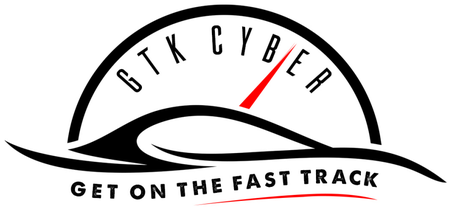
# Module 11: Hunting with Data Science

Increasing the Signal-to-Noise Ratio

GTK CYBER
GET ON THE FAST TRACK


# What is it?


# Semantics




# Threat Hunting

Cyber threat hunting is "the process of ***proactively and iteratively*** searching through networks to detect and isolate **advanced threats** that ***evade*** existing security solutions."

# Data Science

Data science is an interdisciplinary field about scientific methods, processes, and systems to ***extract knowledge or insights from data*** in various forms, either structured or unstructured, similar to data mining.

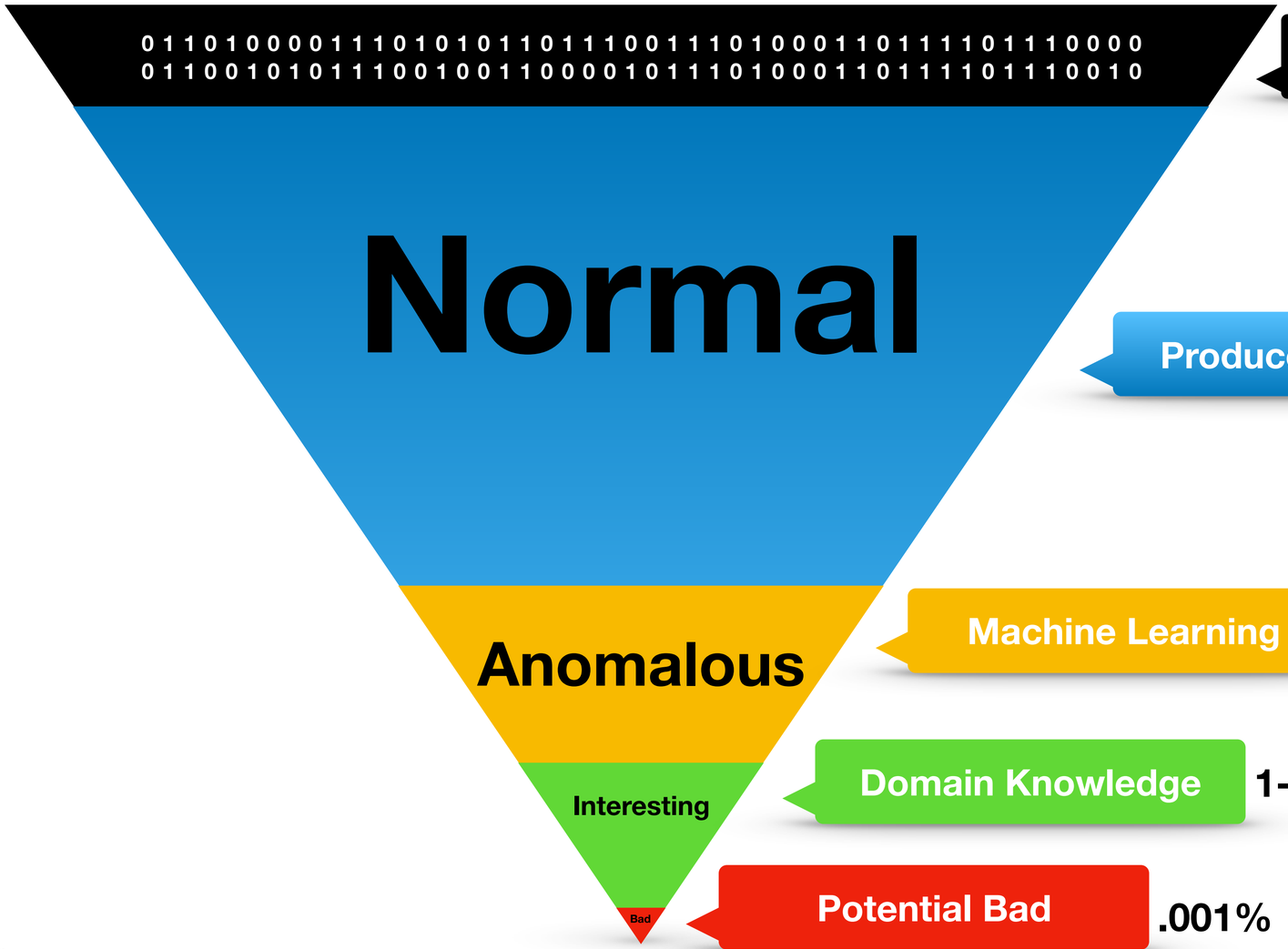

Threat Hunting

+

Data Science

# Hunting with Data Science

# Data Science Hunting Funnel




Network Traffic

Produced Naturally 100%

Machine Learning 10%

Domain Knowledge 1-5%

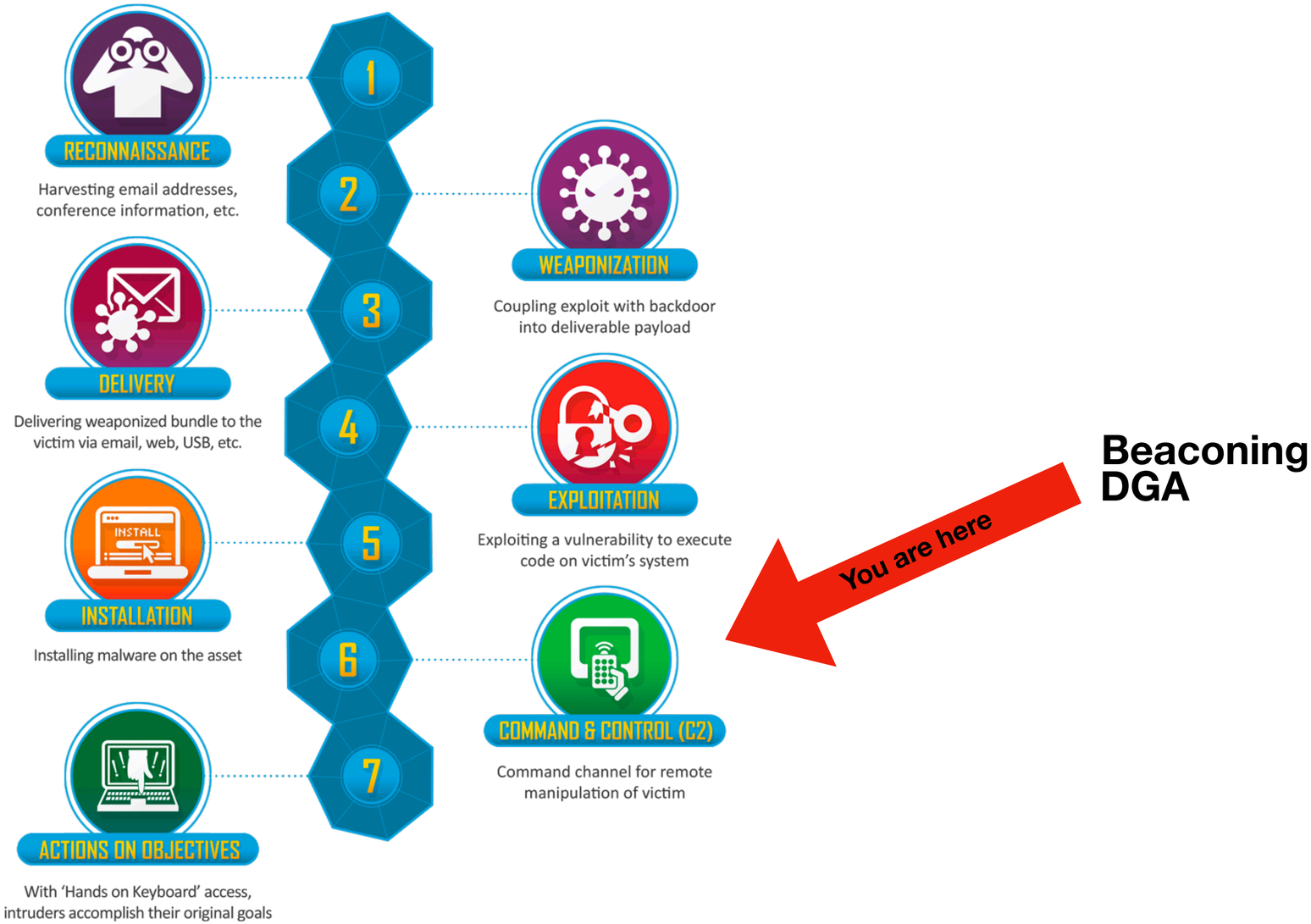
Potential Bad .001%

@HuntOperator

gtkcyber.com

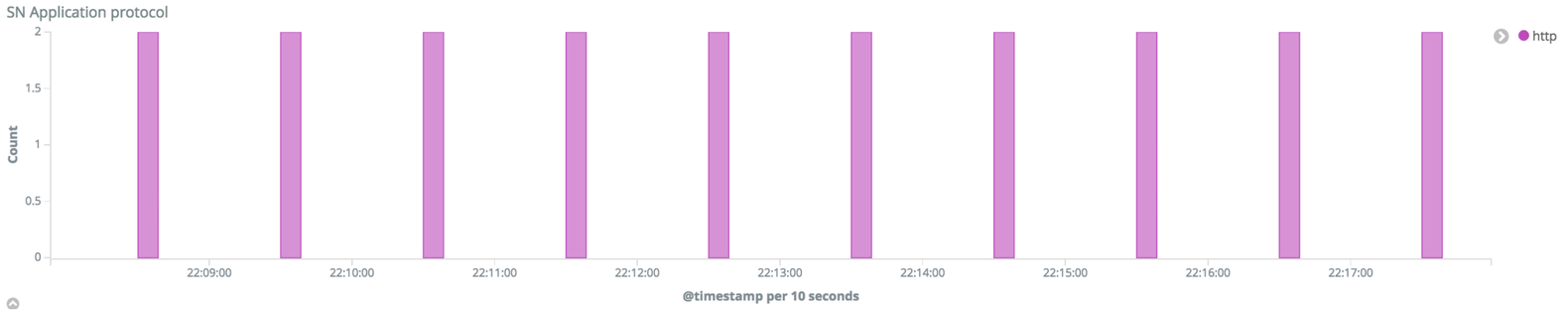

# Cyber Kill Chain

# Cyber Kill Chain




# Beaconing

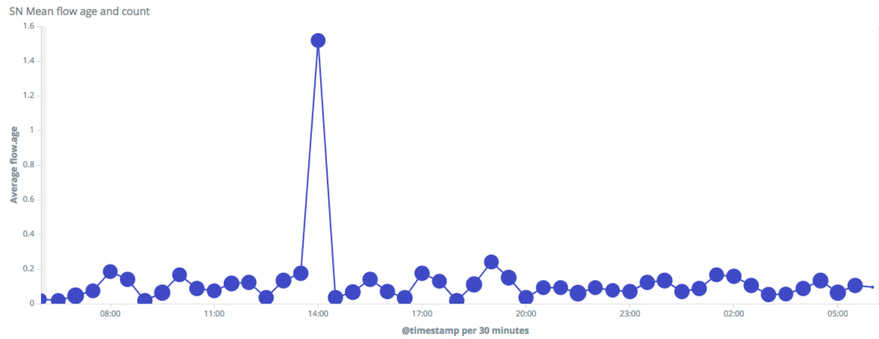
# Beaconing




- Post-Infection
- Early network-related indication of infection
- Used by malware to “phone home” to command and control server

# Detection Challenges

- **Hardset Intervals**
  - Varying window sizes
- **Legit Services**
  - Windows update
  - Virus definition updates




# Beaconing Detection

# Beaconing: Detection

- Free Open Source Software
- Designed for data scientists, security researchers
- Written in Python
- Used for rapid prototyping and development of behavioral analytics
- Intended to make identifying malicious behavior in networks as simple as possible.

Flare
An analytical framework for network data

**https://github.com/austin-taylor/flare**

# Beaconing: Detection


```
[beacon]
es_host=localhost             # IP address of ES Host, which we forwarded to localhost
es_index=logstash-flow-*      # ES index
es_port=9200                  # Logstash port (we forwarded earlier)
es_timeout=480                # Timeout limit for elasticsearch retrieval
min_occur=50                  # Minimum of 50 network occurrences to appear in traffic
min_interval=30               # Minimum interval of 30 seconds per beacon
min_percent=30                # Beacons must represent 30% of network traffic per dyad
window=3                      # Accounts for jitter... For example, if 60 second beacons
                              # occurred at 58 seconds or 62 seconds, a window of 3 would
                              # factor in that traffic.
threads=8                     # Use 8 threads to process (Should be configured)
period=24                     # Retrieve all flows for the last 24 hours.
kibana_version=5              # Your Kibana version. Currently works with 4 and 5
verbose=True                  # Display output while running script
```

**https://github.com/austin-taylor/flare**

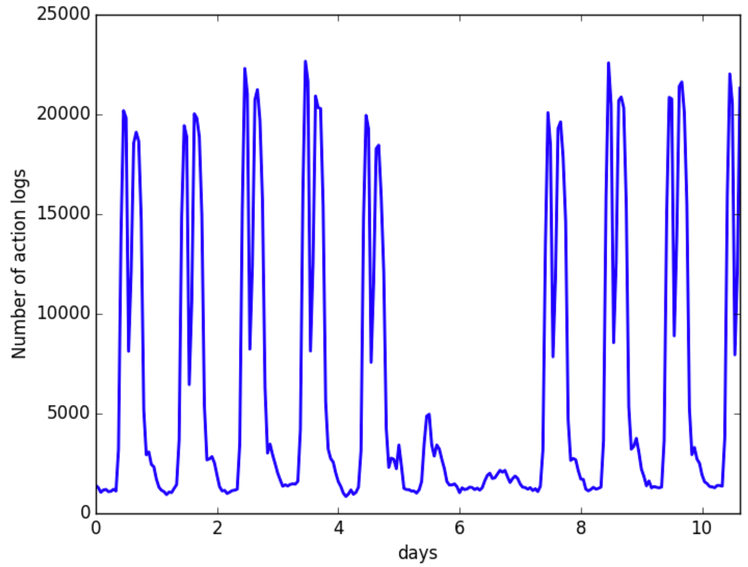

# Beaconing: Data Science

| Time | EveBox | src_ip | src_port | proto | dest_ip | dest_port | http.http_method | http.hostname |
|---|---|---|---|---|---|---|---|---|
| June 13th 2017, 10:49:01.935 | Correlate Flow | 192.168.0.53 | 49,182 | TCP | 160.153.76.129 | 80 | GET | www.huntoperator.com |

# Beaconing: Data Science

**Simple:** src_ip, dest_ip, dest_port -> hash

**More Complex:** Discrete Fourier Transform (DFT)/Fast Fourier transform (FFT)



# Scenario 1

A piece of malware has infected a computer (192.168.0.53) on your network and is trying to reach back to its Command and Control (C2) server (160.153.76.129) in periodic intervals

GTK CYBER

GET ON THE FAST TRACK

# HUNT!

gtkcyber.com


# Beaconing: Hunt

**flare_beacon -c configs/selks4.ini -csv beacons.csv**

```
Austins-MacBook-Pro:flare huntoperator$ flare_beacon -c configs/selks4.ini -csv beacons.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[SUCCESS] Writing csv to beacons.csv
```

**108**

**events to process**


# Beaconing: Hunt

**flare_beacon -c configs/selks4.ini -csv —group —whois —focus_outbound beacons_filtered.csv**

```
Austins-MacBook-Pro:flare huntoperator$ flare_beacon -c configs/selks4.ini --whois --focus_outbound -csv beacons_filtered.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[INFO] Enriching IP addresses with whois information
[INFO] Applying outbound focus - filtering multicast, reserved, and private IP space
[SUCCESS] Writing csv to beacons_filtered.csv
```

**31**

**events to process**

# Beaconing: Hunt

**flare_beacon -c configs/selks4.ini -csv —group —whois —focus_outbound beacons_filtered.csv**

```
Austins-MacBook-Pro:flare huntoperator$ flare_beacon -c configs/selks4.ini --whois --focus_outbound -csv beacons_filtered.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[INFO] Enriching IP addresses with whois information
[INFO] Applying outbound focus - filtering multicast, reserved, and private IP space
[SUCCESS] Writing csv to beacons_filtered.csv
```

## What was applied?

–group: This will group the results making it visually easier to identify anomalies.

–whois: Enriches IP addresses with WHOIS information through ASN Lookups.

–**focus_outbound**: **Filters out multicast, private and broadcast addresses from destination IPs**

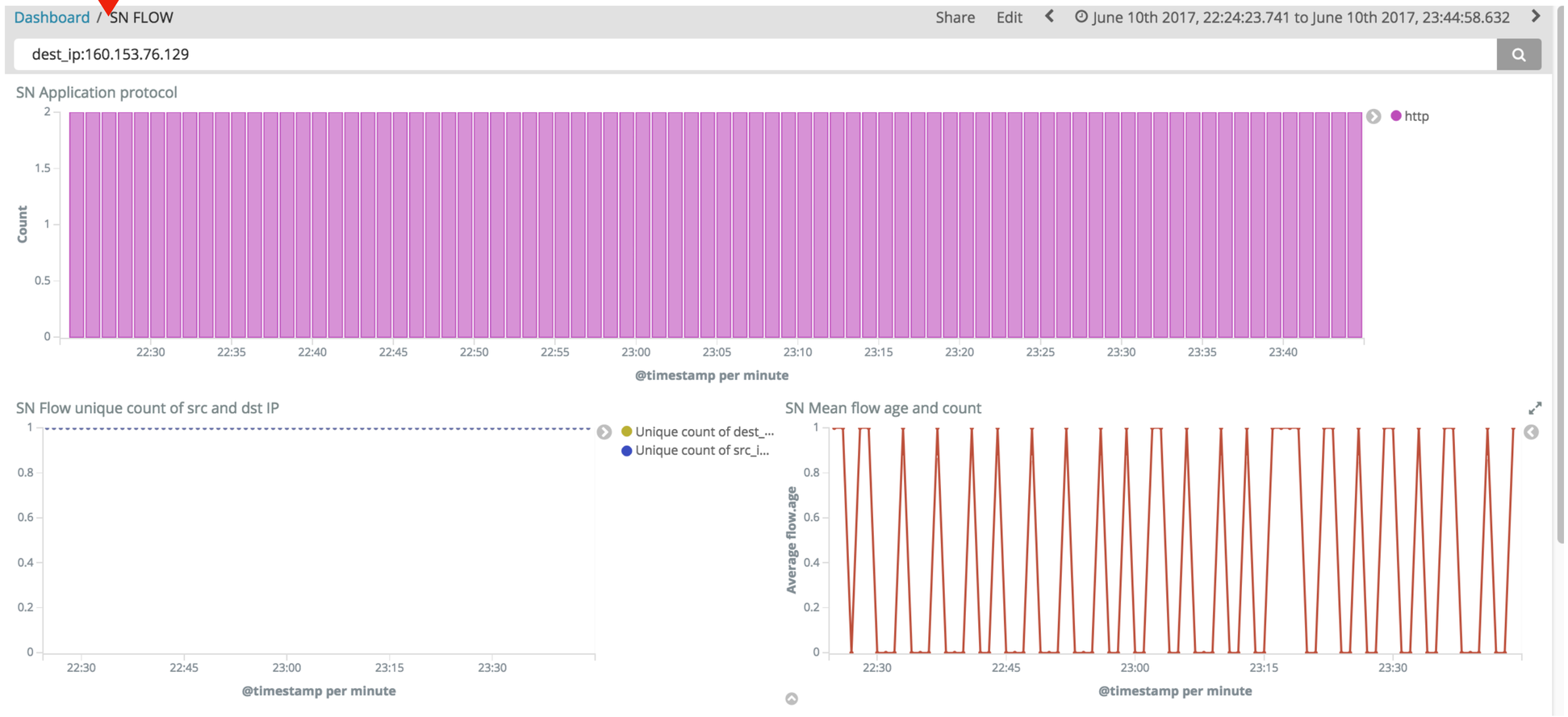
# Beaconing: Hunt

| src_ip | dest_whois | dest_ip | dest_port | bytes_toserver | dest_degree | occurances | percent | interval |
|---|---|---|---|---|---|---|---|---|
| 192.168.0.100 | GOOGLE - Google Inc., US | 8.8.8.8 | 53 | 82 | 16 | 71 | 98 | 3601 |
| | SOFTLAYER - SoftLayer Technologies Inc., US | 198.23.79.227 | 123 | 90 | 1 | 71 | 98 | 3601 |
| 192.168.0.120 | AMAZON-02 - Amazon.com, Inc., US | 52.192.78.22 | 80 | 1432 | 3 | 34 | 70 | 1201 |
| | GOOGLE - Google Inc., US | 8.8.4.4 | 53 | 74 | 7 | 139 | 63 | 61 |
| | | 8.8.8.8 | 53 | 74 | 16 | 140 | 62 | 61 |
| 192.168.0.53 | AS-26496-GO-DADDY-COM-LLC - GoDaddy.com, LLC, US | 160.153.76.129 | 80 | 620 | 3 | 383 | 97 | 61 |
| | GOOGLE - Google Inc., US | 8.8.4.4 | 53 | 233 | 7 | 386 | 95 | 61 |
| | | 8.8.8.8 | 53 | 65 | 16 | 386 | 95 | 61 |
| 192.168.0.60 | AKAMAI-ASN1 , US | 104.95.176.44 | 80 | 8399 | 1 | 3 | 33 | 97 |
| | AMAZON-AES - Amazon.com, Inc., US | 174.129.1.163 | 80 | 1691 | 1 | 77 | 38 | 101 |
| | | 204.236.238.3 | 80 | 885 | 1 | 84 | 39 | 111 |
| | | 23.21.110.37 | 80 | 1691 | 1 | 53 | 35 | 200 |
| | | 23.21.67.210 | 80 | 1691 | 1 | 101 | 32 | 200 |
| | | 23.21.98.133 | 80 | 1691 | 1 | 65 | 40 | 101 |
| | | 23.23.173.186 | 80 | 1691 | 1 | 43 | 32 | 200 |
| | | 23.23.236.78 | 80 | 759 | 1 | 313 | 41 | 112 |
| | | 50.16.215.193 | 80 | 1691 | 1 | 25 | 40 | 101 |
| | | 50.17.254.18 | 80 | 1691 | 1 | 25 | 40 | 200 |
| | DROPBOX - Dropbox, Inc., US | 162.125.32.5 | 443 | 60 | 1 | 53 | 35 | 602 |
| | EDGECAST - MCI Communications Services, Inc. d/b/a Verizon Business, US | 152.195.54.20 | 80 | 556 | 2 | 69 | 49 | 601 |
| | LLNW - Limelight Networks, Inc., US | 69.164.0.0 | 80 | 6242 | 2 | 3 | 33 | 36 |
| | MICROSOFT-CORP-MSN-AS-BLOCK - Microsoft Corporation, US | 13.107.6.151 | 443 | 3985 | 1 | 151 | 41 | 300 |

- **bytes_toserver**: Total sum of bytes sent from IP address to Server
- **dest_degree**: Amount of source IP addresses that communicate to the same destination
- **occurrences**: Number of network occurrences between dyads identified as beaconing.
- **percent**: Percent of traffic between dyads considered beaconing.
- **interval**: Intervals between each beacon in seconds

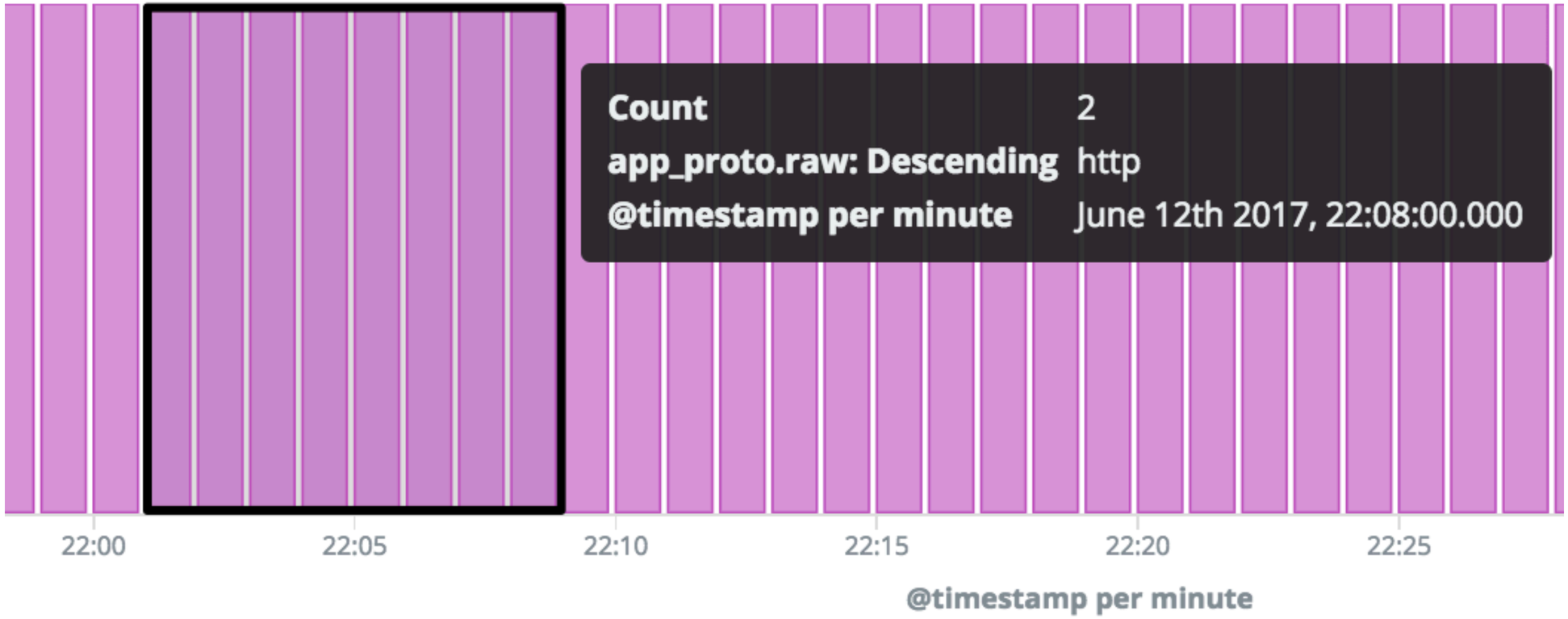
# Beaconing: Hunt

## Validate Results

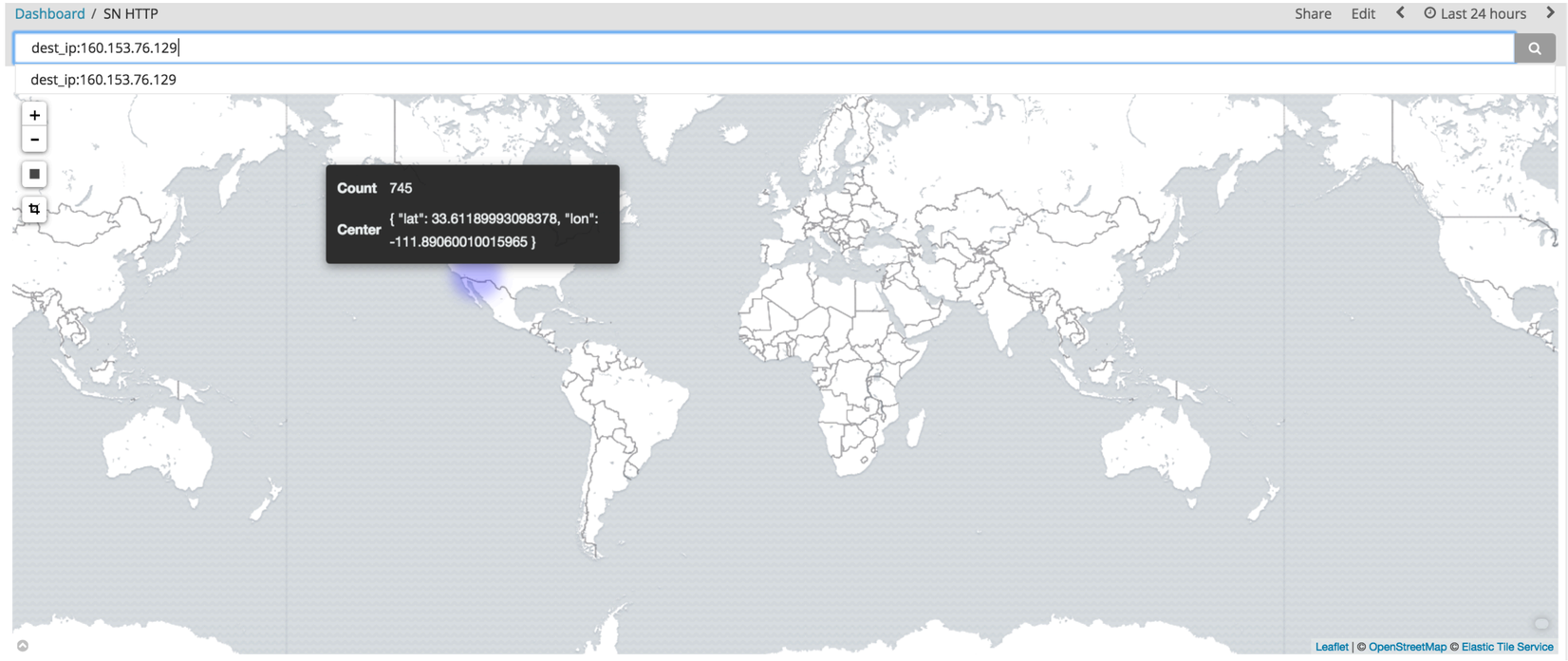


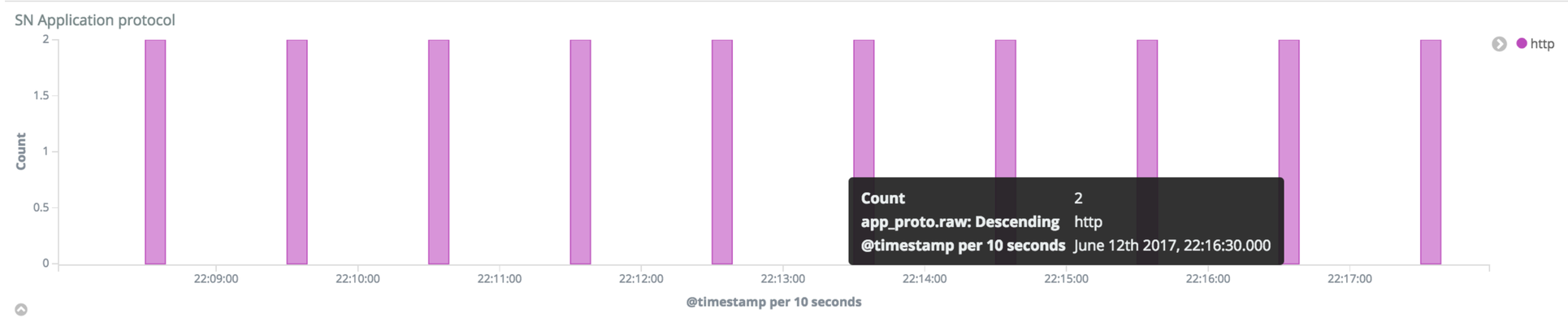
# Beaconing: Hunt

***Drilling in***

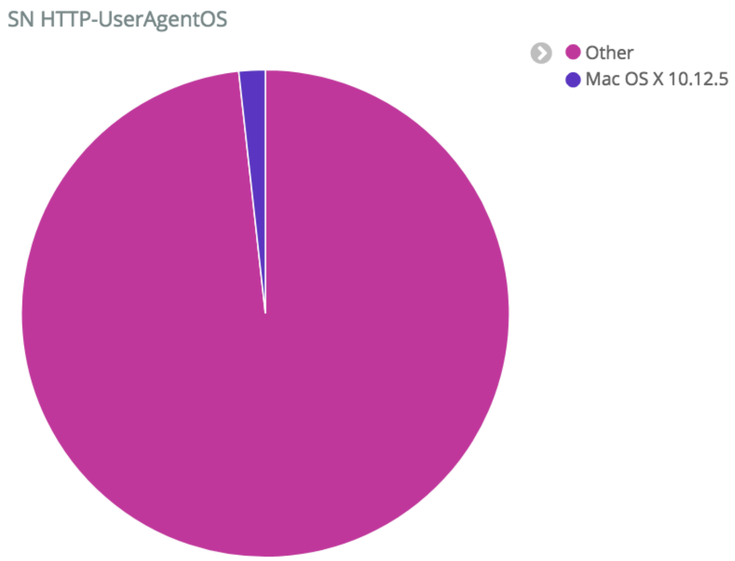


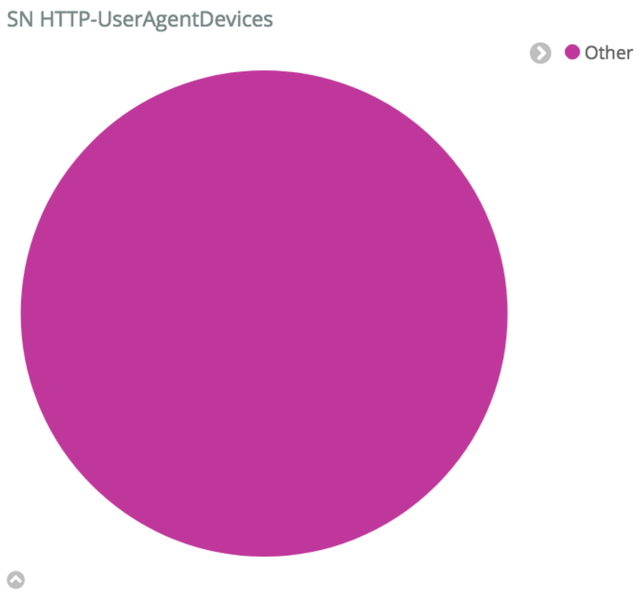
# Beaconing: Hunt

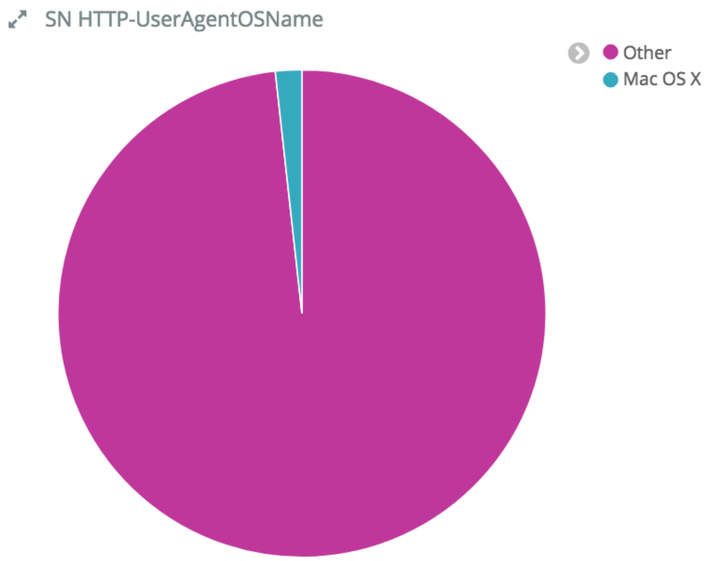


# Beaconing: Hunt




SN HTTP-EventsList

| Time | EveBox | src_ip | src_port | proto | dest_ip | dest_port | http.http_method | http.hostname | http.status | http.protocol | http.server |
|---|---|---|---|---|---|---|---|---|---|---|---|
| June 13th 2017, 10:49:01.935 | Correlate Flow | 192.168.0.53 | 49,182 | TCP | 160.153.76.129 | 80 | GET | www.huntoperator.com | 503 | HTTP/1.1 | Apache/2.4.25 |
| June 13th 2017, 10:48:01.680 | Correlate Flow | 192.168.0.53 | 49,180 | TCP | 160.153.76.129 | 80 | GET | www.huntoperator.com | 503 | HTTP/1.1 | Apache/2.4.25 |
| June 13th 2017, 10:47:01.388 | Correlate Flow | 192.168.0.53 | 49,178 | TCP | 160.153.76.129 | 80 | GET | www.huntoperator.com | 503 | HTTP/1.1 | Apache/2.4.25 |
| June 13th 2017, 10:46:01.102 | Correlate Flow | [illegible] | 49,176 | TCP | 160.153.76.129 | 80 | GET | www.huntoperator.com | 503 | HTTP/1.1 | Apache/2.4.25 |

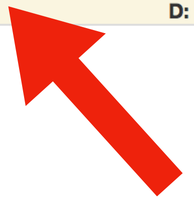

# Beaconing: Hunt

EveBox Inbox Escalated Alerts Events Reports

721700456887339

Refresh Event Type: All

| | Timestamp | Type | Source/Dest | Description |
|---|---|---|---|---|
| > | 2017-06-13 10:50:03<br>7 hours ago | FLOW | **S:** 192.168.0.53<br>**D:** 160.153.76.129 | TCP 192.168.0.53:49182 -> 160.153.76.129:80; Age: 1; Bytes: 6223; Packets: 19 |
| | 2017-06-13 10:50:03<br>7 hours ago | FLOW | **S:** 192.168.0.53<br>**D:** 160.153.76.129 | TCP 192.168.0.53:49182 -> 160.153.76.129:80; Age: 1; Bytes: 6223; Packets: 19 |
| | 2017-06-13 10:49:02<br>7 hours ago | FILEINFO | **S:** 160.153.76.129<br>**D:** 192.168.0.53 | / - Hostname: www.huntoperator.com; Content-Type: text/html |
| | 2017-06-13 10:49:01<br>7 hours ago | HTTP | **S:** 192.168.0.53<br>**D:** 160.153.76.129 | GET - www.huntoperator.com - / |
| | 2017-06-13 10:49:01<br>7 hours ago | ALERT | **S:** 192.168.0.53<br>**D:** 160.153.76.129 | ET POLICY curl User-Agent Outbound |




# Beaconing: Hunt

**HTTP**

*Hostname:* www.huntoperator.com
*Http User Agent:* curl/7.47.0
*Status:* 503
*User Agent.Name:* Other

*Http Content Type:* text/html
*Length:* 1093
*Url:* /
*User Agent.Os:* Other

*Http Method:* GET
*Protocol:* HTTP/1.1
*User Agent.Device:* Other
*User Agent.Os Name:* Other

**GeoIP**

*City Name:* Scottsdale
*Coordinates.1:* 33.6119
*Country Name:* United States
*Latitude:* 33.6119
*Longitude:* -111.8906
*Region Name:* Arizona

*Continent Code:* NA
*Country Code2:* US
*Dma Code:* 753
*Location.0:* -111.8906
*Postal Code:* 85260
*Timezone:* America/Phoenix

*Coordinates.0:* -111.8906
*Country Code3:* US
*Ip:* 160.153.76.129
*Location.1:* 33.6119
*Region Code:* AZ

**Flow - TCP 192.168.0.53 -> 160.153.76.129** [ Open ]

*Age:* 1
*Bytes Toserver:* 752
*Pkts Toserver:* 10
*State:* closed

*Alerted:* true
*End:* 2017-06-13T10:49:02.275746-0400
*Reason:* timeout

*Bytes Toclient:* 5471
*Pkts Toclient:* 9
*Start:* 2017-06-13T10:49:01.717867-0400

**Payload** [ PCAP ]

```
GET / HTTP/1.1
Host: www.huntoperator.com
User-Agent: curl/7.47.0
Accept: */*
```

```
47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a
48 6f 73 74 3a 20 77 77 77 2e 68 75 6e 74 6f 70
65 72 61 74 6f 72 2e 63 6f 6d 0d 0a 55 73 65 72
2d 41 67 65 6e 74 3a 20 63 75 72 6c 2f 37 2e 34
37 2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a
0d 0a 0d 0a
```

# Beaconing: Hunt

## Report for IP 160.153.76.129

Related Reports

Refresh | All Sensors | Filter... | Apply | Clear

Alerts Over Time


**DNS Hostnames Returning 160.153.76.129**

| # | Hostname |
|---|---|
| 1806 | huntoperator.com |

**DNS: Top Requested Hostnames**

No data.

**Outgoing HTTP User Agents**

No data.

**HTTP: Incoming HTTP Request Hostnames**

| # | Hostnames |
|---|---|
| 903 | www.huntoperator.com |

**Alerts: Top Alerts**

| # | Signature |
|---|---|
| 890 | ET POLICY curl User-Agent Outbound |

**Flow**

| | |
|---|---|
| Flows As Client | 0 |
| Flows As Server | 1794 |
| Bytes To... | 1.23 MB |
| Bytes From... | 12.70 MB |
| Packets To... | 16k (16908) |
| Packets From... | 19k (19084) |

**Incoming TLS Server Names (SNI)**

No data.

**TLS Versions as Client**

No data.

**TLS Versions as Server**

No data.

**HTTP: Top Requested Hostnames**

No data.

CASE SOLVED

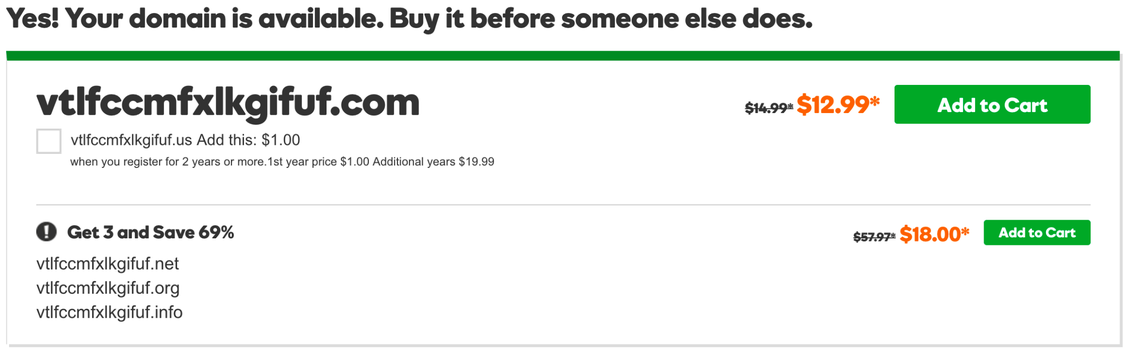

# Domain Generation Algorithms (DGA)

# Domain Generation Algorithms (DGA)

vtlfccmfxlkgifuf.com

## Why DGA?

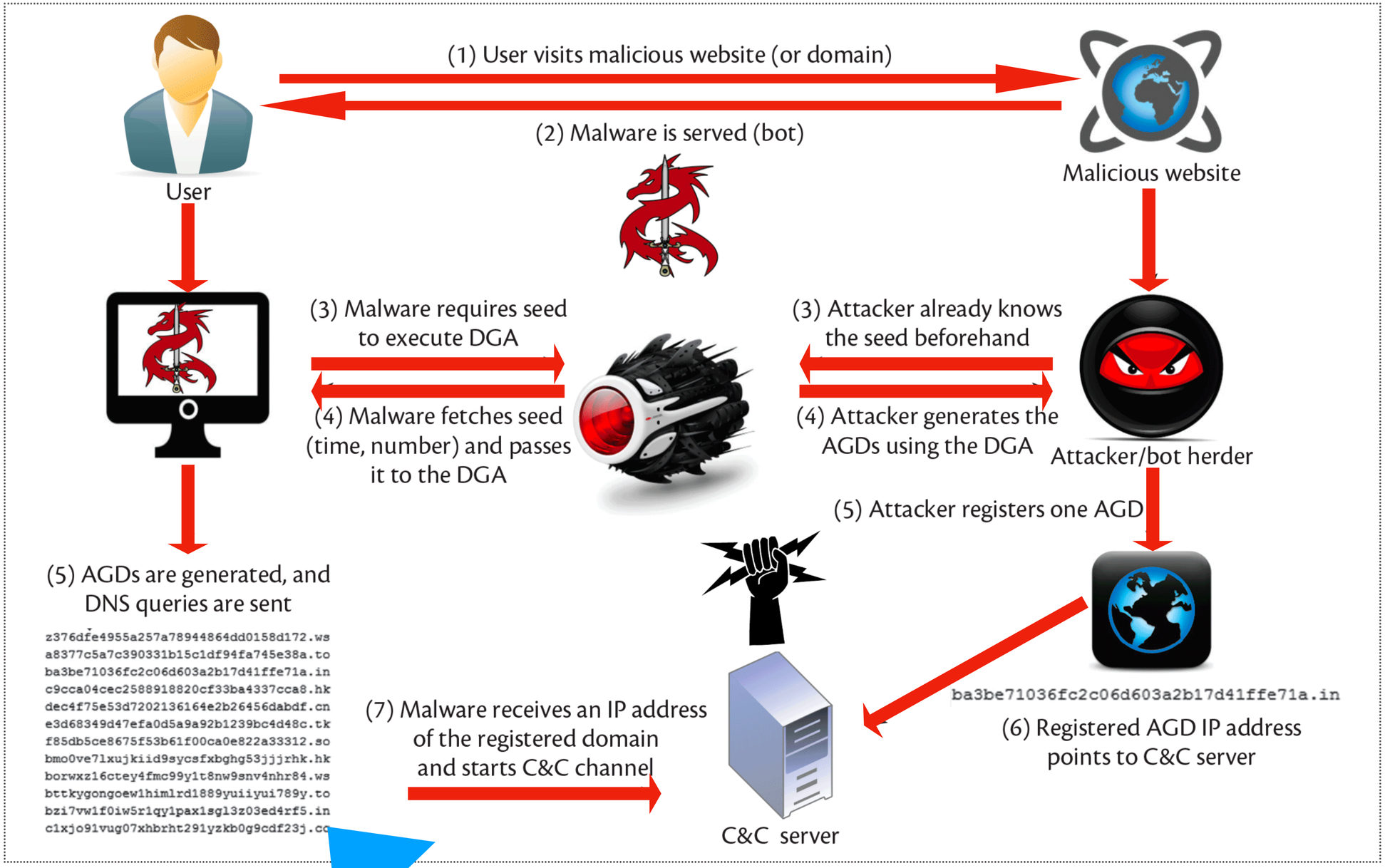
- Deterministic value
- Generate large number of domain names
  - Easy to burn
  - Cheap to register
- Used as a rendezvous point by attacker




In [18]:
```
1  def generate_domain(year, month, day):
2      """Generates a domain name for the given date."""
3      domain = ""
4
5      for i in range(16):
6          year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
7          month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
8          day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
9          domain += chr(((year ^ month ^ day) % 25) + 97)
10
11     return domain + '.com'
```

In [19]:
```
1  generate_domain(2017, 6, 23)
```

Out[19]:
```
'vtlfccmfxlkgifuf.com'
```




*Source: Aditya K. Sood, Sherali Zeadally, "A [illegible] of Domain-Generation Algorithms", IEEE Security & Privacy, vol. 14, no. , pp. 46-53, July-Aug. 2016, doi:10.1109/MSP.2016.76*

**We want to detect this**


I'M GONNA HAVE TO DATA
SCIENCE THE SH*T OUT OF THIS
imgflip.com

# Scenario 2

# Scenario 2

A piece of malware has infected a computer on your network and is making request to domains using DGA in an attempt to communicate to a Command and Control Server

GTK CYBER

GET ON THE FAST TRACK

# HUNT!

gtkcyber.com

# DNS Records

## Record Count: 15408

```
In [62]: 1 dns_records = pd.read_csv('/Users/huntoperator/Downloads/exported_dns_n
```

```
In [63]: 1 dns_records
```

Out[63]:

| | dns_rrname | count |
|---|---|---|
| 0 | daisy.ubuntu.com | 300,129 |
| 1 | srv.myskybell.com | 55,053 |
| 2 | googleapis.l.google.com | 37,005 |
| 3 | ubuntu.com | 30,549 |
| 4 | www.google.com | 28,780 |
| 5 | www.example.org | 26,367 |
| 6 | www.example.com | 26,354 |
| 7 | www.example.net | 26,298 |
| 8 | myskybell.com | 18,296 |

# Import Flare Tools

```
In [64]:  1 from flare.data_science.features import dga_classifier
```

```
In [66]:  1 from flare.tools.alexa import Alexa
          2 from flare.data_science.features import domain_tld_extract
```

```
In [65]:  1 dga_c = dga_classifier()

[*] Initializing... training classifier - Please wait.
[+] Classifier Ready
```

- **DGA Classifier**
  - Random Forrest Classifier
  - N-Grams
  - Uses labelled data
- **Alexa - Top 1M most popular visited websites**
  - Must pay for service now.
  - Umbrella/Majestic are free alternatives
- **Domain TLD Extract - Extracts the Top Level Domain to be checked against Alexa**
  - Also calculate degree from here

# Filter Results

```
In [73]: 1 dns_records['domain_tld'] = dns_records.dns_rrname.apply(lambda x: domain_tld_extract(str(x)))

In [78]: 1 dns_records['dga_predict'] = dns_records.domain_tld.apply(lambda x: dga_c.predict(x))

In [129]: 1 dns_records.head()
```

Out[129]:

| | dns_rrname | count | domain_tld | dga_predict |
|---|---|---|---|---|
| 0 | daisy.ubuntu.com | 300,129 | ubuntu.com | legit |
| 1 | srv.myskybell.com | 55,053 | myskybell.com | legit |
| 2 | googleapis.l.google.com | 37,005 | google.com | legit |
| 3 | ubuntu.com | 30,549 | ubuntu.com | legit |
| 4 | www.google.com | 28,780 | google.com | legit |

```
In [84]: 1 dns_filter = dns_records[dns_records.dga_predict=='dga']

In [131]: 1 dns_filter.shape

Out[131]: (240, 4)
```

still too many results…

# Filter Results

```
1 beacon_df.groupby(self.beacon_dest_ip)[self.beacon_dest_ip].transform('count').fillna(0).astype(int)
```

```
1 dns_filter1['domain_degree'] = dns_filter1.groupby('domain_tld')['domain_tld'].transform('count').fillna(0).
```

and yet…

```
In [103]: 1 dns_filter1 = dns_filter[dns_filter['count'] < 10]
```

```
In [105]: 1 dns_filter1.shape
Out[105]: (162, 4)
```

**Still too many results…**

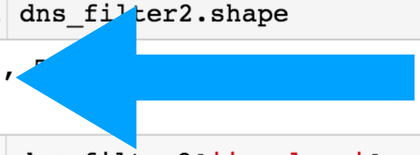
# Filter Results

```
In [117]: 1 alexa = Alexa()

In [123]: 1 dns_filter2 = dns_filter1[dns_filter1['domain_degree'] < 2]

In [119]: 1 dns_filter2.shape

Out[119]: (78,
```



down to 78

```
In [125]: 1 dns_filter2['in_alexa'] = dns_filter2.domain_tld.apply(lambda x: alexa.domain_in_alexa(x))
```

**And finally…**

# Filter Results

## apply Alexa top 1 million check…

```
In [127]:  1 dns_filter2[dns_filter2['in_alexa']==False]
```

Out[127]:

| | dns_rrname | count | domain_tld | dga_predict | domain_degree | in_alexa |
|---|---|---|---|---|---|---|
| **5421** | version.mos.svc.ovi.com.glb.as1248.net | 9 | as1248.net | dga | 1 | False |
| **5765** | ebdr3.com | 8 | ebdr3.com | dga | 1 | False |
| **5847** | i.s-jcrew.com | 8 | s-jcrew.com | dga | 1 | False |
| **5885** | in.ml314.com | 8 | ml314.com | dga | 1 | False |
| **6382** | vtlfccmfxlkgifuf.com | 8 | vtlfccmfxlkgifuf.com | dga | 1 | False |
| **7510** | tags.wdsvc.net | 6 | wdsvc.net | dga | 1 | False |
| **7766** | get35.com | 5 | get35.com | dga | 1 | False |
| **7920** | x64dbg.com | 5 | x64dbg.com | dga | 1 | False |

**and…**

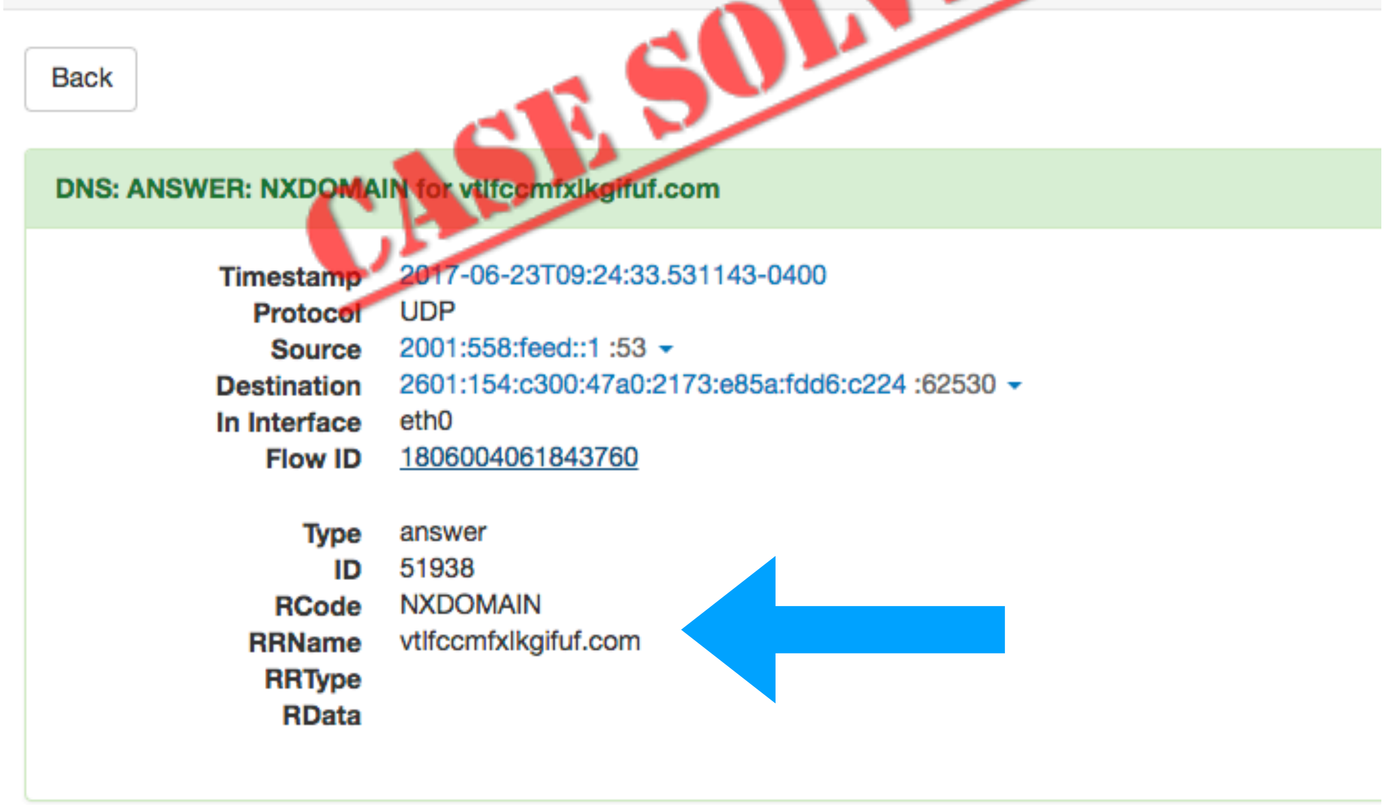
```
In [135]:  1 dns_filter2[dns_filter2['in_alexa']==False].shape

Out[135]: (57,
```

**57 Results!**

# Pass to Analyst

- **Identify Process Generating Traffic**
  - Isolate infected host
  - Begin endpoint investigation…




# Natural Language Processing

# Natural Language Processing
# Use Case

Challenge: Distinguishing between base64 and URIs

**String 1:** **Q09NRSBUTyBNWSBQWVRIT04gQ0xBU1MgSU4gVkVHQVMhISEh**
**String 2:** **forums/diary/Detecting+Random+Finding+Algorithmically+chosen+DNS+names+DGA/19893**

Both of these are properly formatted BASE64 encoded strings,
but only one is intentional

https://github.com/sans-blue-team/blue-team-wiki/blob/gh-pages/Tools/freq.py.md

# Natural Language Processing

Solution: Mark Baggett's freq.py

Produces a **probability score** based off frequency tables in:

- Dictionary
- Your environment


Higher likelihood scores indicate common


Lower likelihood scores indicate not common

https://github.com/MarkBaggett/freq


# Freq.py

## How does it work?

- Identifies likelihood of character occurrence based on frequency analysis

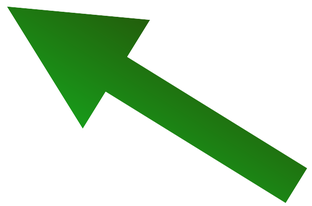
Example: In English text, “m” is usually followed by an “o,” so seeing a “m” followed by something else would be rather unlikely to occur

Similar to high order markov chain

https://github.com/sans-blue-team/blue-team-wiki/blob/gh-pages/Tools/freq.py.md

# Using Freq.py

```
>>>  from freq import *
>>>  fc = FreqCounter()
>>>  fc.load("english_lowercase.freq")
```



Pre-built frequency table

https://isc.sans.edu/diary/freq.py+super+powers%3F/19903

# Using Freq.py

How did we score?

```
>>> fc.probability('forums/diary/Detecting+Random+Finding+Algorithmically+chosen+DNS+names+DGA/19893')
9.490788394012279

>>> fc.probability('Q09NRSBUTyBNWSBQWVRIT04gQ0xBU1MgSU4gVkVHQVMhISEh')
2.578357325221811
```

https://isc.sans.edu/diary/freq.py+super+powers%3F/19903


# Freq-ing awesome…

Build Separate frequency tables for:

- DNS Host names that are normally seen for your environment
- Names of files on your file server
- Host names inside of SSL certificates
- URLs for a specific Web Application
- * Insert any string you want to measure here *

*Runs as a restful API and integrates with logstash*

https://isc.sans.edu/forums/diary/Continuous+Monitoring+for+Random+Strings/20451/

# Freq.py - Operational Examples

DNS - Parent Domain Frequency Analysis

| Domain | Frequency Score |
|---|---|
| tahmqkmvzg.com | 0.434 |
| dsms0mj1bbhn4.cloudfront.net | 1.085 |
| oiugixyixy.com | 1.683 |
| yfksjfsunyiypu.com | 2.053 |
| fwhynaxzwkv.com | 2.329 |
| yahoodns.net | 2.697 |
| muihoc.com | 2.914 |
| qfrwozmoaqc.com | 3.153 |
| aemmiphbweeuef59.com | 3.345 |
| brovztkzbl.com | 3.391 |

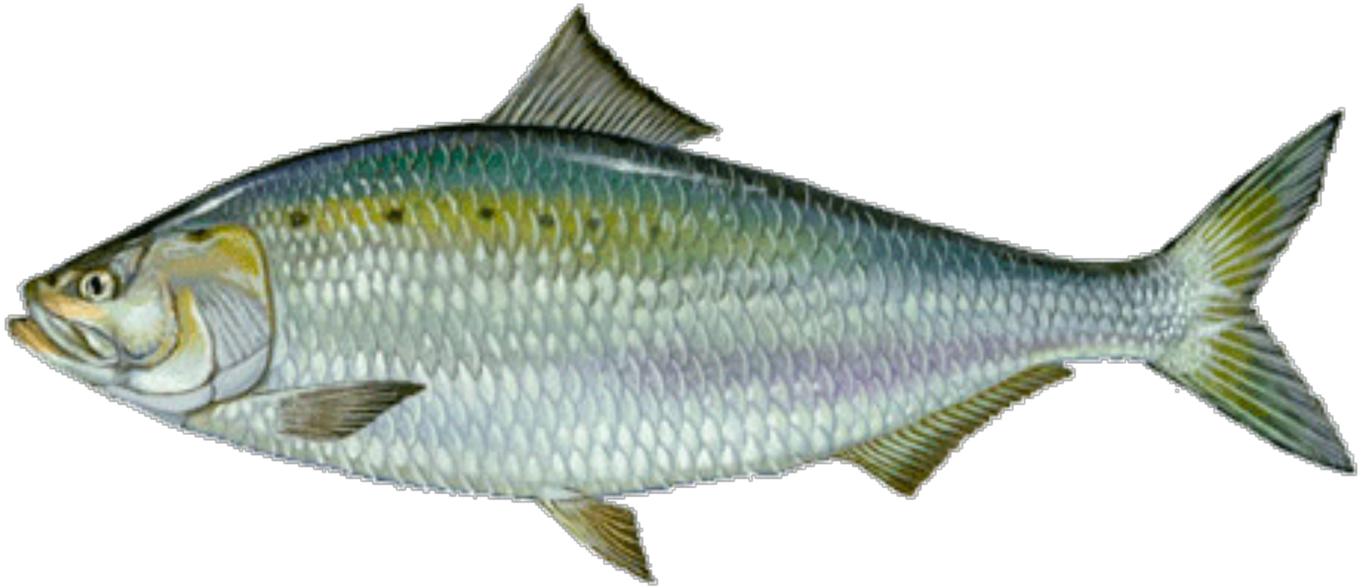
SSL - Certificate Common Name Frequency Analysis

| Common Name | Frequency Score |
|---|---|
| d3fr3g4tgwf | 2.781 |
| www.spidh.org | 3.425 |
| imap.gmx.net | 3.96 |
| www.lilawelt.net | 4.538 |
| www.paypal.com | 5.324 |
| www.google.com | 5.454 |
| www.jeffbryner.com | 6.144 |
| *.s3.amazonaws.com | 6.185 |
| s3.amazonaws.com | 6.185 |
| plesk3.gigaspark.com | 6.458 |

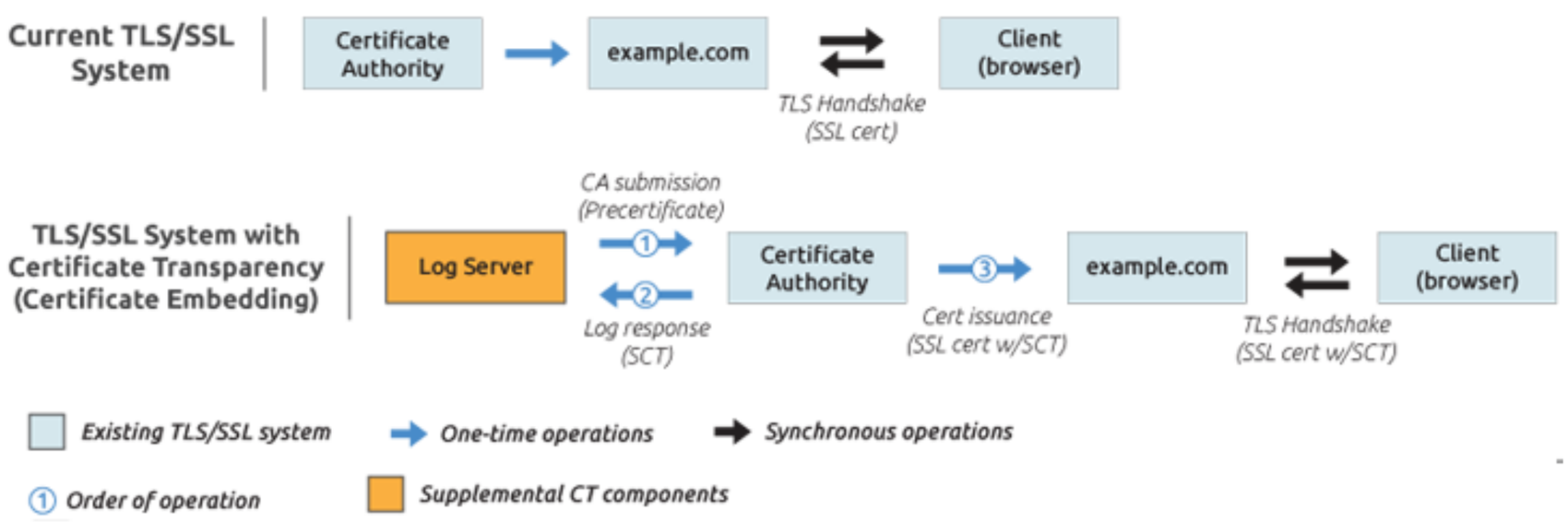
Logstash configs created by Justin Henderson (@SecurityMapper)

https://github.com/Security-Onion-Solutions/security-onion/wiki/FreqServer

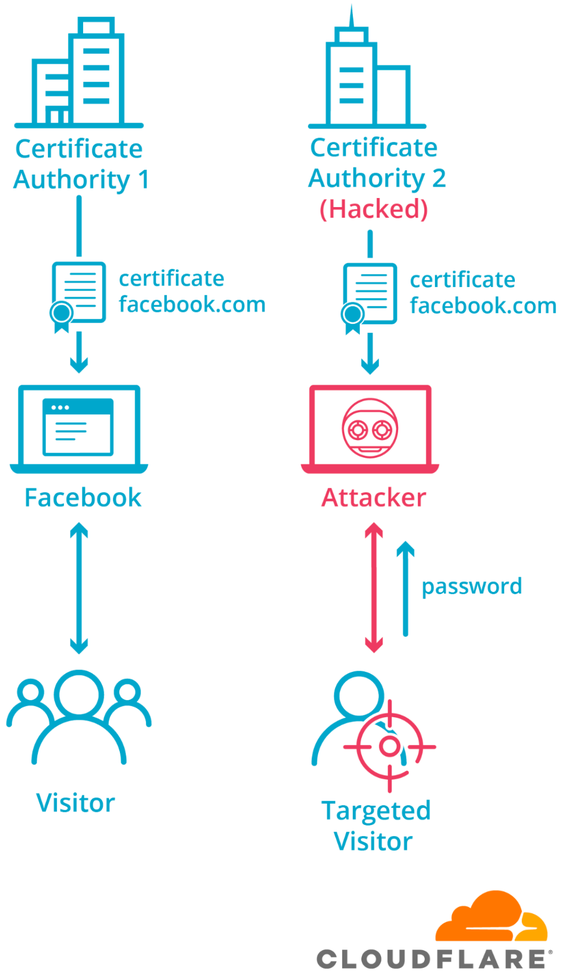
# Phishing



# Certificate Transparency

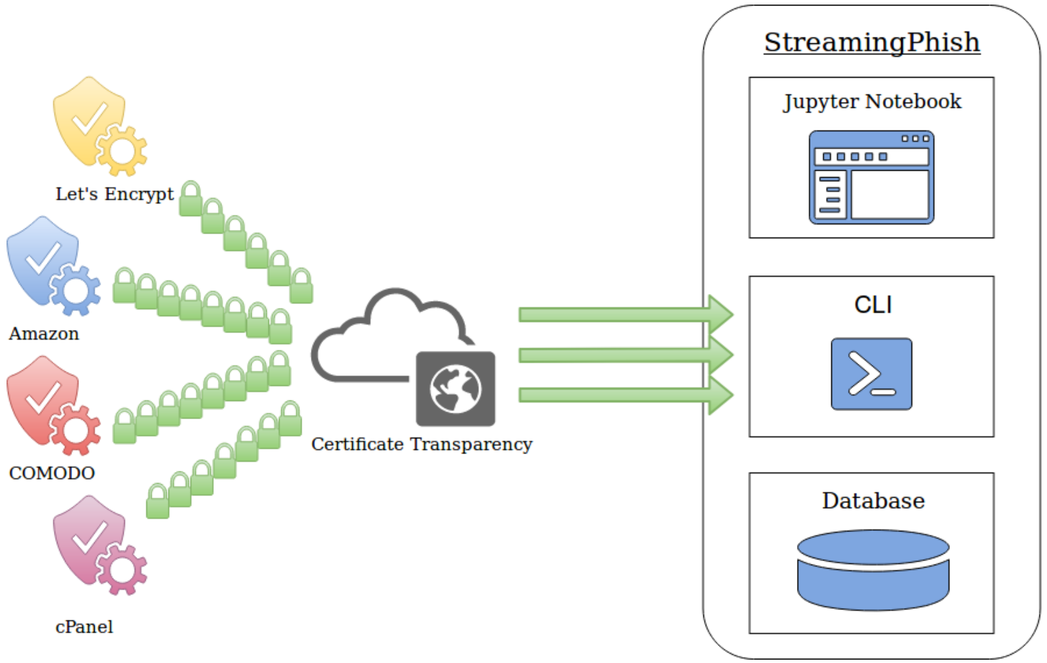


# Certificate Transparency




https://blog.cloudflare.com/introducing-certificate-transparency-and-nimbus/

# Streaming Phish





https://github.com/wesleyraptor/streamingphish


# Streaming Phish

```
[Not Phishing]  paypal.com  0.004
[Not Phishing]  apple.com 0.003
[Not Phishing]  patternex.com 0.000
[Phishing]  support-apple.xyz 0.965
[Phishing]  paypall.com 0.851
[Phishing]  pavpal-verify.com 1.000
```

- "High" threshold is 0.90 and above
- "Suspicious" threshold is between 0.90 and 0.75
- "Low" threshold is between 0.75 and 0.60

Any FQDN with a score of 0.60 or lower will not be logged.

https://github.com/wesleyraptor/streamingphish/blob/master/jupyter/notebooks/StreamingPhish.ipynb


# In Class Exercise

Please take 30 minutes and complete
**Worksheet 11: Hunting with Data Science**