  - Tuning TTPs for finding compromises in your network
  - How to best detect network beacons?

- Threat Hunter TTPs
  - Network data from Zeek / Suricata - signature based
  - SIEM tools / ELK / Log file aggregation

- Challenges:
  - Beacon detection is not suited for signature based TTPs
What are Network Beacons?

- What is a “network beacon”?
  - Network events that reoccur on a timing interval
  - Essentially a heartbeat signal

- Legitimate
  - WiFi
  - Obtain instructions from an API / telemetry data

- Malware
  - Provides mechanism – command & control (C2)
  - Calls home looking for new instructions
Beacon Characteristics

- Patterns in connectivity
  - Predictable timing between connection requests
  - Often similar small packet sizes
  - Connection characteristics that may fall outside of a network baseline / jitter

- Anomaly Detection with Ease?
Beacon Challenges

- Multiple protocol types - HTTP / HTTPS / DNS
- Cloud migrations - data availability

- Encryption for all network communications - TLS
  - less metadata for additional investigations

- Patterns take time - Patience!
  - Intervals over hours to days
  - Window of consideration
  - Data overload
Beacon Complexity

- Smart adversaries
  - Use jitter/dispersion to vary beacon time intervals, payload sizes, etc.
  - FQDN round robins
  - Still needs to be “functional” malware -- limits avoidance techniques

- Malware beacon traffic intermixed with other adversary C2 traffic
  - Impacts pattern detection

- “False positive central”
  - Legitimate software will exhibit beacon-like behavior
  - White list maintenance
The Great Equalizer...

“The network levels the playing field … everything needs to talk on the network…”

- Chris Brenton - activecountermeasures.com

- if malware is on your network — it WILL need to communicate out to the public Internet to be successful of it's intentions:

  1. Checking in with Command and Control (C2) to communicate “next steps”

  1. Providing internal network access routes for lateral movement

  2. Data Exfiltration
Beacon Detection Goals

- Threat Hunters / Analysts can't realistically look at every network connection

- Anomaly Detection
  - Finding the "most" interesting things
    - Score/create a list and investigate
    - A place to start
Beacon Detection History

- CMU CERT
  - Over a decade of Beacon Detection research/strategies
    - Centered around network flow data
    - Sorting/Filtering/aggregating network flow data
      - “help find interesting things”
- Others:
  - Large network vendors
    - Enable “Beacon detection algorithms”
  - Commercial SIEM & Tools vendors and OSS efforts
    - Sorting/Filtering/Scoring
    - Based on Zeek (Bro) datasets
Data Similarities

- Netflow versus Zeek (Bro)
  - Core data is the same
    - Small subset
      - src/dst addys, ports, protocols, timestamps, pkt lengths
  - Additional metadata
    - DNS
    - Creating metadata
      - Timings between connections/flows
        - "delta times"
### Delta Times Example

<table>
<thead>
<tr>
<th>connection_id</th>
<th>sip</th>
<th>dip</th>
<th>port</th>
<th>proto</th>
<th>datetime</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>7393</td>
<td>192.168.31.22</td>
<td>143.235.14.130</td>
<td>443</td>
<td>tcp</td>
<td>2021-08-25 10:57:46.160573959</td>
<td>5.024812</td>
</tr>
<tr>
<td>7393</td>
<td>192.168.31.22</td>
<td>143.235.14.130</td>
<td>443</td>
<td>tcp</td>
<td>2021-08-25 11:02:47.482317924</td>
<td>5.022029</td>
</tr>
<tr>
<td>7393</td>
<td>192.168.31.22</td>
<td>143.235.14.130</td>
<td>443</td>
<td>tcp</td>
<td>2021-08-25 11:12:50.197284937</td>
<td>5.022323</td>
</tr>
<tr>
<td>7393</td>
<td>192.168.31.22</td>
<td>143.235.14.130</td>
<td>443</td>
<td>tcp</td>
<td>2021-08-25 11:17:51.547010899</td>
<td>5.022495</td>
</tr>
<tr>
<td>7393</td>
<td>192.168.31.22</td>
<td>143.235.14.130</td>
<td>443</td>
<td>tcp</td>
<td>2021-08-25 12:43:33.702056885</td>
<td>75.655883</td>
</tr>
<tr>
<td>7393</td>
<td>192.168.31.22</td>
<td>143.235.14.130</td>
<td>443</td>
<td>tcp</td>
<td>2021-08-25 12:48:34.889444113</td>
<td>5.019790</td>
</tr>
<tr>
<td>7393</td>
<td>192.168.31.22</td>
<td>143.235.14.130</td>
<td>443</td>
<td>tcp</td>
<td>2021-08-25 12:58:37.702542067</td>
<td>5.024801</td>
</tr>
</tbody>
</table>
Current Techniques for Finding Beacons

● “Top Talker” reports
  ○ High frequency beacons will show on these hourly/daily reports

● Primitive Clustering
  ○ consistent sequences - based on threshold
    vs.

💥 Setting threshold high will miss beacons & low will increase false positives
Techniques for Finding Beacons

- **Standard Deviation** - How far does the data differ from the average (mean)?
  - low spread (lower standard deviation) → beacon
  - coefficient of variance = relative standard deviation
    - As % - measures closeness of data to the average value (mean)
  - Identify thresholds
    - “Score” the result & generate list high probability targets
    - Provides more data points
Challenges of Existing Solutions

- "Black-boxed"
  - Limited options for tool / data tuning - “enable” checkbox
    - “Input the data - ok, here you go”
    - list of suspects / scores
  - OSS solutions often are no longer maintained
    - undocumented algorithms
Challenges of Existing Solutions

- Beacon suspects
  - Existing solutions - will help find beacons in many cases
  - How was the conclusion reached?
    - Not easy to follow - trust us
  - Threat Hunters / analysts
    - likely not a statistics expert
  - Few beacons are alike - approach varies on each instance
Guiding Principles...

1. Help analysts level-up & understand details of the data
   - Jupyter Notebooks - annotated analysis techniques for guidance
   - Provide documented options — analyst can’t always afford $ vendor solutions
   - Easier (& faster) data filtering & transformation

2. Remove “black box” where possible
   - “Unlock” standard Python libraries — scikit-learn / numpy
   - Remove DIY for clustering or standard deviation
   - Assume analyst is smart - capable of following the data flow

3. Leverage benefits of newer techniques in data analysis
   - Unsupervised Machine Learning - Clustering
4 Project Goals

A. Fast & scalable data management
B. Viable cluster analysis
C. Comparisons of disparate time spans
D. Automation
Key #1: Data

1. Log files to intermediate “delta” dataset:
   - [connection_id {sip, dip, port, protocol}, deltas]

2. Filtering:
   - External traffic only
   - Remove very short delta times (< 8 seconds)
     - Visible in a top talkers report anyway
   - Remove connections that are not “cluster ready”
     - Connection sets < 5 members for a time_span

3. Capture & focus on common protocols first (http|s)

4. 88% reduction in file size (4.9MB intermediate for a 43.6MB log file)
   and more opportunity to continue to reduce storage sizes
<table>
<thead>
<tr>
<th>connection_id</th>
<th>sip</th>
<th>dip</th>
<th>port</th>
<th>proto</th>
<th>datetime</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>192.168.31.21</td>
<td>172.217.9.200</td>
<td>443</td>
<td>tcp</td>
<td>2021-08-25 13:24:35.651804924</td>
<td>5.046017</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connection_id</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018</td>
<td>10.784478</td>
</tr>
<tr>
<td>2018</td>
<td>5.046017</td>
</tr>
<tr>
<td>2018</td>
<td>10.105615</td>
</tr>
<tr>
<td>2018</td>
<td>10.088304</td>
</tr>
<tr>
<td>2018</td>
<td>10.3064</td>
</tr>
</tbody>
</table>
Key #2: Clusters Identify Potential Beacons

1. KMEANS
   - ✅ Easy to automate (input values like n_clusters are calculable)
   - ✅ Clusters all data by default
   - ⚥💥 Does not filter outliers (1 datapoint can cluster)
   - ⚥💥 As a result, additional logic required to ID high likelihood IPs

1. DBSCAN
   - ✅ Automatically removes outliers
   - ⚥EPS can be tricky
   - 🌰 Overlapping spans of minutes (0-5, 3-15, 12-24, etc.) mitigates EPS calc

1. KBINS, HDDBSCAN, OPTICS, pycluster, & graph-based approaches...
   - 🗓️ Potentials for the future...
KMeans 'elbow' mappings...

- 1 : 1.6880582826666664
- 2 : 0.18531155999999988
- 3 : 0.0558509533333392
- 4 : 0.003462349999998585

KMeans optimal elbow is: 2
### DBSCAN eps and minpts are “hyperparameters”
No algorithm to calculate
EPS = \frac{\text{span}_{\text{delta}}}{2} \times 0.10

\text{DBScan} [00, 05] = \text{EPS}: 0.25
\text{DBScan} [02, 15] = \text{EPS}: 0.65
\text{DBScan} [15, 35] = \text{EPS}: 1.0
\text{DBScan} [30, 60] = \text{EPS}: 1.5
Results

90k connections
104 client IPs <-- 906 destination IPs
>= 5 records

with at least 1 cluster reporting > .50 likelihood
= 98 unique connections \{sIP, dIP, port, protocol\}
14 distinct destination IPs

with at least 1 cluster reporting >= .85 likelihood
= 58 unique connections
4 distinct destination IPs (actual beacons!)
Integration with ELK Stack

Query Elasticsearch directly
Write only deltas
Significant speed up, since no intermediate files
== Realtime alerts on result sets
Output of this Project

1. The research and results outlined in this presentation
2. Software (soon to be OSS) containing:
   a. Jupyter notebook to walk analyst through the technical details
   b. Easy to use scripts automating the analysis of production bro/zeek logs
      i. Truncate full log to delta files
      ii. Generate “top targets” report
   C. Docker container for easy integration with existing ELK stack installation
3. Many future opportunities to continue research...
Future Work

- Continue to refine clusters:
  - Improve calculations for DBSCAN via *kneed*
  - Continue to remove outliers (& “verified” non-beacons)
  - Research HDDBSCAN, OPTICS, *pycluster*, & graph-based approaches

- Improve network connection windows (time spans)
- Compare individual beacons vs. aggregates
- Visualizations, automation & resulting notifications
- Testing w/ more diverse & larger data sets