Traditional and Advanced Techniques for Network Beacon Detection

Tom Podnar

Dustin Updyke

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

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?

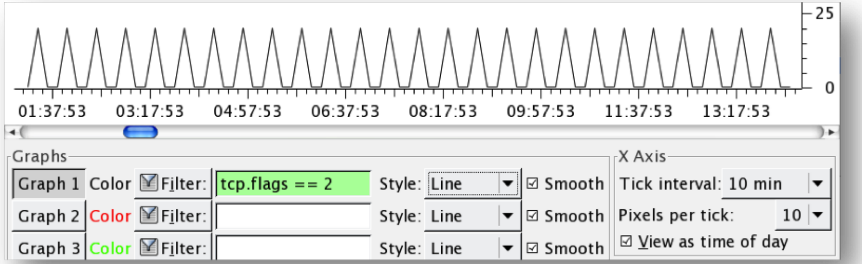
- What is a “network beacon”?
  - Network events that reoccur on a timing interval
  - Essentially a heartbeat signal

  - 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

connection_id	sip	dip	port	proto	datetime	delta
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 10:57:46.160573959	5.024812
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 11:02:47.482317924	5.022029
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 11:07:48.857930899	5.022927
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 11:12:50.197284937	5.022323
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 11:17:51.547010899	5.022495
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 11:22:52.929891109	5.023048
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 11:27:54.349085093	5.023653
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 12:43:33.702056885	75.655883
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 12:48:34.889444113	5.019790
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 12:53:36.214499950	5.022084
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 12:58:37.702542067	5.024801
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 13:08:40.251667976	10.042485
7393	192.168.31.22	143.235.14.130	443	tcp	2021-08-25 13:13:41.612114906	5.022674

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
    - 61 - 62 - 60 - 59 - 58 - 62 - 60 - 59 - 59 - 61 - 62 - 59 - 58

vs.

    - 61 - 62 - 60 - 59 - 58 - **72 - 76** - 59 - 59 - 61 - 62 - 59 - 58

💥 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**

    - [x] "Score" the result & generate list high probability targets
    - [x] 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

A

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

A

{"ts":1629896329.161627,"uid":"CZTGrG13E7t8MmHbSa","id.orig_h":"192.168.30.154","id.orig_p":63966,"id.resp_h":"52.11.181.174","id.resp_p":443,"proto":"tcp","service":"s
","duration":65.31691098213196,"orig_bytes":1391,"resp_bytes":4651,"conn_state":"SF","local_orig":true,"local_resp":false,"missed_bytes":0,"history":"ShADTadttTffFFrr","orig_pkts":52,"orig_ip_bytes":6283,"resp_pkts":51,"resp_ip_bytes":16245,"community_id":"1:/JkbRJs+jkQl5uHdjkErQHcQQLY="}
{"ts":1629896328.712649,"uid":"CNoqGV2gjhv4EGJ09e","id.orig_h":"192.168.20.153","id.orig_p":61933,"id.resp_h":"35.161.38.217","id.resp_p":443,"proto":"tcp","service":"ssl","duration":66.01725792884827,"orig_bytes":1511,"resp_bytes":4653,"conn_state":"SF","local_orig":true,"local_resp":false,"missed_bytes":0,"history":"ShADTadttTffFFFrr","orig_pkts":52,"orig_ip_bytes":6643,"resp_pkts":51,"resp_ip_bytes":16251,"community_id":"1:KcbJABuig2PNjwVht8vWWBhvIiM="}
{"ts":1629896400.718531,"uid":"CShRZr4JARfebnaBbf","id.orig_h":"192.168.30.10","id.orig_p":53975,"id.resp_h":"172.217.12.238","id.resp_p":80,"proto":"tcp","duration":0.000270128250122070 3,"orig_bytes":0,"resp_bytes":0,"conn_state":"S0","local_orig":true,"local_resp":false,"missed_bytes":0,"history":"S","orig_pkts":2,"orig_ip_bytes":104,"resp_pkts":0,"resp_ip_bytes":0,"community_id":"1:cuuccF7e6sZitBjBBXlFCYZncpc="}
{"ts":1629896330.699297,"uid":"CV4dl1312KMJIAZLqf","id.orig_h":"192.168.31.19","id.orig_p":59108,"id.resp_h":"35.190.141.208","id.resp_p":443,"proto":"tcp","service":"ssl","duratio 3669896125794,"orig_bytes":1333,"resp_bytes":4466,"conn_state":"SF","local_orig":true,"local_resp":false,"missed_bytes":0,"history":"ShADTadttTffFFrr","orig_pkts":52, _bytes":6109,"resp_pkts":51,"resp_ip_bytes":15690,"community_id":"1:qKk1UzW0bns5CPECLjx4KS3QJHQ="}
{"ts":1629896401.51377,"uid":"CXfAGT2Zt1c9x6cUm8","id.orig_h":"192.168.30.214","id.orig_p":58407,"id.resp_h":"52.39.105.74","id.resp_p":80,"proto":"tcp","duration":0.00006198883056640625,"orig_bytes":0,"resp_bytes":0,"conn_state":"S0","local_orig":true,"local_resp":false,"missed_bytes":0,"history":"S","orig_pkts":2,"orig_ip_bytes":104,"resp_pkts":0,"resp_ip_bytes":0,"community_id":"1:ItM85A3UK7I5NrSAsPdH289fQrg="}

connection_id	sip	dip	port	proto	datetime	delta
018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:19:32.889918089	10.784478
018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:24:35.651804924	5.046017
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:34:41.989473104	10.105615
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:44:47.288450956	10.088304
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:55:05.672874928	10.306401

connection_id	delta
2018	10.784478
2018	5.046017
2018	10.105615
2018	10.0883
2018	10.3064

Key #2: Clusters Identify Potential Beacons

B

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
     - Input values are not algorithmic (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...
   - 31 Potentials for the future...

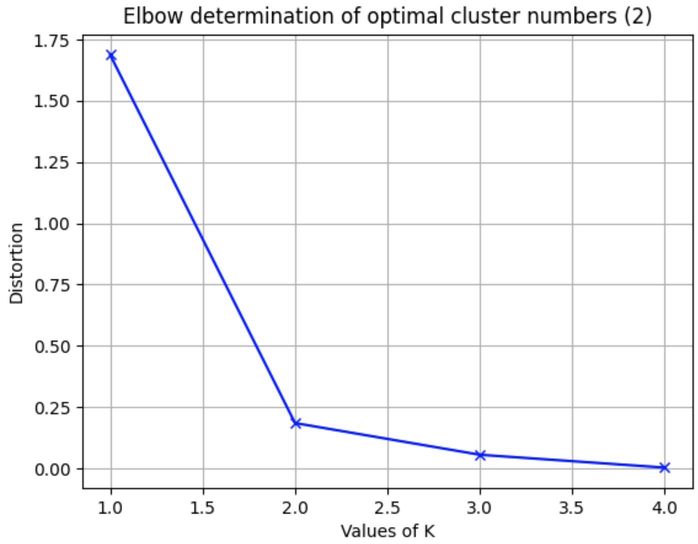
B

connection_id	sip	dip	port	proto	datetime	delta
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:19:32.889918089	10.784478
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:24:35.651804924	5.046017
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:34:41.989473104	10.105615
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:44:47.288450956	10.088304
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:55:05.672874928	10.306401

KMeans 'elbow' mappings...

- 1 : 1.6880582826666664
- 2 : 0.18531155999999988
- 3 : 0.05585095333333392
- 4 : 0.0034623499999998585

KMeans optimal elbow is: 2



B

connection_id	sip	dip	port	proto	datetime	delta
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:19:32.889918089	10.784478
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:24:35.651804924	5.046017
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:34:41.989473104	10.105615
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:44:47.288450956	10.088304
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:55:05.672874928	10.306401

DBSCAN eps and minpts are “hyperparameters”

No algorithm to calculate

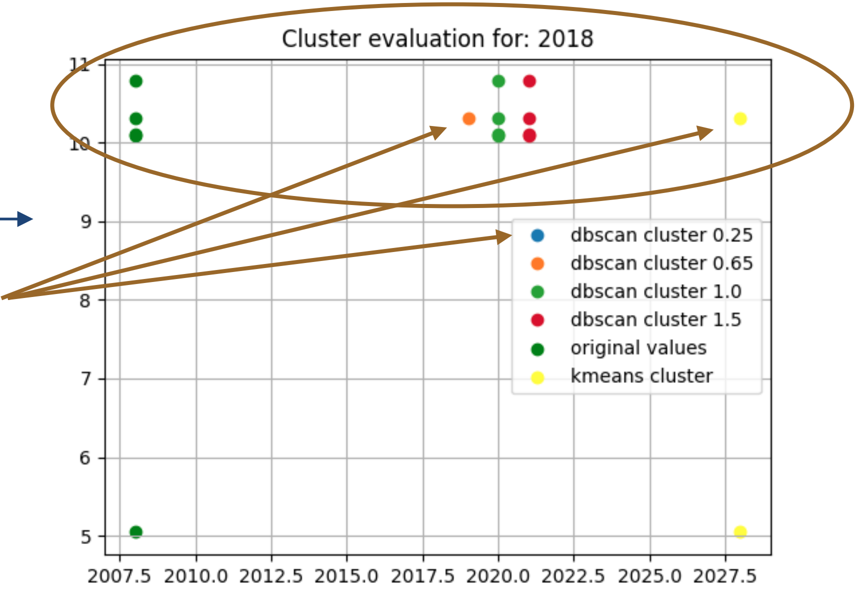
?

C

connection_id	sip	dip	port	proto	datetime	delta
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:19:32.889918089	10.784478
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:24:35.651804924	5.046017
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:34:41.989473104	10.105615
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:44:47.288450956	10.088304
2018	192.168.31.21	172.217.9.200	443	tcp	2021-08-25 13:55:05.672874928	10.306401

EPS = (span_delta / 2) * .10

DBScan [00, 05] = EPS: 0.25
DBScan [02, 15] = EPS: 0.65
DBScan [15, 35] = EPS: 1.0
DBScan [30, 60] = 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!)

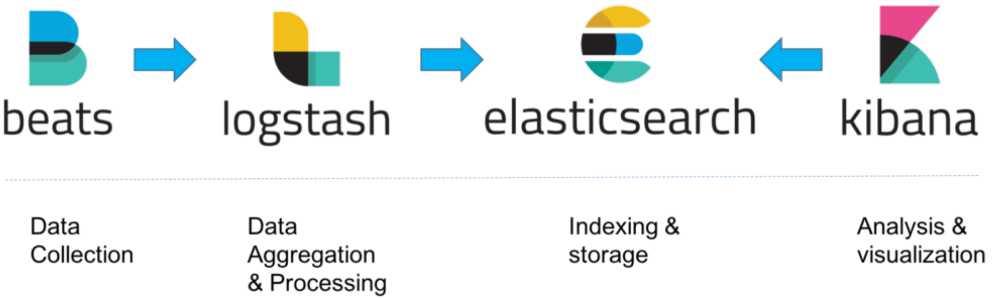
Integration with ELK Stack

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