Network Threat Hunter Training

Level 1

Thanks to our sponsors!

ACTIVE | COUNTERMEASURES

Other courses I'm teaching

- Advanced Network Threat Hunting
  - 12/13 - 12/16 (4 hours per day)
  - $495

https://www.antisyphontraining.com/advanced-network-threat-hunting-w-chris-brenton/

- Getting Started with Packet Decoding
  - 10/19 - 10/22
  - 12/07 - 12/10
  - Pay what you want, $25+

https://www.antisyphontraining.com/event/getting-started-with-packet-decoding-w-chris-brenton/

Before we get started

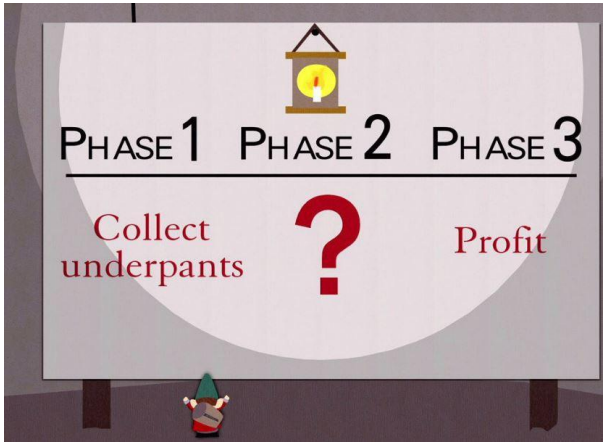
- You'll need the class VM to do the labs
- Or run the install script
- Or deploy on DigitalOcean
- Login info:
  - Name: thunt
  - Pass: aybab2u
- This should have been done before class :-)
- Slides are available on Discord

Logistics

- 10 minute break at top of each hour
- 20 minute break at 3 hour point
- Use the Discord channel for discussion
  - #acm-webcast-chat channel
- The team is monitoring for your questions

In this webcast

- I'm going to question some industry accepted standard practices
  - Because what we are doing is broken
  - And it's not getting any better
  - Will diverge from the norm
- Please keep an open mind
- Prime cognitive bias fodder



Modern attackers

- The vision of a lone hacker in the basement is dangerously outdated
- It's about profit, not mass infection
  - Attacks are now well funded
- Attacks are now targeted which means:
  - They do their homework on your environment
  - Malware is customized for your campaign
  - Attack infrastructure is customized as well
- ***Attackers innovate for each new target***

How we (try to) catch the bad guys

- Centralized log collection
- Write "signatures" to identify patterns that may indicate an attack
  - Patterns in the log messages
  - Matches against intel feeds
- Alert on signature matches
- Follow up on alerts

Limitations of system logging

- Syslog was not designed for security
  - Facility 13 is "security/log audit"
  - But rarely used in a general security context
  - More appropriate as a severity level
  - But there is no "security" severity level
- No standard for message context
  - Different platforms log events differently
  - Different applications log events differently
- Decoder ring not included

Limitations of deployment

- Every device and system?
- Are you sure?
- Are you REALLY sure?
  - I have yet to see an environment that can accurately make this claim
  - Even when you log, adversaries can disable this
- **"Fail open" system**
  - Can access Internet without logging and no alert
  - Can you detect disabled logging?

What are signatures?

- Basically RegEx for logs
- Match known bad patterns
- Because adversaries have stopped innovating and we now know all of the possible bad patterns they can use
- Oh wait…
- Sigs are also the 1990's anti-virus model

Lack of innovation

- Log RegEx matching is old
  - Older than IDS
  - Older than firewalls
- First SANS logging course early 2000's
- Not much has changed

OK to still wear parachute pants?

Is there data showing it's broken?

- Persistent versus ransomware actors
  - Detect time shouldn't count on actor disclosure
- Dwell time for persistent is on the rise

https://www.crowdstrike.com/blog/2019-services-report-key-findings-part-1/

- Dwell time ranges from 40 - 900 days

https://www.techrepublic.com/article/cybersecurity-malware-lingers-in-smbs-for-an-average-of-800-days-before-discovery/

- We are getting worse at self detection

https://investors.fireeye.com/news-releases/news-release-details/fireeye-mandiant-m-trends-2020-report-reveals-cyber-criminals

So is log review threat hunting?

- Just to review
  - Protocol can't describe security events
  - It's a fail open system
  - We try to pattern match on old attack patterns
  - False positive rates are extremely high
  - It's old technology
- The data says otherwise
- This process is clearly broken
- We need to assess new ideas and improve

I'm good, I use threat intel feeds

- Match on IP because someone said it's bad
- Also based on 1990's AV technology
- Is the data really actionable?
  - Adversaries frequently change IPs and DNS
  - Tend to use shared IP space
  - The accuracy is dependent on the reporter
- A threat intel match does not mean you've prevented an attack

Bing bot - false positive

This IP address has been reported a total of **142** times from 115 distinct sources. 23.101.169.3 was first reported on June 13th 2018, and the most recent report was **2 days ago**.

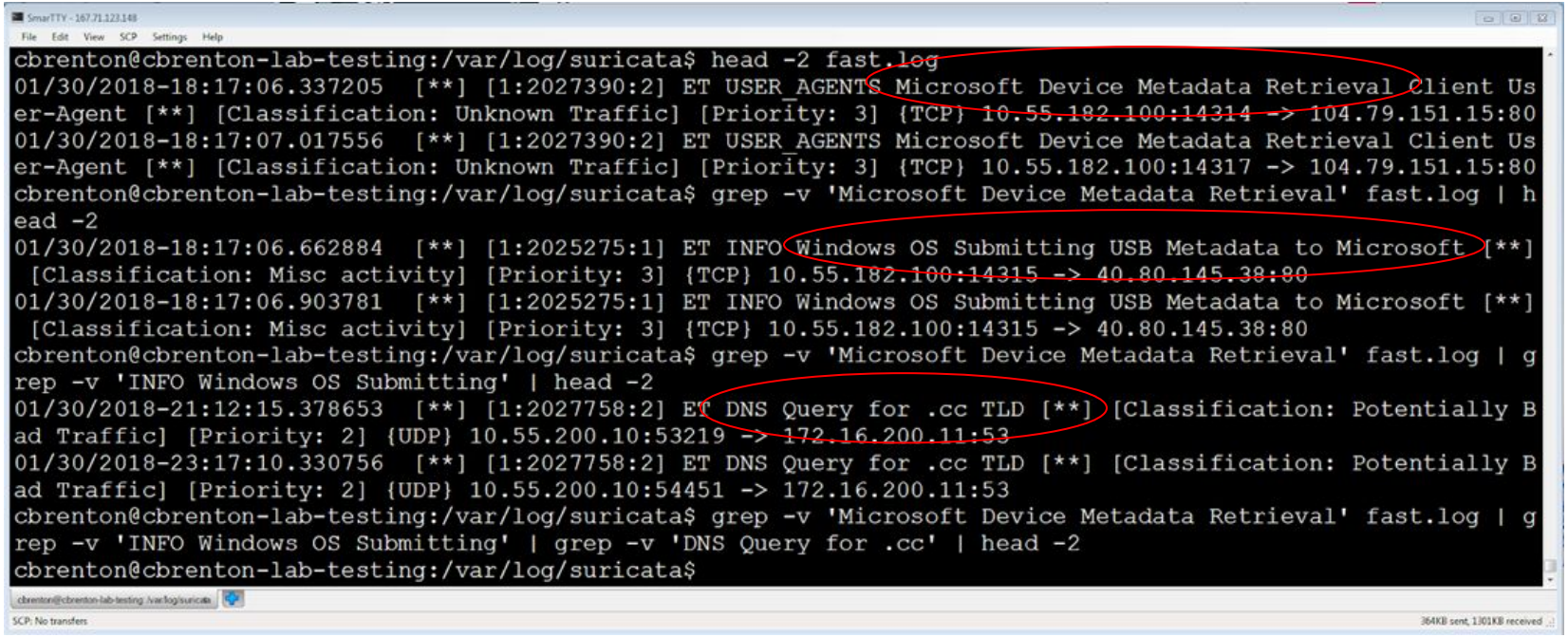
Recent Reports: We have received reports of abusive activity from this IP address within the last week. It is potentially still actively engaged in abusive activities.

Reporter	Date	Comment	Categories
✔ Anonymous	11 Jan 2019		Web Spam
Anonymous	03 Jan 2019		Web Spam Hacking Brute-Force
Anonymous	28 Dec 2018	Bing bot out of control. Still attempting to hit my site, eve n when banned.	Web Spam Bad Web Bot
Anonymous	26 Dec 2018	2200 blocked hits on my blog. Wordfence has blocked it. Wasn't sure what category to select (Br ... show more	Brute-Force
Anonymous	25 Dec 2018	This ip showing as Microsoft Azure, location Chicago ha s been on all three of my blogs at Blogger an ... show more	Blog Spam
Anonymous	23 Dec 2018	just blockd it	Brute-Force Bad Web Bot
Anonymous	21 Dec 2018	1465 website hits in one day - not sure why	Brute-Force
✔ Deny_IP	18 Dec 2018	US bad_bot	Web App Attack
Anonymous	16 Dec 2018	Runs all Javascript on page, showing up in Google Anal ytics and ad reporting as an individual unique ... show more	Web Spam Bad Web Bot

Sample threat feed

```
################################################################
## Master Feed of known, active and non-sinkholed C&Cs IP
## addresses
##
## Feed generated at: 2019-07-11 15:12
##
## Feed Provided By: John Bambenek of Bambenek Consulting
## jcb@bambenekconsulting.com // http://bambenekconsulting.com
## Use of this feed is governed by the license here:
## http://osint.bambenekconsulting.com/license.txt
##
## For more information on this feed go to:
## http://osint.bambenekconsulting.com/manual/c2-ipmasterlist.txt
##
## All times are in UTC
################################################################
5.79.79.211,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
23.105.99.15,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
23.107.124.53,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
23.110.13.197,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
23.236.62.147,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
23.89.102.179,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
23.89.20.107,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
27.124.28.149,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
31.11.33.228,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
35.169.58.188,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
35.186.238.101,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
43.230.142.125,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
43.241.196.105,IP used by banjori C&C,2019-07-11 15:03,http://osint.bambenekconsulting.com/manual/banjori.txt
```

Can I threat hunt with my NIDS?

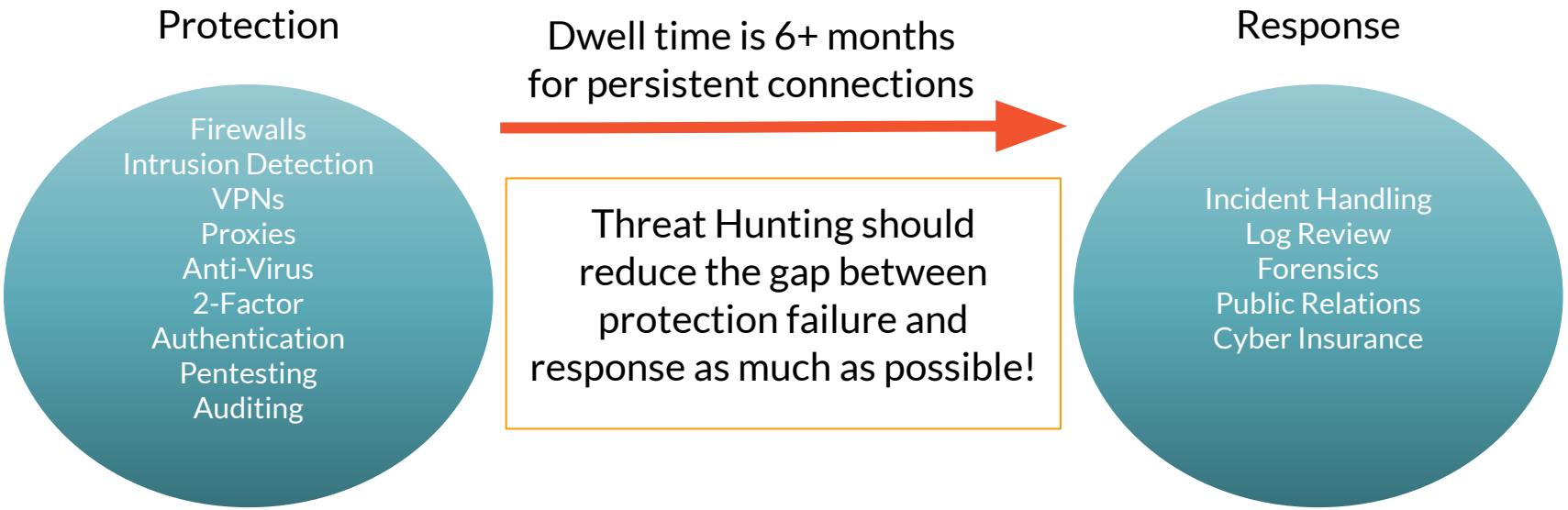


But empire and dnscat2 were missed

What Threat Hunting should be

- A proactive validation of all systems connected to the organization's network
- Needs to include all systems
  - Desktops, laptops, cellphones, tablets
  - Servers, network gear, printers
  - IoT, IIoT, any type of Internet "Thing"
- Execute without making assumptions
- Deliverable is a compromise assessment

The Purpose of Threat Hunting



The process of threat hunting

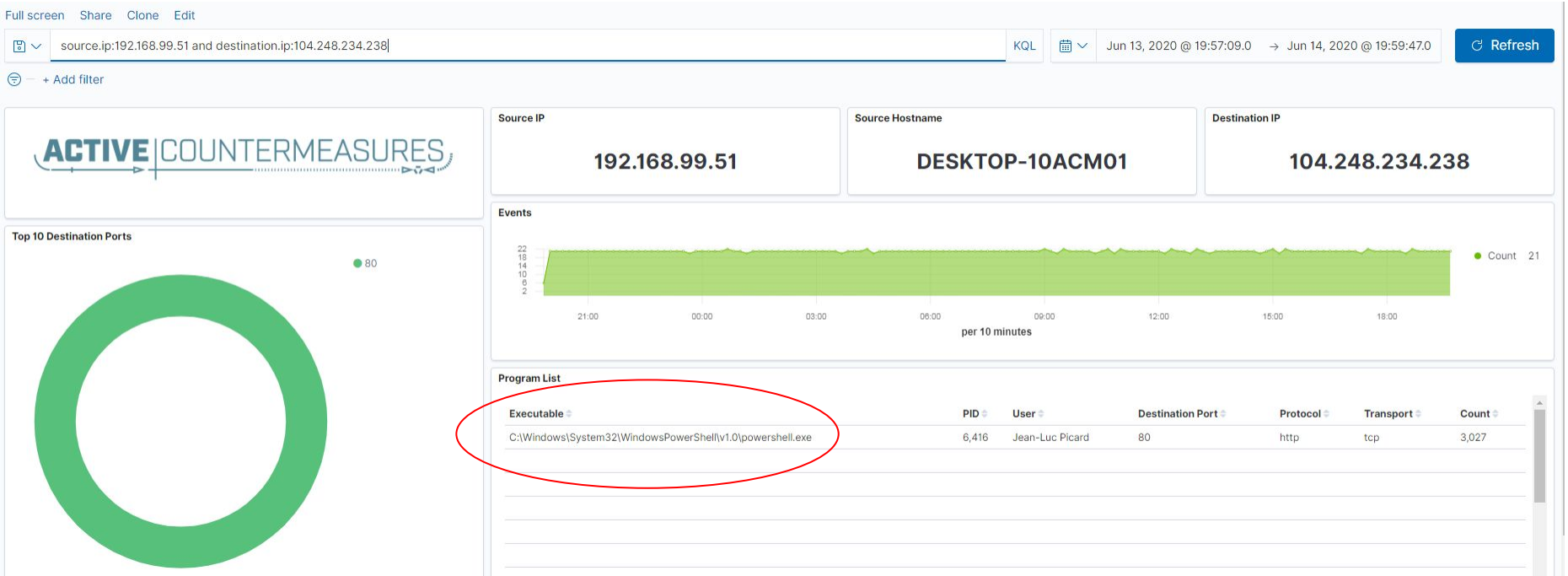
- Review the integrity of every device
  - Desktops, servers, network gear, IoT, IIoT, etc.
- Generate one of 3 dispositions
  - I'm pretty certain the system is safe
  - I'm pretty certain the system is compromised
  - I'm unsure of state so will collect additional info to derive one of the above two results
- Leverage context for host log review

Proposal - Start with the network

- The network is the great equalizer
  - You see everything, regardless of platform
  - High level assessment of the terrain
- You can hide processes but not packets
- Malware is usually controlled
  - Which makes targeting C2 extremely effective
  - Identify compromise when C2 "calls home"
  - Must be frequent enough to be useful
- Wide view so you can target from there

Start on the network

THEN pivot to the system logs



C2 Detection Techniques

Where to Start

- Traffic to and from the Internet
  - Monitor internal interface of firewall
- Packet captures or Bro/Zeek data
- Analyze in large time blocks
  - More data = better fidelity
  - Minimum of 12 hours, 24 is ideal
- Analyze communications in pairs
  - Every outbound session passing the firewall
  - Ignore internal to internal (high false positive)

Threat score system

- Our job is to disposition IPs
- How do you know when to make a choice?
- A numeric system can help guide you
  - Score of 0 = system is safe
  - Score of 100 = system is compromised
- Score modifiers
  - Major - A clue that strongly indicates integrity state
  - Minor - A clue that peripherally indicates integrity state

Score examples

- Major score modifier
  - Persistency of connection
  - Unexpected protocol on well known port
  - Moving lots of data to a threat intel IP address
- Minor modifier
  - Moving lots of data to a random IP
  - Unique client signature
  - self signed digital certificate
  - EV digital certificate (reduce score)

Threat hunting process order

- Persistent connection?
  - No = No further action required
  - Yes = Go to next step
- Abnormal protocol behaviour?
- Reputation check of external IP
- Investigation of internal IP
- Disposition
  - Safe = whitelist
  - Compromised = incident handling

Does targeting C2 have blind spots?

- Attackers motivated by gain
  - Information
  - Control of resources
- Sometimes "gain" does not require C2
  - Just looking to destroy the target
  - Equivalent to dropping a cyber bomb
  - We are talking nation state at this level
- NotPetya
  - Worm with no C2 designed to seek and destroy

Techniques Vs Methodology

- We are going to deep dive on finding C2
- It's important to understand what needs to happen "under the hood"
- Some of these techniques don't scale
  - Manually breaking out connection pairs
  - But that's OK
- Will focus on tools in a later module
- For now, focus on just the techniques

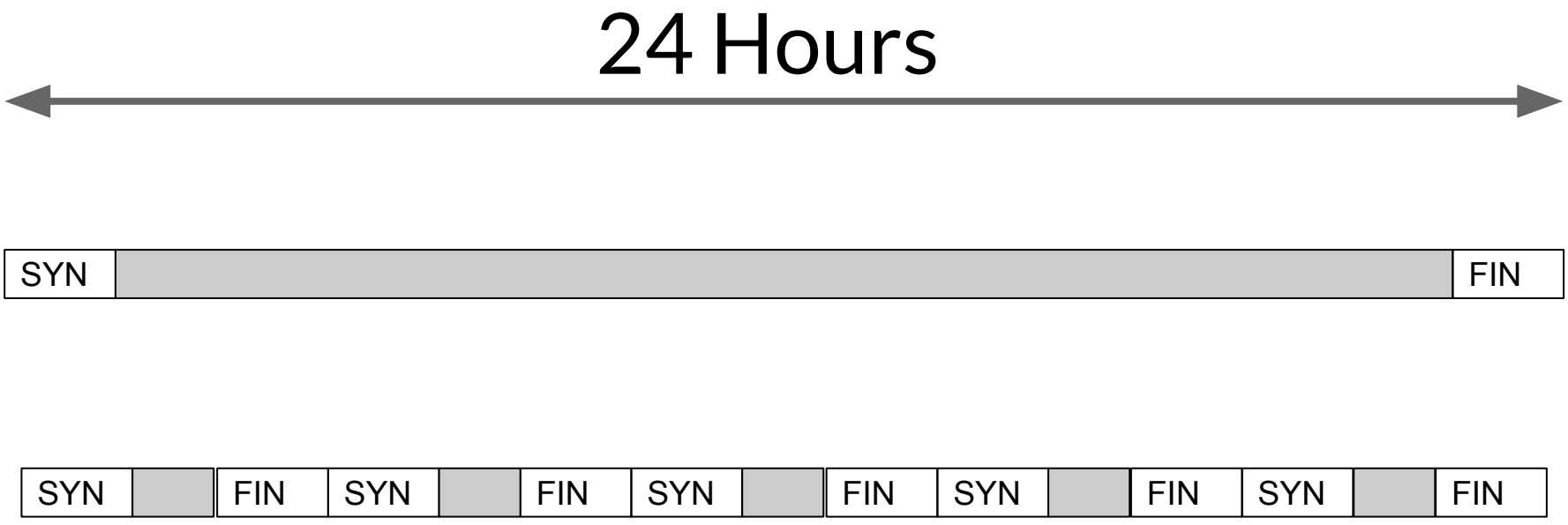
Bad guys Vs. Red Teams

- Bad guys = C2 is part of a business model
- Red team = C2 is why they get paid
- Much harder to detect red team C2 than the real bad guys
  - In the wild, most evil C2 beacons <= 1/minute
  - Red team on long term contract <= 1/week
- Focus will be on the bad guys

Long connections

- You are looking for:
- Total time for each connection
  - Which ones have gone on the longest?
- Cumulative time for all pair connections
  - Total amount of time the pair has been in contact
- Can be useful to ignore ports or protocols
  - C2 can change channels

Long connection examples



Connection timing from Bro/Zeek

```
cbrenton@zeek-3-3-rc2:/opt/bro/logs/2019-07-17$ zcat conn.00\:00\:00-01\:00\:00.log.gz | head -10
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   conn
#open   2019-07-17-00-00-00
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       proto   ser
vice    duration        orig_bytes      resp_bytes      conn_state      local_orig      local_resp
missed_bytes    history orig_pkts       orig_ip_bytes   resp_pkts       resp_ip_bytes   tunnel_pare
nts
#types  time    string  addr    port    addr    port    enum    string  interval        count   cou
nt      string  bool    bool    count   string  count   count   count   count   set[string]
1563321592.266216       CRP5W73KxGUYtn2XQh      185.176.27.30   48086   104.248.191.205 20391   tcp
-       0.265051        0       0       REJ     F       F       0       SrR     2       80      1
40      (empty)
1563321592.266218       CjZ8aQ2AoHDrsheUAj      185.176.27.30   48086   104.248.191.205 20391   tcp
-       0.265051        0       0       REJ     F       F       0       SrR     2       80      1
40      (empty)
cbrenton@zeek-3-3-rc2:/opt/bro/logs/2019-07-17$
```

cat conn.log | column -t | less -S

```
#set_separator     ,
#empty_field       (empty)
#unset_field       -
#path              conn
#open              2019-11-20-14-10-06
#fields            ts                  uid                 id.orig_h  id.orig
#types             time                string              addr       port
1517336042.090842  CgqsIo2Hq3bGIrXD8k  10.55.182.100       14291      10.233.
1517336042.279652  CWVu5C2AkLs8nB7657   192.168.88.2        55638      165.227
1517336052.713711  CT1q291fksTSxPdUA7   10.55.100.111       49544      165.227
1517336043.354471  CO0pDK2yijeceiToMc   192.168.88.2        28736      165.227
1517336053.289571  CVSp971KXPXmTyrC9j   10.55.100.111       49544      165.227
1517336048.965466  CUSHUX2I19K21KXI81   10.55.182.100       14266      192.30.
1517336044.438943  ChVkR83XE3MpwdAJb6   192.168.88.2        42647      165.227
1517336045.517672  CMlJtg2HCHT1MMgeH4   192.168.88.2        20440      165.227
1517336051.090978  C1eIfe1G9iN6qOMEEf   10.55.182.100       14291      10.233.
1517336046.587421  Cxddlb4tdm5OLOFzel   192.168.88.2        39912      165.227
1517336047.665136  C4wzjK29XgyJaLkbui   192.168.88.2        42658      165.227
1517336057.180792  CysQps3yTdVaPbv0fk   10.55.100.111       49545      165.227
1517336057.742766  COZfRhUFpUK9a5YCl    10.55.100.111       49545      165.227
1517336048.741779  CZu9YJ1eNPDOYeJSI    192.168.88.2        16271      165.227
.
```

Cumulative talk time with Zeek

```
thunt@thunt-labs:~/lab1$ cat conn.log | zeek-cut id.orig_h id.resp_h duration | sort
| grep -v -e '^$' | grep -v '-' | datamash -g 1,2 sum 3 | sort -k 3 -rn | head
192.168.99.51   167.71.97.235   86389.659357
192.168.99.51   52.179.219.14   4067.394413
192.168.99.51   52.184.217.56   2936.172839
192.168.99.51   52.184.216.246  2825.858
192.168.99.52   239.255.255.250 2507.626732
fe80::d048:42e0:8448:187c       ff02::c 2434.977049
192.168.99.51   239.255.255.250 2374.546469
fe80::2126:bcd7:16f4:8cdb       ff02::c 2368.234679
192.168.99.51   13.107.5.88     1317.047871
192.168.99.51   52.167.249.196  868.46966
thunt@thunt-labs:~/lab1$
```

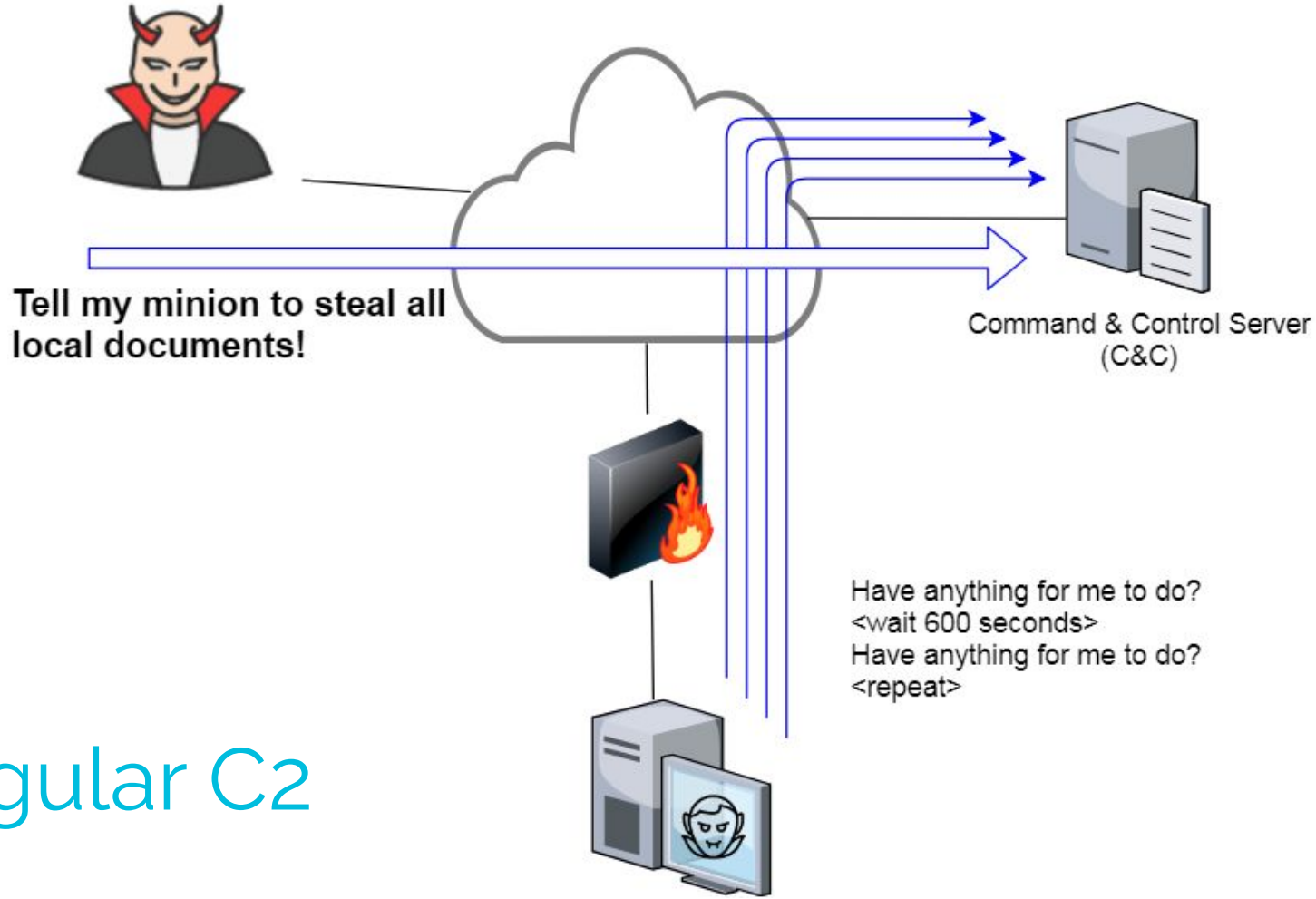
What about firewalls?

- Surprisingly hard to get this info
- "Timing" tends to be TTL, not duration
- BSD
  - pftop - output connection age in seconds
- Junos
  - show security flow session extensive node all
  - Duration in seconds

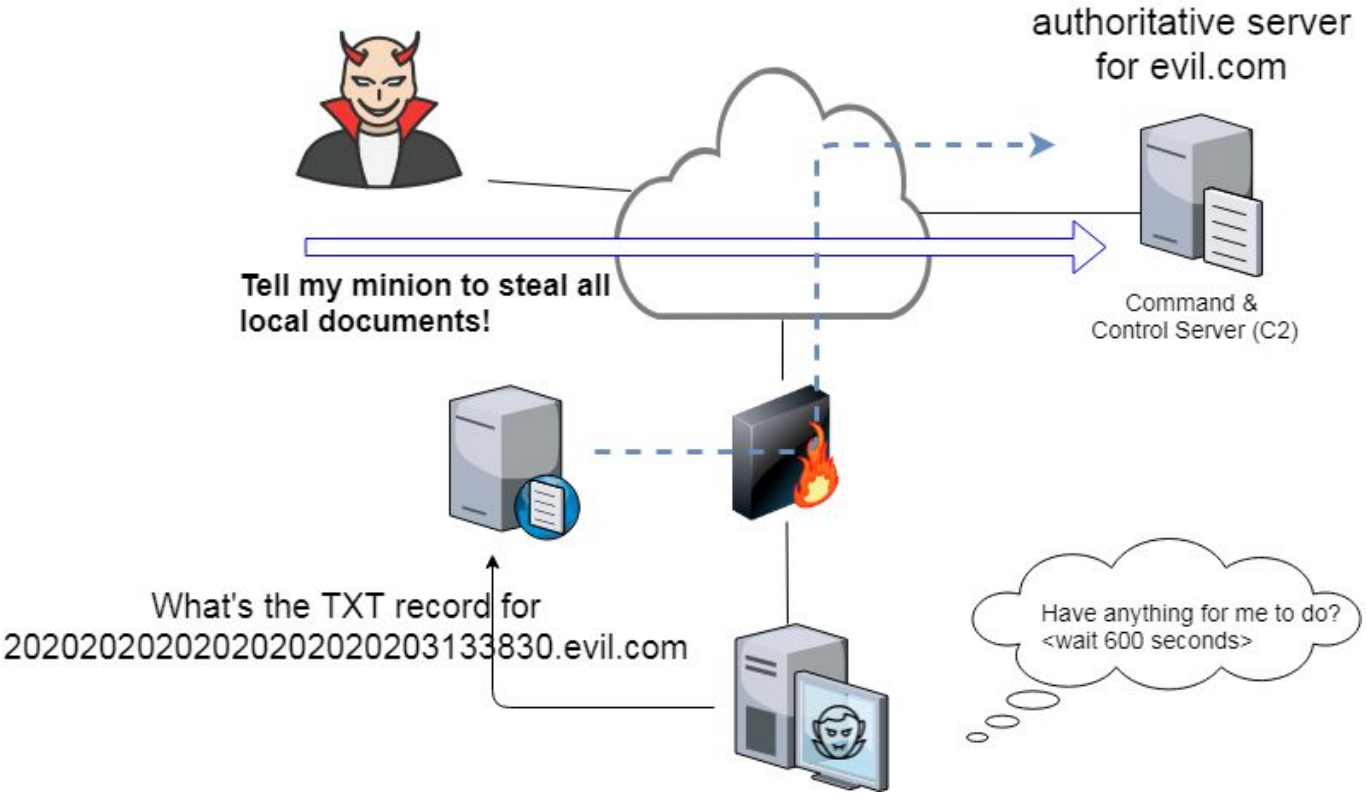
What is a beacon?

- Repetitive connection establishment between two IP addresses
  - Easiest to detect
- Repetitive connection establishment between internal IP and FQDN
  - Beacon broken up over multiple IP's
    - Usually a CDN provider
  - Target IPs also destination for legitimate traffic
  - Far more difficult to detect

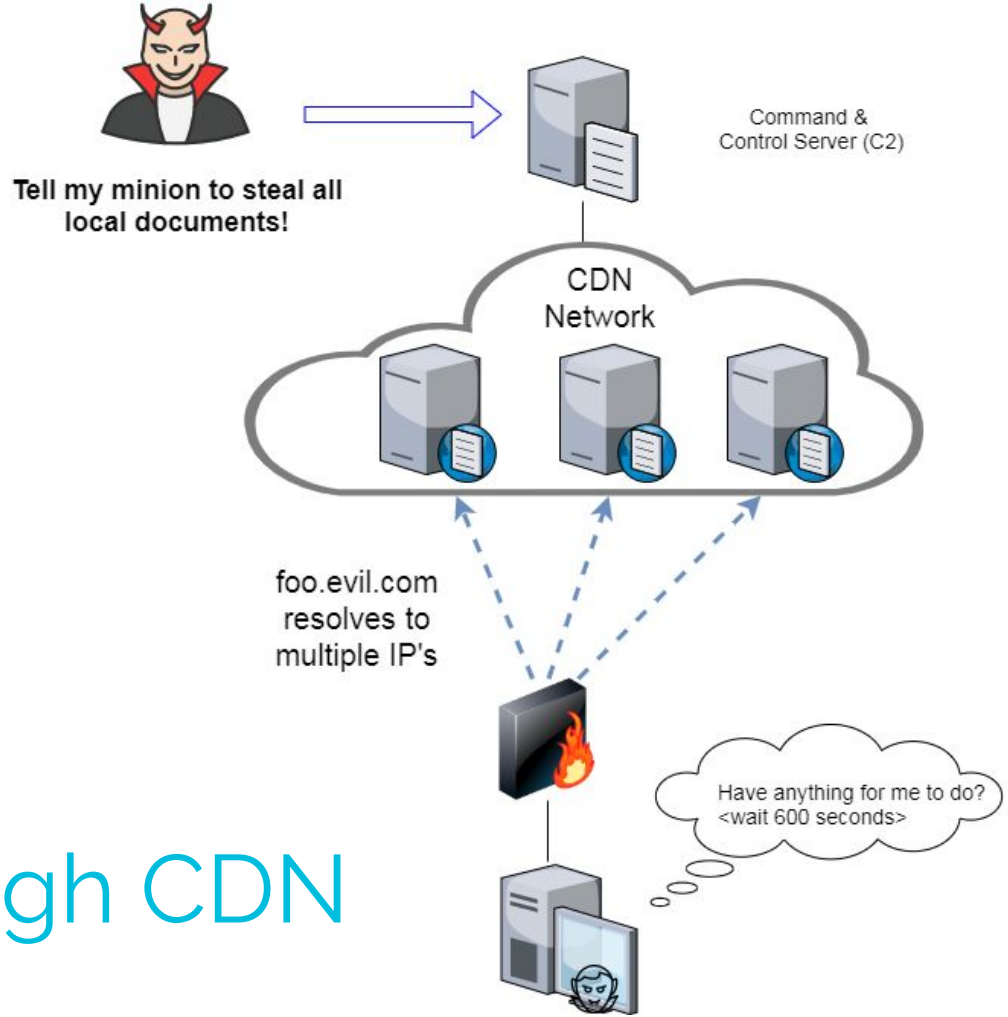
Regular C2


Tell my minion to steal all local documents!
Command & Control Server (C&C)
Have anything for me to do?
<wait 600 seconds>
Have anything for me to do?
<repeat>

C2 over DNS


authoritative server
for evil.com
Tell my minion to steal all
local documents!
Command &
Control Server (C2)
What's the TXT record for
20202020202020202020203133830.evil.com
Have anything for me to do?
<wait 600 seconds>

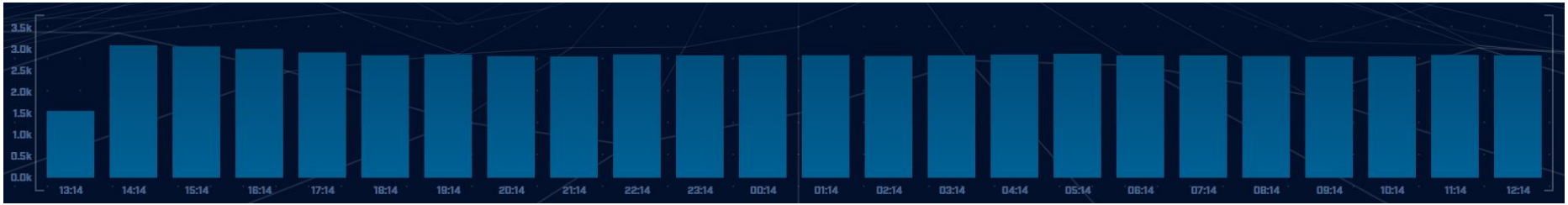
C2 through CDN



Beacon detection based on timing

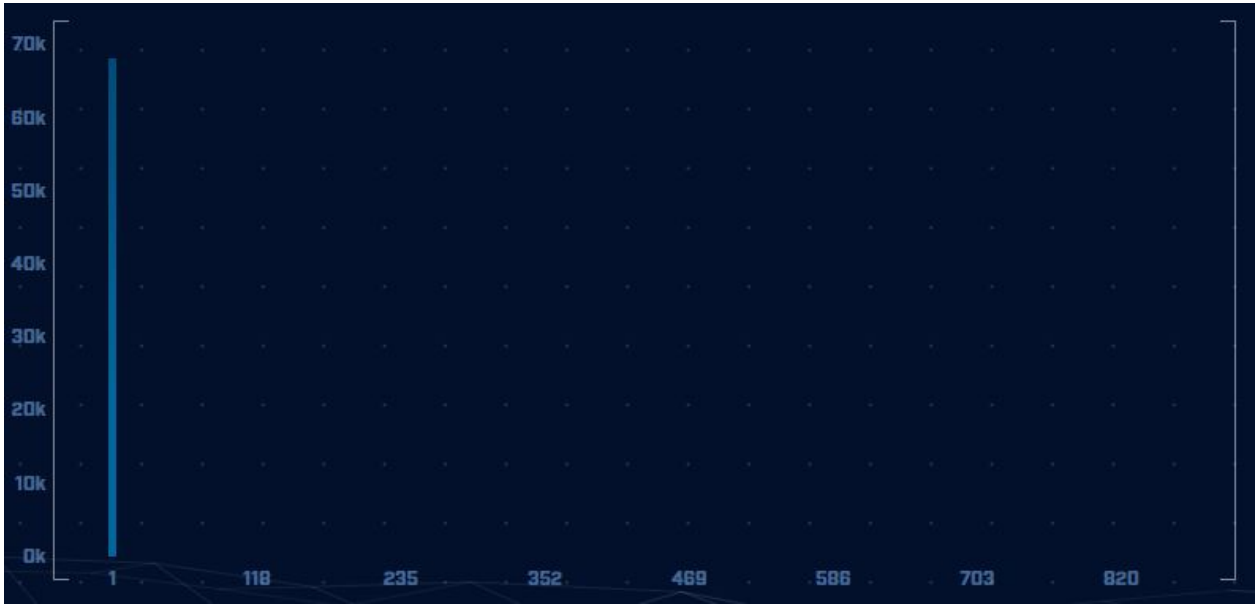
- May follow an exact time interval
  - Technique is less common today
  - Detectable by k-means
  - Potential false positives
- May introduce "jitter"
  - Vary connection sleep delta
  - Avoids k-means detection
  - False positives are extremely rare
- Short enough delta for terminal activities

Connection quantity VS time



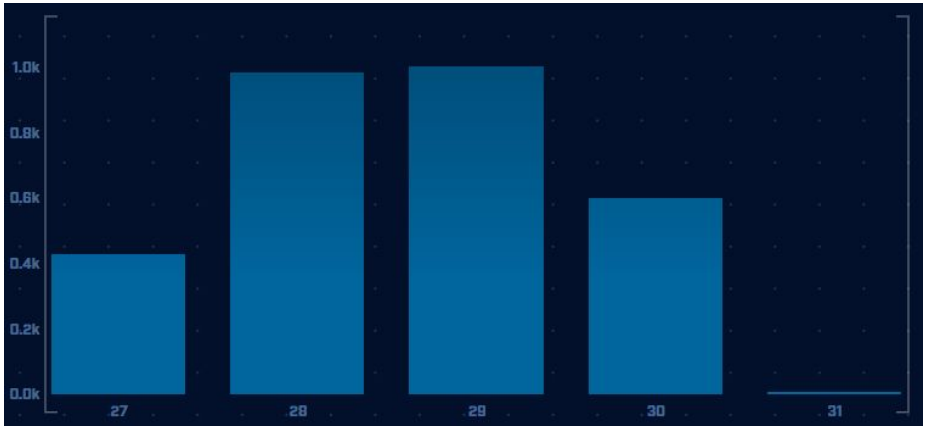
Each bar represents the number of times the source connected to the destination during that one hour time block

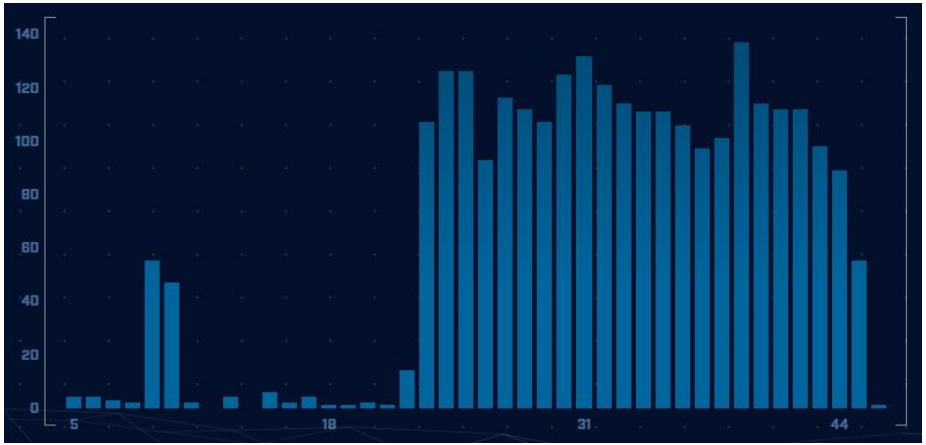
Connect time deltas with no jitter



How often a specific time delta was observed

Connection time deltas with jitter



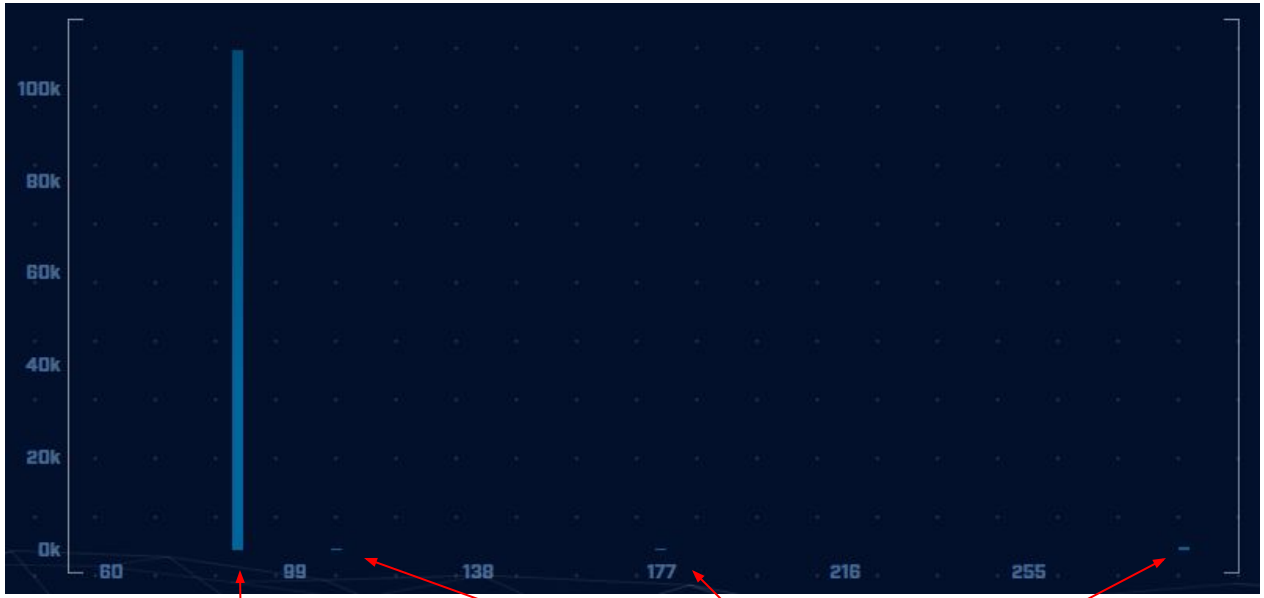


Cobalt Strike will typically produce a bell curve

Detection based on session size

- Focuses on detection of the heartbeat
- Variations from the heartbeat indicate activation of C2 channel
- Session size can help reveal info regarding commands being issued
- Possible to randomly pad but this is extremely rare

Session size analysis



Heartbeat

Activation

Session size analysis with Zeek

```
ritabeakerlab@ritabeakerlab:~/lab1$ cat conn*.log | bro-cut id.orig_h id.resp_h orig_bytes | grep 68.183.1
38.51 | sort | uniq -c | sort -rn | head
   2868 10.0.2.15       68.183.138.51   546
      1 10.0.2.15       68.183.138.51   -
ritabeakerlab@ritabeakerlab:~/lab1$
```

Heartbeat only
Not yet in use!

Detecting beacons with jitter

- Easier to detect when normalized out over long periods of time
  - Average the time deltas for each hour
  - Plot over 24 hours
- Should make a beacon even more suspect
  - False positives don't obscure their beacon timing
  - High probability of being evil

Is there a business need?

Potential false positives

- False positives will not show signs of jitter
- Some common false positives:
  - NTP
  - Windows message bus, widgets, etc.
  - Some remote desktop tools
- More of a miscatorization:
  - Long connections with pauses longer than the timeout of your monitoring tool
  - Zeek timeout defaults to 5 minutes
  - Some MS traffic pauses for 35 minutes

C2 Detection Techniques Part 2

Minor modifiers for review

- Protocol compliance
- External IP address
- Internal IP address

Unexpected app or port usage

- There should be a business need for all outbound protocols
- Research non-standard or unknown ports
  - TCP/5222 (Chrome remote desktop)
  - TCP/5800 & 590X (VNC)
  - TCP/502 (Modbus)

Unknown app on standard port

- C2 wants to tunnel out of environment
  - Pick a port likely to be permitted outbound
  - Does not always worry about protocol compliance
- Check standard ports for unexpected apps
  - Indication of tunneling
- Different than app on non-standard port
  - This is sometimes done as "a feature"
  - Example: SSH listening on TCP/2222

Bro/Zeek decodes many apps

- Detect over 50 applications
  - HTTP, DNS, SIP, MYSQL, RDP, NTLM, etc. etc.
- Fairly easy to add new ones
  - Example: HL7 if you are in healthcare
- Checks all analyzers for each port
- Does not assume WKP = application

Bro/Zeek example

```
$ cat conn.log | bro-cut  id.orig_h id.resp_h id.resp_p proto
service orig_ip_bytes resp_ip_bytes

183.131.82.99    104.248.191.205  22     tcp  ssh  1923   0
112.85.42.229    104.248.191.205  22     tcp  -    344    0
104.248.191.205  67.207.67.3      53     udp  dns  42     126
81.22.45.150     104.248.191.205  7180   tcp  -    80     40
110.49.40.4      104.248.191.205  445    tcp  -    52     40
81.22.45.150     104.248.191.205  7404   tcp  -    80     40
```

Unexpected protocol use

- Attackers may bend but not break rules
- This can result in:
  - Full protocol compliance
  - Abnormal behaviour
- Need to understand "normal"
  - For the protocol
  - For your environment

Example: Too many FQDNs

- How many FQDNs do domains expose?
  - Most is < 10
  - Recognizable Internet based vendors 200 - 600
    - Microsoft
    - Akamai
    - Google
    - Amazon
- Greater than 1,000 is suspicious
- Could be an indication of C2 traffic

Detecting C2 over DNS

- Capture all DNS traffic
  - Capture tool of your choice
  - Longer the capture time, the better
- Filter so it's DNS traffic only
- Extract to text so we can sort and count
- Review total FQDNs per domain

Counting FQDNs per domain

```
cbrenton@cbrenton-lab-testing:~/lab-thunt$ tshark -r thunt-lab.pcapng -T fields -e dn
s.qry.name | sort | uniq | rev | cut -d '.' -f 1-2 | rev | sort | uniq -c | sort -rn
| head -10
  62468 r-1x.com
    154 akamaiedge.net
    125 akadns.net
    121 edgekey.net
    104 amazonaws.com
     67 microsoft.com
     51 dynect.net
     45 parsely.com
     44 akam.net
     43 cloudfront.net
cbrenton@cbrenton-lab-testing:~/lab-thunt$
```

Breaking it down

```
cbrenton@cbrenton-lab-testing:~/lab-thunt$ tshark -r thunt-lab.pcapng -T fields -e dn
s.qry.name | sort | uniq | head -4

0000011239458783cf.dnsc.r-1x.com
00000176d2f1ce66e2.dnsc.r-1x.com
0001011239458783cf.dnsc.r-1x.com
cbrenton@cbrenton-lab-testing:~/lab-thunt$ tshark -r thunt-lab.pcapng -T fields -e dn
s.qry.name | sort | uniq | rev | head -4

moc.x1-r.csnd.fc3878549321100000
moc.x1-r.csnd.2e66ec1f2d67100000
moc.x1-r.csnd.fc3878549321101000
cbrenton@cbrenton-lab-testing:~/lab-thunt$ tshark -r thunt-lab.pcapng -T fields -e dn
s.qry.name | sort | uniq | rev | cut -d '.' -f 1-2 | rev | head -4

r-1x.com
r-1x.com
r-1x.com
```

Show all instances of unique FQDNs queried

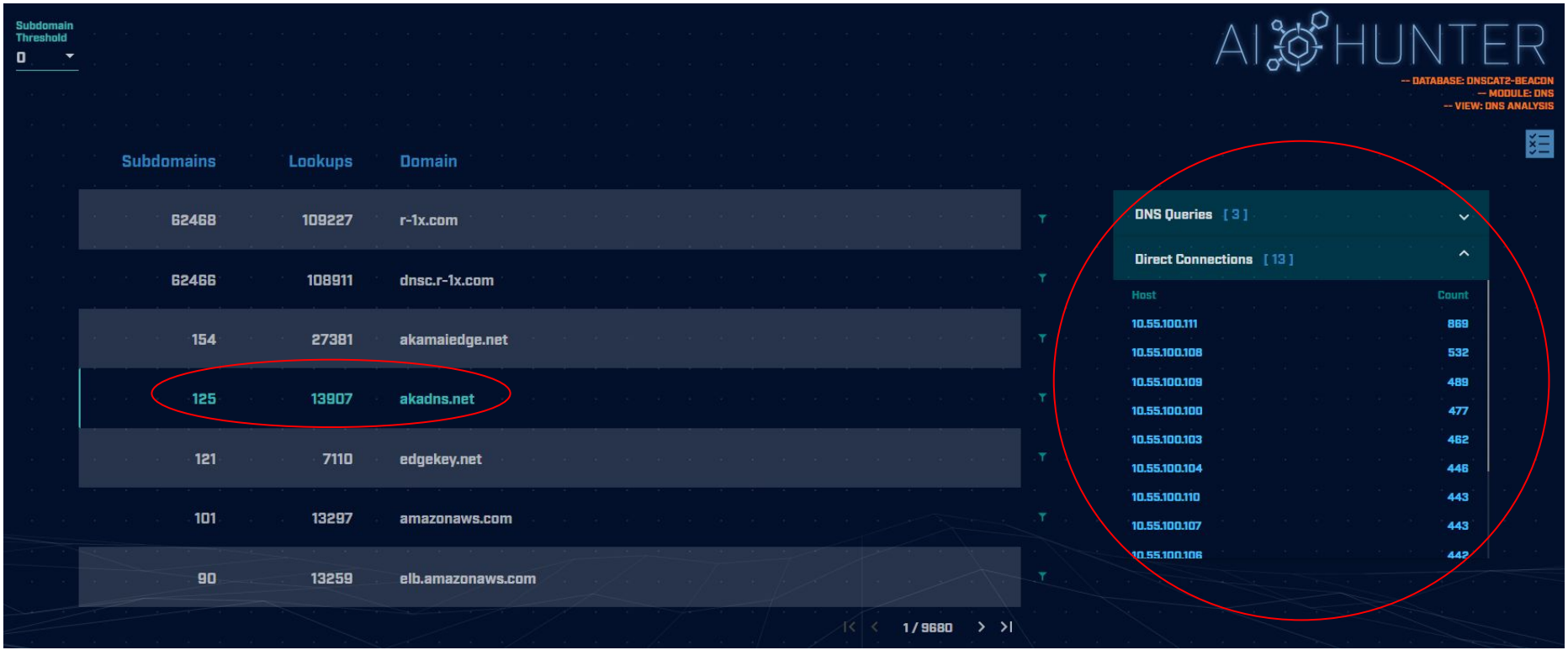
Reverse the characters on the line so we can "cut" first two fields

Cut out subdomains and reverse characters on the line. We can now count the number of unique FQDNs queried per domain

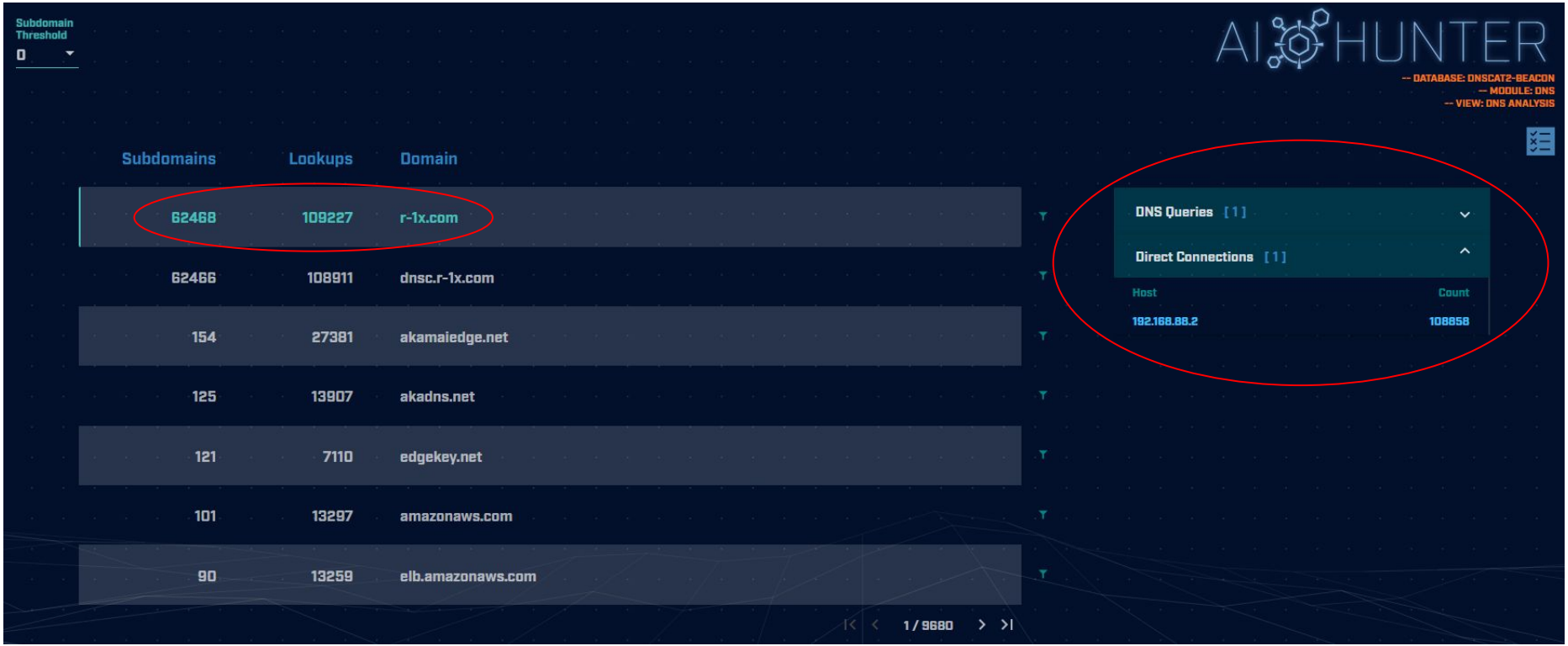
Bonus checks on DNS

- Check domains with a lot of FQDNs
- Get a list of the IPs returned
- Compare against traffic patterns
  - Are internal hosts visiting this domain?
  - Is it just your name servers?
- Unique trait of C2 over DNS
  - Lots or FQDN queries
  - But no one ever connects to these systems

Normal DNS query patten



Things that make you go "hummm"



Look for unique HTTP user agents

```
cbrenton@aih-3-3-rc2:~/test/testing$ cat http.08_33_18-09_00_00.log | bro-cut user_agent
 | sort | uniq -c | sort
      1 -
      1 Python-urllib/3.5
     22 Microsoft-WNS/10.0
     26 Microsoft-CryptoAPI/10.0
     30 Microsoft BITS/7.8
     55 Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.17134.590
     72 Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
cbrenton@aih-3-3-rc2:~/test/testing$
cbrenton@aih-3-3-rc2:~/test/testing$
cbrenton@aih-3-3-rc2:~/test/testing$ grep Python http.08_33_18-09_00_00.log
1552574001.145136       CLLPdJ1nLAOdIIwyHe      10.55.254.107   42292   91.189.95.15
80      1       GET     changelogs.ubuntu.com   /meta-release-lts       -       1.1
Python-urllib/3.5       0       4386    200     OK      -       -       (empty) -
-       -       -       -       -       FhGf5d4pejzo7Ob3ll      -       text/plain
cbrenton@aih-3-3-rc2:~/test/testing$ _
```

Unique SSL Client Hello: Zeek + JA3

SSL/TLS Hash	Seen	Requests	Sources
5e573c9c9f8ba720ef9b18e9fce2e2f7	1	clientservices.googleapis.com	10.55.182.100
bc6c386f480ee97b9d9e52d472b772d8	2	clients4.google.com, 556-emw-319.mktoresp.com	10.55.182.100
f3405aa9ca597089a55cf8c62754de84	2	builds.cdn.getgo.com	10.55.182.100
28a2c9bd18a11de089ef85a160da29e4	2	mediaredirect.microsoft.com	10.55.100.105, 10.55.182.100
08bf94d7f3200a537b5e3b76b06e02a2	4	files01.netgate.com	192.168.88.2

Check destination IP address

- Start simple
  - Who manages ASN?
  - Geolocation info?
  - IP delegation
  - PTR records
- Do you recognize the target organization?
  - Business partner or field office
  - Current vendor (active status)
- Other internal IP's connecting?

Check threat intel on target IP

- Need to understand:
  - When was the record first created?
  - Why was the record created?

https://www.abuseipdb.com/check/<ip address>
https://dnslytics.com/ip/<IP address>
https://transparencyreport.google.com/safe-browsing/search?url=<IP, FQDN or URL>

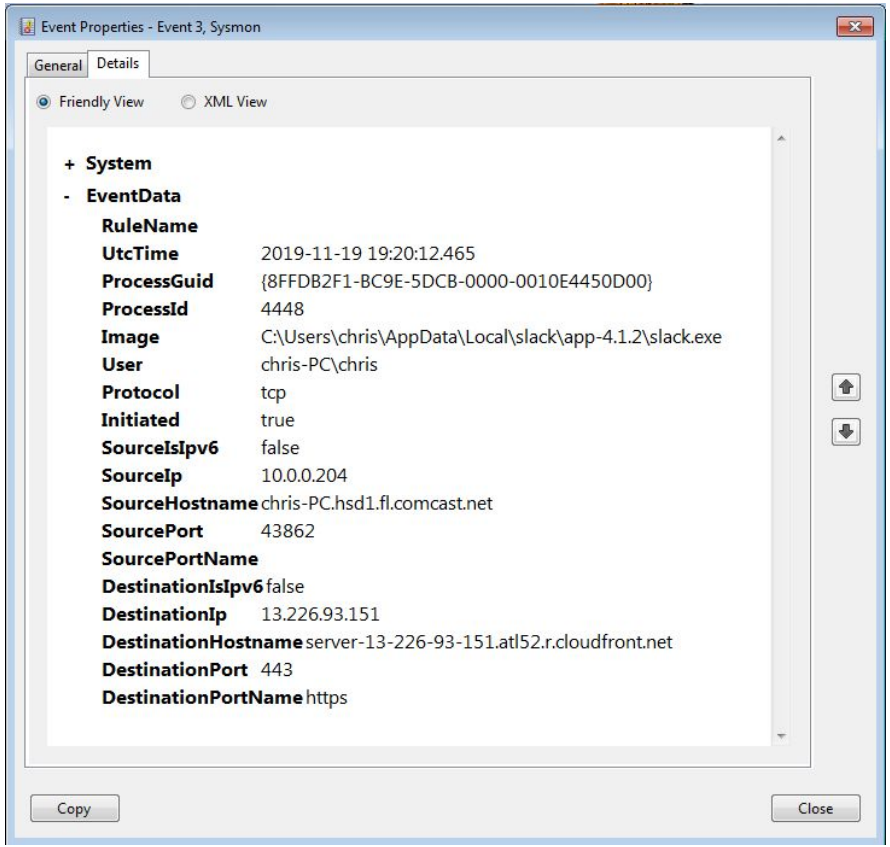
Internal system

- Info available varies greatly between orgs
- Inventory management systems
- Security tools like Carbon Black
- OS projects like BeaKer
- Internal security scans
- DHCP logs
- Login events
- Passive fingerprinting

Leverage internal host logging

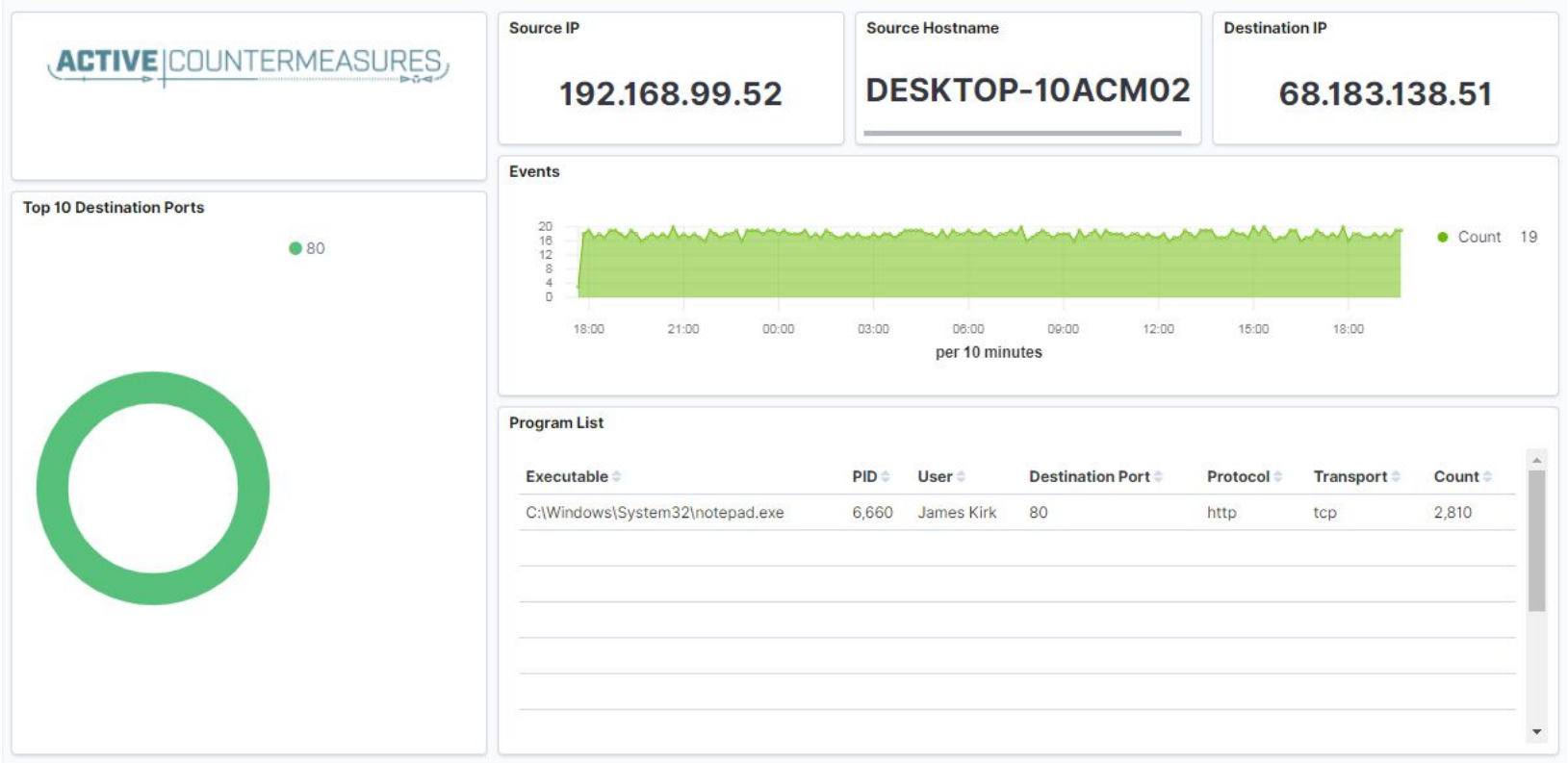
- Network shows suspicious traffic patterns
- Use this data to pivot to host logs
- Filter your logs based on:
  - Suspect internal host
  - Timeframe being analyzed
- Anything stand out as unique or odd?

Sysmon Event ID Type 3's



Map outbound connections to the applications that created them.

Sysmon Type 3 + BeaKer



But I have no system logs!

- Might be a good time to start collecting them
- Full packet captures from system
- Apply additional network tools to collect more data

What next?

- Assign points to connection persistence
  - How certain are you that it's automated?
- Assign points to the protocol review
- Assign points to the endpoint research
- Remember negative points are OK
- Add the score, how certain are you?
  - Safe = add to whitelist
  - Scary = Trigger incident response
  - Still unsure = Collect more data

C2 Detection Tools

tcpdump

- What's it good for?
  - Lightweight packet capturing tool
  - Cross platform support (windump on Windows)
- When to use it
  - Audit trail of all traffic
  - Can also filter to see only specific traffic
  - Can be fully automated
- Where to get it

https://www.tcpdump.org/

Tcpdump example

- Debian/Ubuntu
  - Place the following in /etc/rc.local
- Red Hat/CentOS, Fedora
  - Place the following in /etc/rc.d/rc.local
- Grabs all traffic and rotates every 60 min
  - Date/time stamped and compressed

```
#Place _above_ any "exit" line
mkdir -p /opt/pcaps
screen -S capture -t capture -d -m bash -c "tcpdump -ieth0 -G
3600 -w '/opt/pcaps/`hostname -s`.%Y%m%d%H%M%S.pcap' -z bzip2"
```

tshark

- What's it good for?
  - Extracting interesting fields from packet captures
  - Multiple passes to focus on different attributes
  - Combine with text manipulation tools
  - Can be automated
- When to use it
  - Both major and minor attributes
- Where to get it

https://www.wireshark.org/

Tshark example - DNS queries

```
$ tshark -r thunt-lab.pcapng -T fields -e dns.qry.name
udp.port==53 | head -10

6dde0175375169c68f.dnsc.r-1x.com
6dde0175375169c68f.dnsc.r-1x.com
0b320175375169c68f.dnsc.r-1x.com
0b320175375169c68f.dnsc.r-1x.com
344b0175375169c68f.dnsc.r-1x.com
344b0175375169c68f.dnsc.r-1x.com
0f370175375169c68f.dnsc.r-1x.com
0f370175375169c68f.dnsc.r-1x.com
251e0175375169c68f.dnsc.r-1x.com
251e0175375169c68f.dnsc.r-1x.com
```

Tshark example - user agents

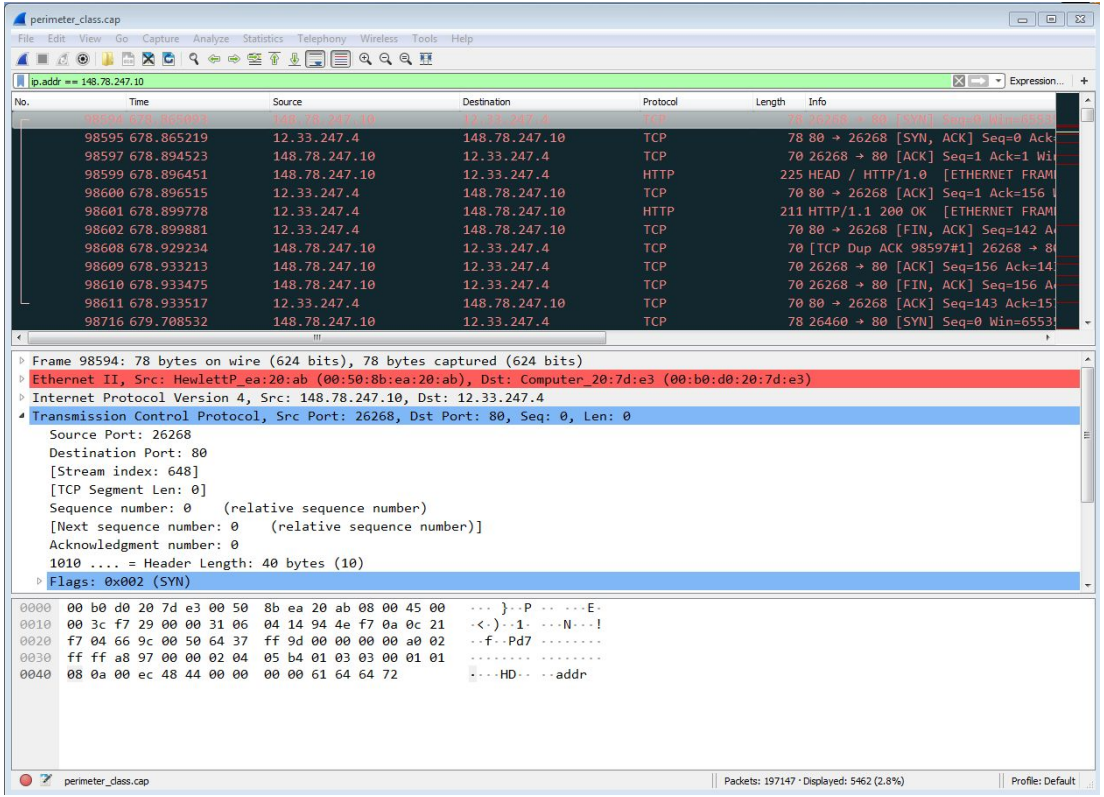
```
$ tshark -r sample.pcap -T fields -e http.user_agent tcp.
dstport==80 | sort | uniq -c | sort -n | head -10
      2 Microsoft Office/16.0
      2 Valve/Steam HTTP Client 1.0 (client;windows;10;1551832902)
      3 Valve/Steam HTTP Client 1.0
     11 Microsoft BITS/7.5
     11 Windows-Update-Agent
     12 Microsoft-CryptoAPI/6.1
    104 PCU
```

Wireshark

- What's it good for?
  - Packet analysis with guardrails
  - Stream level summaries
- When to use it
  - As part of a manual analysis
  - When steps cannot be automated
- Where to get it

https://www.wireshark.org/

Useful when I have a target



Bro/Zeek

- What's it good for?
  - Near real time analysis
  - More storage friendly than pcaps
- When to use it
  - When you need to scale
  - When you know what attributes to review
- Where to get it

https://www.zeek.org/
sudo apt -y install zeek zeek-aux

Gets you zeek-cut tools

Bro/Zeek example

```
$ cat conn.log | zeek-cut  id.orig_h id.resp_h id.resp_p
proto service orig_ip_bytes resp_ip_bytes

183.131.82.99    104.248.191.205  22     tcp  ssh  1923   0
112.85.42.229    104.248.191.205  22     tcp  -    344    0
104.248.191.205  67.207.67.3      53     udp  dns  42     126
81.22.45.150     104.248.191.205  7180   tcp  -    80     40
110.49.40.4      104.248.191.205  445    tcp  -    52     40
81.22.45.150     104.248.191.205  7404   tcp  -    80     40
```

Bro/Zeek example - cert check

```
$ cat ssl* | zeek-cut id.orig_h id.resp_h id.resp_p
validation_status | grep 'self signed' | sort | uniq
122.228.10.51   192.168.88.2    9943    self signed certificate in
certificate chain
24.111.1.134    192.168.88.2    9943    self signed certificate in
certificate chain
71.6.167.142    192.168.88.2    9943    self signed certificate in
certificate chain
```

ngrep

- Pattern match on passing packets
- Like "grep" for network traffic
- Useful for quick checks
  - NIDS with signature better choice for long term
- Useful switches
  - "-q" = Don't print "#" for non-matches
  - "-I" = Read a pcap file

https://github.com/jpr5/ngrep
sudo apt install ngrep

Ngrep example

```
cbrenton@cbrenton-lab-testing:~/pcaps$ ngrep -q -I odd.pcap Admin | head -15
input: odd.pcap
match: Admin

T 148.78.247.10:26922 -> 12.33.247.4:80 [AP]
  GET /cfide/Administrator/startstop.html HTTP/1.0..Host: 12.33.247.4..User-Agent: Mozilla/5.0 [en] (Win
  95; U)..Referer: http://12.33.247.4/..X-Forwarded-For: 148.64.147.168..Cache-Control: max-stale=0..Pra
  gma: no-cache......Cv

T 12.33.247.4:80 -> 148.78.247.10:26922 [AP]
  HTTP/1.1 404 Not Found..Date: Tue, 25 Jun 2002 00:34:58 GMT..Server: Apache..Connection: close..Conten
  t-Type: text/html; charset=iso-8859-1....<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">.<HTML><HEA
  D>.<TITLE>404 Not Found</TITLE>.</HEAD><BODY>.<H1>Not Found</H1>.The requested URL /cfide/Administrato
  r/startstop.html was not found on this server.<P>.</BODY></HTML>.....

T 12.33.247.4:80 -> 148.78.247.10:26922 [AFP]
cbrenton@cbrenton-lab-testing:~/pcaps$ _
```

Datamash

- What's it good for?
  - Similar to the R-base tools, but more extensive
  - Performing simple calculation on data
- When to use it
  - Performing calculations on multiple lines
  - Statistical analysis
- Where to get it

https://www.gnu.org/software/datamash/
sudo apt install datamash

Datamash example

```
cbrenton@cbrenton-lab-testing:~/lab3$ cat conn.log | bro-cut id.orig_h id.resp_h duration | s
ort -k 3 -rn | head
192.168.1.105   143.166.11.10   328.754946
192.168.1.104   63.245.221.11   41.884228
192.168.1.104   63.245.221.11   31.428539
192.168.1.105   143.166.11.10   27.606923
192.168.1.102   192.168.1.1     4.190865
192.168.1.103   192.168.1.1     2.652339
192.168.1.105   192.168.1.1     1.596499
192.168.1.103   192.168.1.1     1.234217
192.168.1.102   192.168.1.1     1.025109
192.168.1.103   192.168.1.1     0.770762
cbrenton@cbrenton-lab-testing:~/lab3$ cat conn.log | bro-cut id.orig_h id.resp_h duration | s
ort | datamash -g 1,2 sum 3 | sort -k 3 -rn | head
192.168.1.105   143.166.11.10   356.361869
192.168.1.104   63.245.221.11   73.312767
192.168.1.102   192.168.1.1     5.464553
192.168.1.103   192.168.1.1     4.956918
192.168.1.105   192.168.1.1     1.99374
192.168.1.104   77.67.44.206    1.706412
192.168.1.104   198.189.255.75  1.049496
192.168.1.104   192.168.1.1     0.972653
cbrenton@cbrenton-lab-testing:~/lab3$
```

Duplicate entries

RITA

- What's it good for?
  - Beacon & long conn at scale
  - Some secondary attributes
- When to use it
  - Can better organize Bro/Zeek data
  - Good when you are comfortable scripting
  - Will scale but can be time consuming
- Where to get it

https://github.com/activecm/rita

RITA example - beacons

```
cbrenton@cbrenton-lab-testing:~$ rita show-beacons thunt-lab | head
Score,Source IP,Destination IP,Connections,Avg. Bytes,Intvl Range,Size Range,Top Intv
l,Top Size,Top Intvl Count,Top Size Count,Intvl Skew,Size Skew,Intvl Dispersion,Size
Dispersion
1,10.55.100.111,165.227.216.194,20054,92,29,52,1,52,7774,20053,0,0,0,0
1,192.168.88.2,165.227.88.15,108858,199,860,230,1,89,53341,108319,0,0,0,0
0.838,10.55.200.10,205.251.194.64,210,308,29398,4,300,70,109,205,0,0,0,0
0.835,10.55.200.11,205.251.197.77,69,308,1197,4,300,70,38,68,0,0,0,0
0.834,10.55.100.111,34.239.169.214,34,1259,5,14388,1,156,15,30,0,0,0,0
0.834,192.168.88.2,13.107.5.2,27,198,2,33,12601,73,4,15,0,0,0,0
0.833,10.55.100.107,23.52.162.184,24,2397,43356,52,1800,467,18,18,0,0,0,0
0.833,10.55.100.107,23.52.161.212,24,5404,43235,52,1800,505,19,21,0,0,0,0
0.833,10.55.100.111,23.52.161.212,27,5379,37752,92,1800,505,17,20,0,0,0,0
cbrenton@cbrenton-lab-testing:~$ _
```

Scale is 0 - 1 with 1.0 being a perfect beacon score

RITA example - C2 over DNS

```
thunt@thunt-one-day:~$ rita show-exploded-dns test | head -10
Domain,Unique Subdomains,Times Looked Up
cymru.com,227,502
hash.cymru.com,224,485
malware.hash.cymru.com,222,341
akadns.net,134,19282
edgekey.net,116,6342
akamaiedge.net,116,19680
microsoft.com,91,3116
amazonaws.com,89,6369
com.edgekey.net,83,5401
thunt@thunt-one-day:~$ _
```

Passer

```
TC,172.1.199.23,TCP_43,open,
TC,172.16.199.23,TCP_55443,open,
UC,172.16.199.23,UDP_626,open,serialnumberd/clientscanner likely nmap
scan Warnings:scan
UC,172.16.199.23,UDP_1194,open,openvpn/client Warnings:tunnel
UC,172.16.199.23,UDP_3386,open,udp3386/client
UC,172.16.199.23,UDP_5632,open,pcanywherestat/clientscanner
Warnings:scan
UC,172.16.199.23,UDP_64738,open,shodan_host/clientscanner abcdefgh
Unlisted host Warnings:scan
DN,2001:db8:1001:0000:0000:0000:0000:0015,AAAA,ns3.markmonitor.com.,
DN,fe80:0000:0000:0000:189f:545b:7d4c:eeb8,PTR,Apple
TV._device-info._tcp.local.,model=J105aA
```

C2 Labs

What We Will Cover

- This section is mostly hands on labs
- Implement what you have learned
- Lab format:
  - Given a problem
    - Use earlier content to help solve
  - Given hints
    - If you don't know where to start, try the hints
  - Given the exact commands
  - Solution
    - Complete walk through of the solution

Reminder

- All lab files are on the VM
  - No network access needed
- Login info
  - Name = thunt
  - Password = aybab2u
- Labs are in /home/thunt/lab*

Find long connections

- Files located in /home/thunt/lab1
- Provided with pcap and Zeek log files
- Identify
  - Top 10 longest connections between private and legal IP addresses (internal to external)
  - Top 10 cumulative communication time between private and legal IP addresses (internal to external)

Find long conns - Hints

- Long connections is a relative term. You need to know the length of time being audited.
- Pcaps don't store connection duration
- Zeek stores duration in conn.log
- Zeek-cut extracts fields from Zeek logs
- Datamash is useful for adding values

Useful commands to try

```
capinfos -aeu <pcap file>

cat conn.log | zeek-cut id.orig_h
id.resp_h duration | sort -k 3 -rn | head

cat conn.log | zeek-cut id.orig_h
id.resp_h duration | sort | grep -v -e
'^$' | grep -v '-' | datamash -g 1,2 sum 3
| sort -k 3 -rn | head
```

Long conns - Answers

- Need to ID how long the pcap captured
- Use Zeek conn.log to easily get duration
- Need to extract:
  - Source IP (id.orig_h)
  - Destination IP (id.resp_h)
  - Duration of each connection (duration)
- Need to be able to:
  - Add up connection time between IP's
  - Present longest results first

less -S conn.log

```
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   conn
#open   2021-02-17-17-25-17
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
#types  time    string  addr    port    addr    port    enum    string  interval
1591289958.726326       CbKb5j3ZEYav4R0YVb      192.168.99.51   52833   104.248.23
1591289968.892657       Cpo9lX3puToMh46G9l      192.168.99.51   52831   23.223.200
1591289988.618982       CcNiZE1KwsmTDb29tk      192.168.99.51   52834   104.248.23
1591289986.217731       CS1Mfu3sq8jpYoWSJ9      192.168.99.52   5353    224.0.0.25
1591289986.218581       CUtQmj4vjvZfA0pGm8      fe80::d048:42e0:8448:187c       53
1591289986.219093       CILjxK28TlZc2TC315      fe80::d048:42e0:8448:187c       60
```

"Duration" = duration

Identify time window being audited

```
thunt@thunt:~/lab1$ capinfos -aeu trace1.pcap
File name:           trace1.pcap
Capture duration:    86398.498096 seconds
First packet time:   2020-06-04 16:59:02.292525
Last packet time:    2020-06-05 16:59:00.790621
thunt@thunt:~/lab1$ _
```

24 hours = 86,400 seconds

Plan B for files too large for capinfos:

```
tcpdump -tttt -n -r <filename> | awk 'NR==1; END{print}'
```

Longest unique connections

```
thunt@thunt:~/lab1$ cat conn.log | zeek-cut id.orig_h id.resp_h duration | sort -k 3 -rn | head
192.168.99.51   167.71.97.235   86389.659357
192.168.99.51   104.248.234.238 243.768999
192.168.99.51   104.118.9.117   166.139547
192.168.99.51   72.21.91.29     134.888177
192.168.99.51   52.184.216.246  129.075227
192.168.99.51   52.167.249.196  128.957107
192.168.99.51   52.184.216.246  128.481757
192.168.99.51   13.107.5.88     128.346889
192.168.99.51   52.179.219.14   128.116421
192.168.99.51   13.107.5.88     128.042647
thunt@thunt:~/lab1$
```

Duration is just short of the full 86,398 second capture time

Longest talk time

```
thunt@thunt:~/lab1$ cat conn.log | zeek-cut id.orig_h id.resp_h duration | sort |
grep -v '-' | datamash -g 1,2 sum 3 | sort -k 3 -rn | head
192.168.99.51   167.71.97.235   86389.659357
192.168.99.51   52.179.219.14   4067.394413
192.168.99.51   52.184.217.56   2936.172839
192.168.99.51   52.184.216.246  2825.858
192.168.99.52   239.255.255.250 2507.626732
fe80::d048:42e0:8448:187c       ff02::c 2434.977049
192.168.99.51   239.255.255.250 2374.546469
fe80::2126:bcd7:16f4:8cdb       ff02::c 2368.234679
192.168.99.51   13.107.5.88     1317.047871
192.168.99.51   52.167.249.196  868.46966
thunt@thunt:~/lab1$ _
```

Note the first entry is still the same, but all others are new.
IPv6 addresses have shifted info to the right.

Investigate the longest talkers

- Let's investigate the external IP of the two longest session
  - 167.71.97.235
  - 52.179.219.14
- We'll use two common research methods
  - "host" command
  - AbuseIPDB
    - https://www.abuseipdb.com/
  - ThreatCrowd
    - https://www.threatcrowd.org/

Investigate - hints

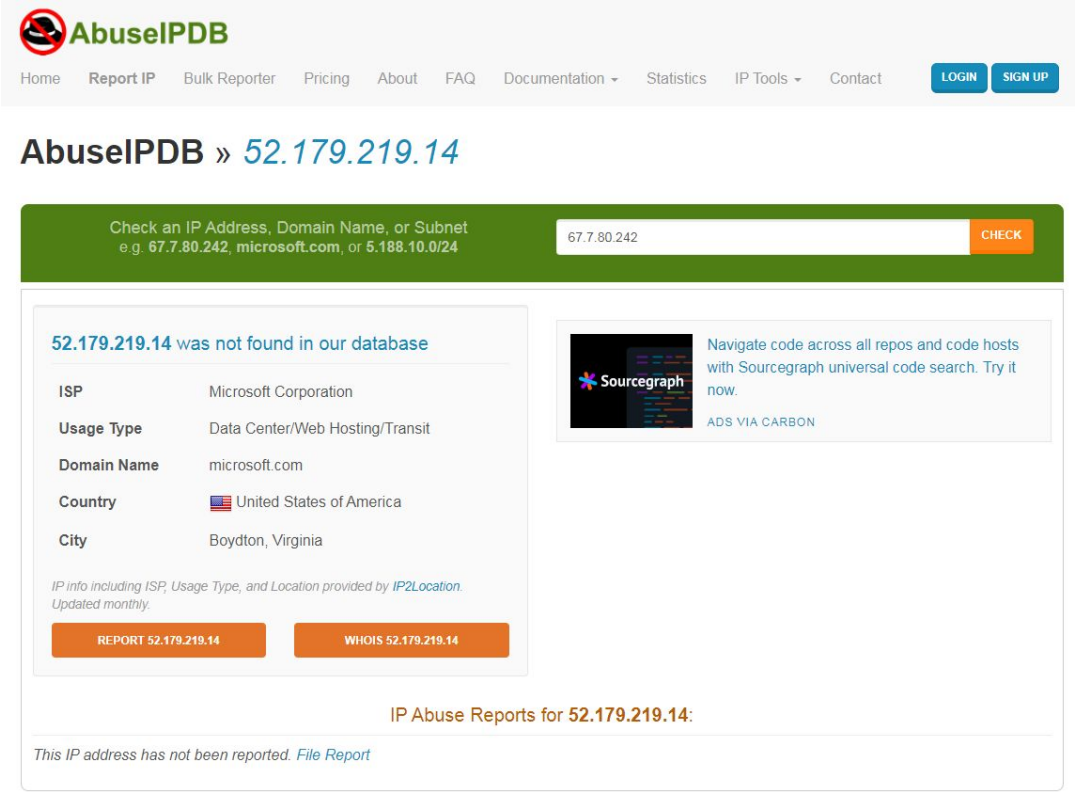
- You were given the two IP addresses to research
- The "host" command is run from the command line
- Use a browser to connect to the two research Websites and enter each IP

One out of two is not bad

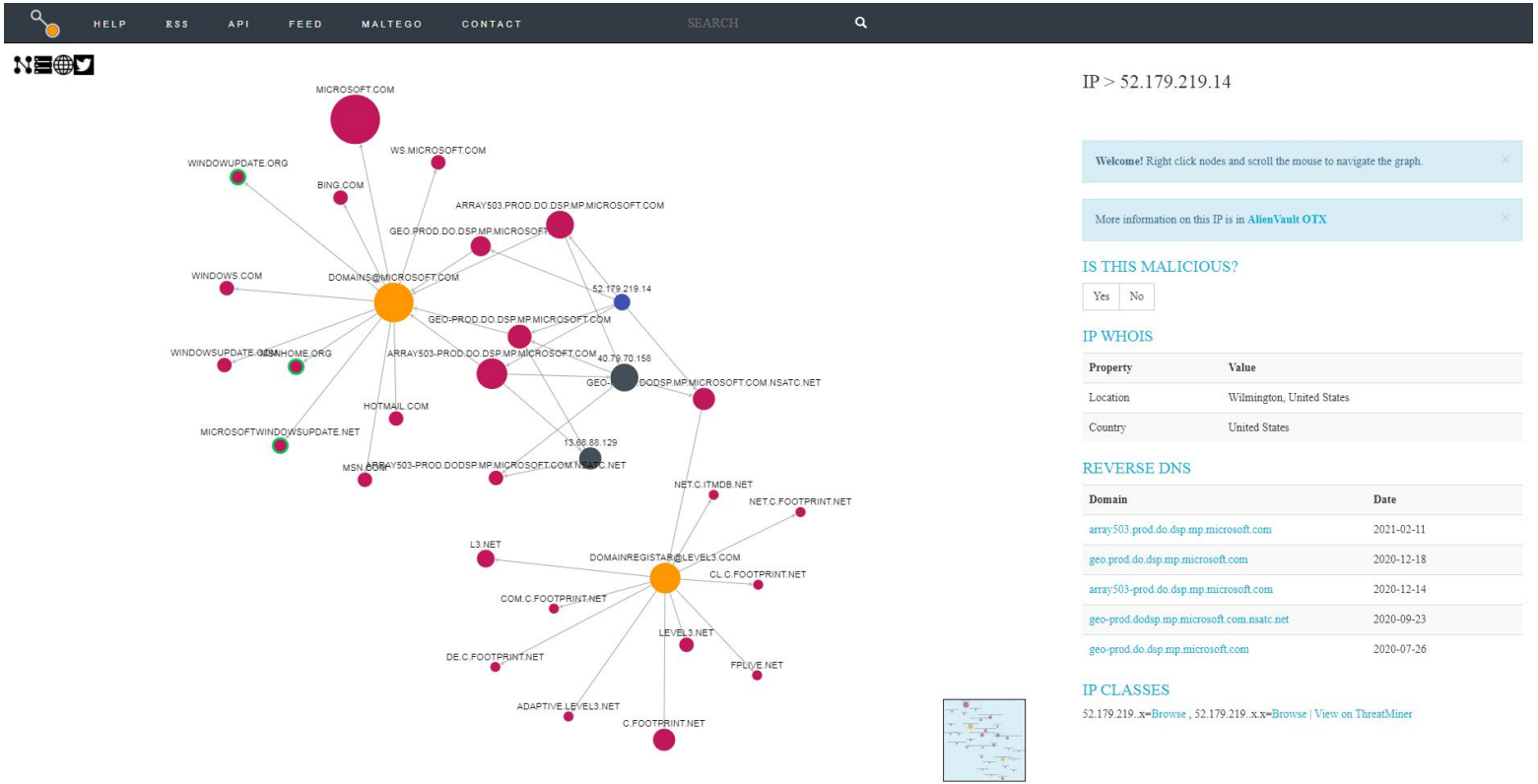
```
thunt@thunt:~/lab1$ host 167.71.97.235
235.97.71.167.in-addr.arpa domain name pointer demo1.aihhosted.com.
thunt@thunt:~/lab1$ host 52.179.219.14
Host 14.219.179.52.in-addr.arpa. not found: 3(NXDOMAIN)
thunt@thunt:~/lab1$ _
```

Is there a business need for demo1.aihhosted.com?
If so, that one connection can be considered safe.

AbuseIPDB data on 2nd IP



ThreatCrowd data on 2nd IP



Running down 52.179.219.24

- Looks like Windows calling home to Microsoft
- Can we verify this?
- Let's find the entry in conn.log and see if we can find corroborating data points

Looks like it's SSL/TLS traffic

```
thunt@thunt-labs:~/lab1$ grep 52.179.219.14 conn.log | head -5
1591290650.463848       Ce8vuV9pdZN1TTE21       192.168.99.51   52863   52.179.219.14
443     tcp     ssl     65.389372       1270    3035    SF      -       -       0
ShADadFf        14      1842    12      3527    -
1591292050.459124       CbnymM8GhENDKN6ol       192.168.99.51   52938   52.179.219.14
443     tcp     ssl     95.406423       1270    3036    SF      -       -       0
ShADdaFf        14      1842    12      3528    -
1591293617.574816       CdOPg52V3t5AGGkjyf      192.168.99.51   52999   52.179.219.14
443     tcp     ssl     68.280122       1270    3036    SF      -       -       0
ShADdaFf        14      1842    11      3488    -
1591295064.955993       CKXOFb4bJ1gZgX3sW2      192.168.99.51   53150   52.179.219.14
443     tcp     ssl     128.116421      1269    3036    RSTR    -       -       0
ShADdar 12      1761    11      3488    -
1591295092.098734       CjkEjk4m4GL1LSMJMd      192.168.99.51   53153   52.179.219.14
443     tcp     ssl     113.248030      1246    3036    SF      -       -       0
ShADdaFf        14      1818    11      3488    -
thunt@thunt-labs:~/lab1$ _
```

Entry in ssl.log

```
thunt@thunt-labs:~/lab1$ grep 52.179.219.14 ssl.log | head -2
1591290650.502177       Ce8vuV9pdZN1TTE21       192.168.99.51   52863   52.179.219.14
443     TLSv12  TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 x25519  array503.prod.do.dsp.
mp.microsoft.com        F       -       h2      T       Fd3zBI3qZR5omLoAi7,FWU71E32do
A3zMCOH (empty) CN=*.prod.do.dsp.mp.microsoft.com,OU=DSP,O=Microsoft,L=Redmond,ST=WA,
C=US    CN=Microsoft ECC Content Distribution Secure Server CA 2.1,O=Microsoft Corpor
ation,L=Redmond,ST=Washington,C=US       -       -
1591292050.498723       CbnymM8GhENDKN6ol       192.168.99.51   52938   52.179.219.14
443     TLSv12  TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 x25519  array503.prod.do.dsp.
mp.microsoft.com        F       -       h2      T       Fle59121NCCmoWeZnf,FMXEBQ1sio
EzUSPeza        (empty) CN=*.prod.do.dsp.mp.microsoft.com,OU=DSP,O=Microsoft,L=Redmon
d,ST=WA,C=US    CN=Microsoft ECC Content Distribution Secure Server CA 2.1,O=Microsof
t Corporation,L=Redmond,ST=Washington,C=US       -       -
thunt@thunt-labs:~/lab1$
```

x509.log info

```
thunt@thunt-labs:~/lab1$ grep Fd3zBI3qZR5omLoAi7 x509.log
1591290650.543514       Fd3zBI3qZR5omLoAi7      3       33000000188C1ABDC391569B5F000
000000018       CN=*.prod.do.dsp.mp.microsoft.com,OU=DSP,O=Microsoft,L=Redmond,ST=WA,
C=US    CN=Microsoft ECC Content Distribution Secure Server CA 2.1,O=Microsoft Corpor
ation,L=Redmond,ST=Washington,C=US       1574361906.000000       1613933106.000000
id-ecPublicKey  ecdsa-with-SHA384       ecdsa   256     -       prime256v1      *.pro
d.do.dsp.mp.microsoft.com       -       -       -       F       -
thunt@thunt-labs:~/lab1$
```

Answers

- Longest connection appears to be business partner related
- Second longest is is used in keeping Windows 10 updated
- Neither appear to be malware related
- Is there a business need for this?
  - If no, hunt down app and kill it
  - If yes, whitelist to remove from future hunts

Find beacons by session size

- Use the same data files as last lab
- Identify which internal IP's are connecting to individual external IP's most frequently
- Focus on IP pairs that create thousands of connections per days
  - Beacons can have smaller quantities, but we need to start somewhere
- Is there consistency in session size?
  - Possible beacon?

Find beacons - hints

- You need to be able to clearly identify:
  - Number of unique connections over 24 hours
    - Not the number of packets
  - The amount of payload data transferred
- Pick targets - Who has most connections?
- Zeek displays both bytes sent and received
  - Focus on bytes sent
  - orig_bytes

Useful commands to try

```
cat conn.log | zeek-cut id.orig_h id.resp_h | sort
| uniq -c | sort -rn | head

cat conn.log | zeek-cut id.orig_h id.resp_h
orig_bytes | grep 192.168.99.51 | grep
104.248.234.238 | sort | uniq -c | sort -rn | head
```

Answers - most connections

```
thunt@thunt:~/lab1$ cat conn.log | zeek-cut id.orig_h id.resp_h | sort | uniq -c |
 sort -rn | head
   3011 192.168.99.51   104.248.234.238
    336 fe80::b8d7:3773:ab6e:7fc9       ff02::1:3
    336 192.168.99.54   224.0.0.252
    332 fe80::194f:796e:70e6:a5be       ff02::1:3
    332 192.168.99.55   224.0.0.252
    330 fe80::fd16:6e8:118e:81cd        ff02::1:3
    330 192.168.99.53   224.0.0.252
    319 fe80::d048:42e0:8448:187c       ff02::1:3
    319 192.168.99.52   224.0.0.252
    297 192.168.99.51   208.67.222.222
thunt@thunt:~/lab1$
```

The first looks potentially suspicious (no time analysis)
The rest are just local multicast traffic

Session size analysis

```
thunt@thunt:~/lab1$ cat conn.log | zeek-cut id.orig_h id.resp_h orig_bytes | grep
192.168.99.51 | grep 104.248.234.238 | sort | uniq -c | sort -rn | head
   3011 192.168.99.51   104.248.234.238 477
thunt@thunt:~/lab1$
```

Every session resulted in 477 bytes sent to external host
This could indicate a beacon that was not activated over the 24 hours

Payload analysis with ngrep

- We found a suspicious IP pair
  - 192.168.99.51 to 104.248.234.238
- Let's analyze the payloads in these sessions
- Multiple tools can help here
  - But ngrep easily focuses on payload
- Use "host" parameter to focus in on the above IPs

Payload analysis - hints

- Ngrep is normally used to search for patterns within the payload of all packets
- You can use BP filters to:
  - Focus on specific IP addresses
  - Focus on specific ports
  - "host" focuses on specific IP addresses
- Helpful switches
  - "-q" = Don't print "#" for packets that don't match
  - "-I" (capital letter i) = Read from pcap file

Useful commands to try

```
ngrep -q -I trace1.pcap host 192.168.99.51 and host
104.248.234.238 | less
```

Things that make you go "humm"

```
thunt@thunt:~/lab1$ ngrep -q -I trace1.pcap host 192.168.99.51 and host 104.248.23
4.238 | head -20
input: trace1.pcap
filter: ( host 192.168.99.51 and host 104.248.234.238 ) and ((ip || ip6) || (vlan
&& (ip || ip6)))

T 192.168.99.51:52833 -> 104.248.234.238:80 [AP] #4
  GET /rmvk30g/eghmbblnphlaefbmmnoenohhoncmcepapefjjekpleokhjfjmnmijghedkienpli
  dbbcmgdjldbegpeemiboacnfcpnbnnhlmjbpcejfpecdioiddklfegefcjbcnagjclnoijpajlpkk
  egakmpdddojnlphegeehaacmofggdfkagpbighfkndllaamndepdanhnogedkaodhgakiigohemin
  oolnaobdiiokpebghapnghbebkepiffoo(jden;1;4;1 HTTP/1.1..Accept: text/html, ima
  ge/gif, image/jpeg, *; q=.2, */*; q=.2..Connection: keep-alive..User-Agent: M
  ozilla/4.0 (Windows 7 6.1) Java/1.7.0_11..Host: 104.248.234.238..Cache-Contro
  l: no-cache....

T 104.248.234.238:80 -> 192.168.99.51:52833 [A] #5
  ......

T 104.248.234.238:80 -> 192.168.99.51:52833 [AP] #6
  HTTP/1.1 200 OK..Date: Thu, 4 Jun 2020 16:59:22 GMT..Server: Apache/2.2.15 (C
  entOS)..X-Powered-By: PHP/5.3.27..Content-Type: application/octet-stream..Con
  nection: close..Content-Length: 0....
```

What data are we sending?

- Is this the only URI we send to this host?
- We could eyeball it, but…
- Zeek stores this type of data
  - It's in the http.log file
- Let's use this log to identify all of the URI's requested from this external host

URI request - hints

- Zeek-cut is your friend
- We should extract
  - Source IP
  - Destination IP
  - The "uri" string
- Grep can focus on the traffic we care about
- Remember the threat hunter's mantra
  - sort | uniq | sort

Useful commands to try

```
cat http.log | zeek-cut id.orig_h id.resp_h uri |
grep 104.248.234.238 | sort | uniq -c | sort -rn
```

Single minded request

```
thunt@thunt:~/lab1$ cat http.log | zeek-cut id.orig_h id.resp_h uri | grep 104.248
.234.238 | sort | uniq -c | sort -rn
   3011 192.168.99.51   104.248.234.238 /rmvk30g/eghmbblnphlaefbmmnoenohhoncmcepap
efjjekpleokhjfjmnmijghedkienplidbbcmgdjldbegpeemiboacnfcpnbnnhlmjbpcejfpecdioiddkl
fegefcjbcnagjclnoijpajlpkkegakmpdddojnlphegeehaacmofggdfkagpbighfkndllaamndepdanhn
ogedkaodhgakiigoheminoolnaobdiiokpebghapnghbebkepiffooljden;1;4;1
thunt@thunt:~/lab1$
```

Answers

- 3,011 connections to external host
- Always sending the same odd "GET" request
- HTTP header data looks forged
- This really looks like a C2 channel
- Google search for "rmvk30g"
  - Looks like Fiesta EK malware

https://www.malware-traffic-analysis.net/2014/04/05/index.html

Look for C2 over DNS

- Move to the "lab2" directory
- Check to see if C2 over DNS is in play
- Consider any domain with more than 1,000 FQDNs in it suspect
  - Not interested in total quantity of queries
  - Interest in quantities of unique FQDNs

C2 over DNS - hints

- Zeek has a log file just for DNS traffic
- "query" field shows what was looked up
- Need a way to count hosts within a domain
- Some helpful text manipulation tools
  - sort = Pull together matching lines
  - uniq = Remove repeat entries
  - rev = Reverse the characters on a line
  - cut = Remove a section of characters on a line

Useful commands to try

```
cat dns.log | zeek-cut query | sort | uniq |
rev | cut -d . -f 1-2 | rev | sort | uniq -c |
sort -rn | head
```

C2 over DNS - Zeek

```
thunt@thunt:~/lab2$ cat dns.log | zeek-cut query | sort | uniq | rev | cut -d . -f
 1-2 | rev | sort | uniq -c | sort -rn | head
   2074 honestimnotevil.com
      1 ne.jp
      1 in-addr.arpa
      1 -
thunt@thunt:~/lab2$
```

That first entry looks pretty odd

Answers

- We looked up 2,074 FQDNs within honestimnoteveil.com
- This extremely high for a domain we do not recognize
- Could very well indicate C2 over DNS

Query types used by C2

- Many C2 over DNS tools use TXT record types to create channel
- This is why many orgs focus on this type
  - Leverage NIDs signatures
- Is that true for this C2 channel?
- Lab time!
  - Identify what record types were used

Hints - C2 over DNS record types

- Will need to extract "qtype_name" and "query" for each record
- We only care about "honestimnotevil" records
- Once these are extracted, we can "cut" out the query types and use our mantra to summarize

Useful commands to try

```
cat dns.log | zeek-cut qtype_name query | grep
honestimnotevil | cut -f 1 | sort | uniq -c |
sort -rn
```

A mix of query types

```
thunt@thunt:~/lab2$ cat dns.log | zeek-cut qtype_name query | grep honestimnotevil
 | cut -f 1 | sort | uniq -c | sort -rn
    707 MX
    692 TXT
    675 CNAME
thunt@thunt:~/lab2$
```

707 + 692 + 675 = 2,074 (same as number of FQDNs found in first lab)

Answers

- Three different query types were used
  - Fairly even spread of quantities
- May be done to reduce the number of records for a specific type
- While TXT was used, may not be needed
- We can't just look for TXT records and hope to always catch C2

Repeat the labs with RITA

- Let's see if RITA makes this easier
- Zeek logs already imported into RITA
- Dataset names match directory names
  - Lab1 & lab2
- Repeat analysis for each
- Note: RITA scores beacons, investigate 0.8 or higher scores
- Type "rita" to get a list of commands

Hints

- List current databases
  - rita list or rita show-databases
- Look for long connections
  - rita show-long-connections <database name>
- Look for beacons
  - rita show-beacons <database name>
- Look for C2 over DNS
  - rita show-exploded-dns <database name>

Useful commands to try

```
rita show-databases
rita show-long-connections lab1 | head

rita show-long-connections lab1 | cut -d ,
-f 1,2,4 | sort | datamash -H -t , -g 1,2
sum 3 | sort -t , -k 3 -rn | head

rita show-beacons lab1 | head

rita show-exploded-dns lab1 | head
```

Answers - Lab1

```
thunt@thunt:~/lab1$ rita show-long-connections lab1 | head -5
Source IP,Destination IP,Port:Protocol:Service,Duration
192.168.99.51,167.71.97.235,9200:tcp:-,86389.7
192.168.99.51,104.248.234.238,80:tcp:http,243.769
192.168.99.51,104.118.9.117,443:tcp:ssl,166.14
192.168.99.51,72.21.91.29,80:tcp:- 80:tcp:http,134.888
thunt@thunt:~/lab1$
thunt@thunt:~/lab1$ rita show-beacons lab1 | head -5
Score,Source IP,Destination IP,Connections,Avg. Bytes,Intvl Range,Size Range,Top I
ntvl,Top Size,Top Intvl Count,Top Size Count,Intvl Skew,Size Skew,Intvl Dispersion
,Size Dispersion
0.885,192.168.99.51,104.248.234.238,3011,1101,246,621,28,689,1019,2856,0,0,1,0
0.835,192.168.99.51,52.179.224.121,72,396,11,2,1200,183,69,69,0,0,0,0
0.586,192.168.99.51,208.67.220.220,60,245,7741,30,1,80,3,17,0.117434,-0.25,991,4
0.585,192.168.99.51,52.184.217.56,30,5258,2687,122,900,1810,1,15,-0.434783,0,305,1
thunt@thunt:~/lab1$
thunt@thunt:~/lab1$ rita show-exploded-dns lab1 | head -5
Domain,Unique Subdomains,Times Looked Up
microsoft.com,24,226
mp.microsoft.com,14,117
dsp.mp.microsoft.com,9,109
prod.do.dsp.mp.microsoft.com,8,107
thunt@thunt:~/lab1$
```

Answers - Lab2

```
thunt@thunt:~/lab1$ rita show-long-connections lab2 | head -5
No results were found for lab2
thunt@thunt:~/lab1$ rita show-beacons lab2 | head -5
No results were found for lab2
thunt@thunt:~/lab1$ rita show-exploded-dns lab2 | head -5
Domain,Unique Subdomains,Times Looked Up
honestimnotevil.com,2074,2074
8806d9a9068226a33b26e65071a0d496c751246292ec22b36bb5761c2762.5da0b7f90908be408ac43
eb80a.honestimnotevil.com,21,21
5da0b7f90908be408ac43eb80a.honestimnotevil.com,21,21
6a22df8dcd8e5032f95c2406362b70ddc5843efe182166d82ecf895312d7.60a5291b4324545e080e6
2a0ea.honestimnotevil.com,7,7
thunt@thunt:~/lab1$
```

Answers - Final

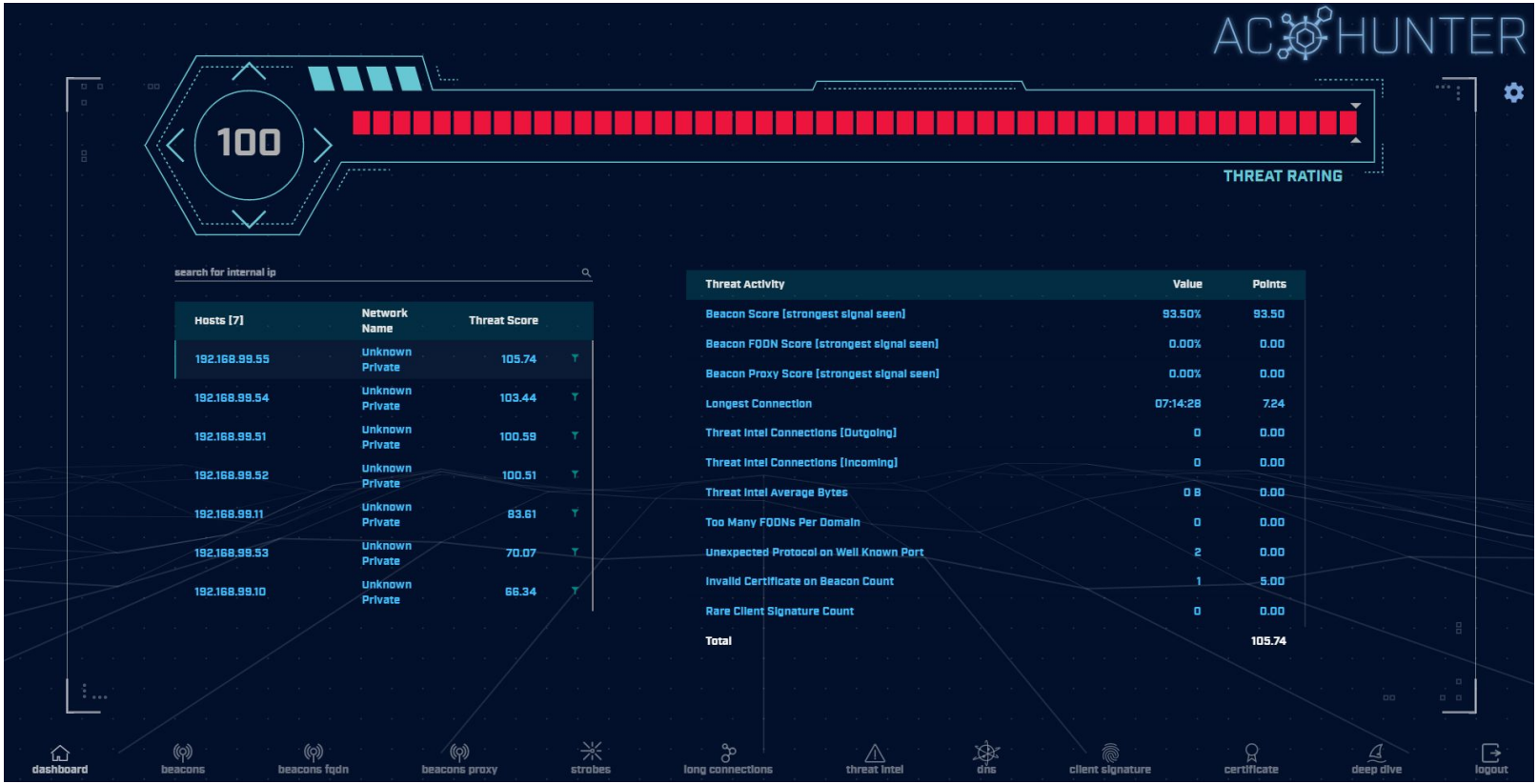
- RITA provides a consistent interface for identifying C2
- Screens pull in additional helpful info
- Even very slow beacons can be detected
- Investigation can be scripted
- Open source, so anyone can use it for free

Next steps

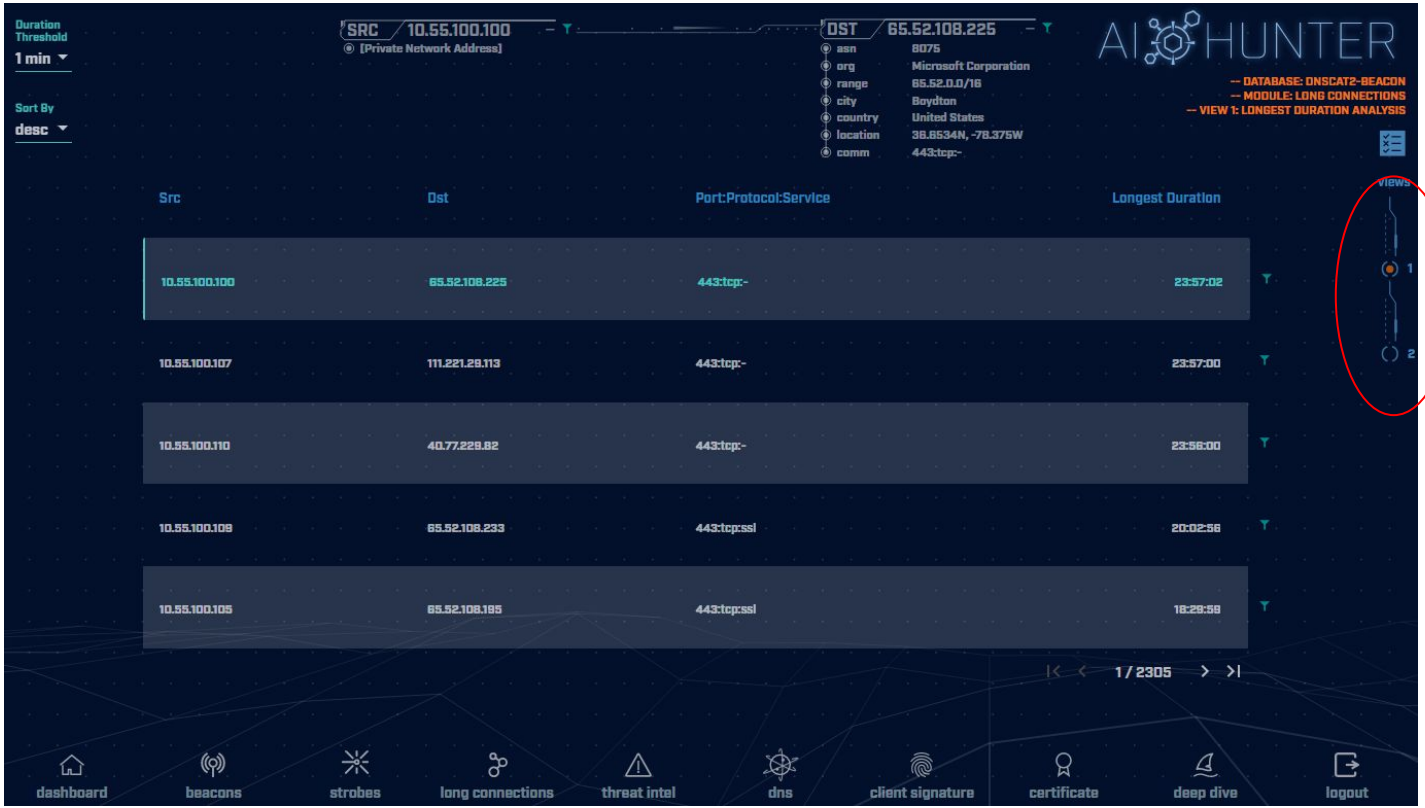
- Do we feel confident in flagging anything we have seen as requiring incident handling?
- Are there any connections that need more research?
  - What should this research be?
  - Do we need to involve any other teams?
  - If we need more data collection, for how long?

Quick demo

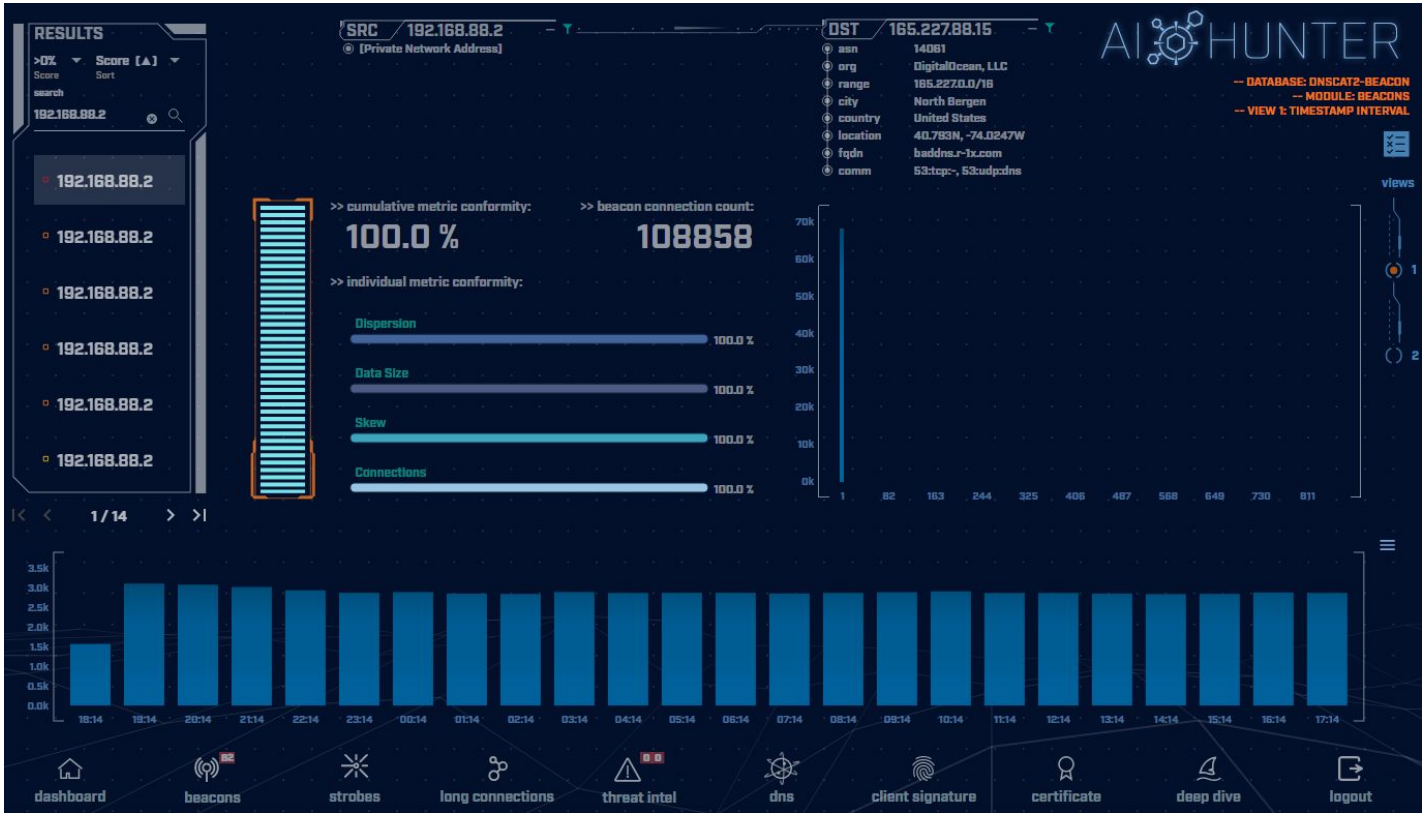
- Similar data, seen through AI-Hunter
- Inexpensive commercial solution
- Automates much of the hunting process



24 active hunts of 24-hours of data every single day
Top results scored, alerts sent to SIEM

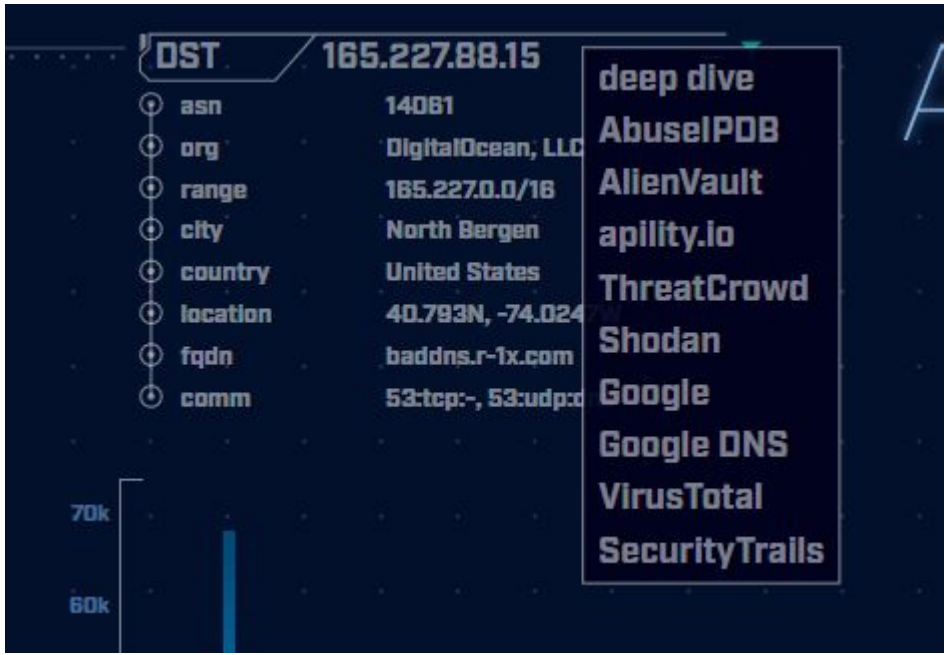


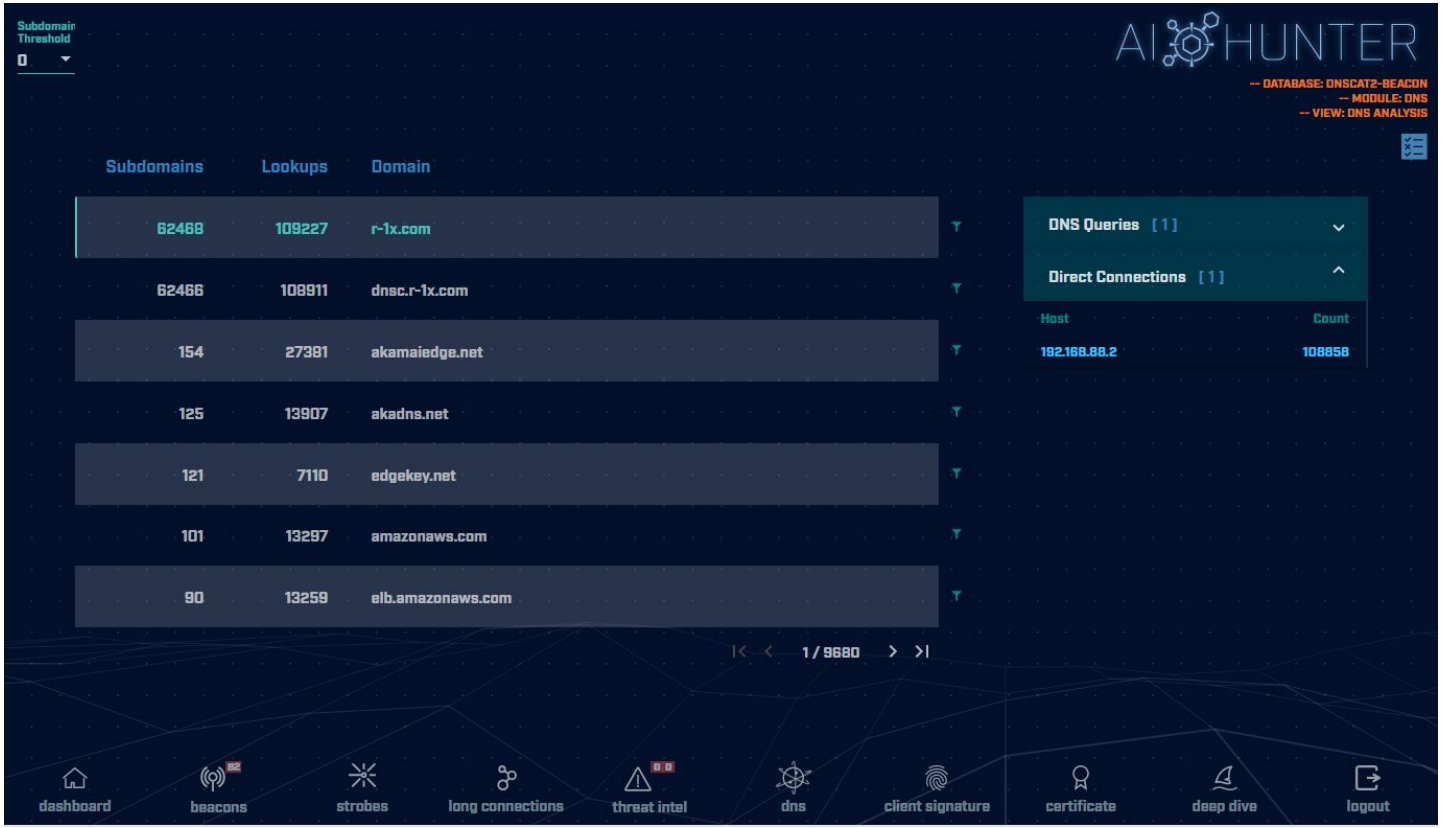
Long connections with lots of intel
View both individual and cumulative



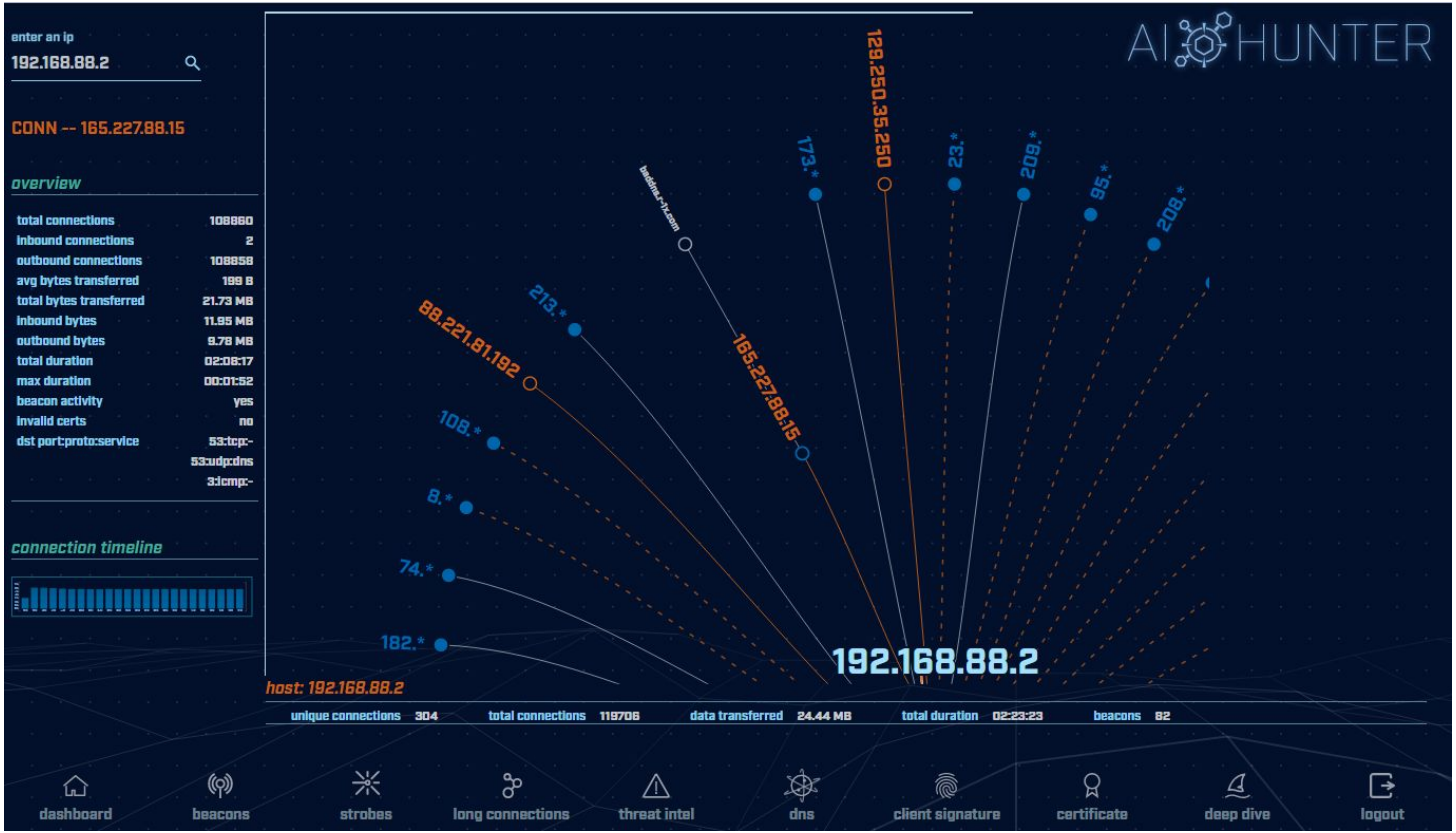
Clear beacon analysis
By both timing and session size

Resources to dig deeper





C2 over DNS analysis



Deep dive analysis

Take home lab

- This is a bonus lab to do on your own
  - Wait at least a week
  - Will help identify what training "stuck"
  - Answers are provided at the end
- Move to the "lab3" directory
- Check for long connections and beacons
- Investigate any suspect external IP's
- Do you see anything of concern?
- Hints and answers after "Wrap Up" slide

Continue your training

- Advanced Network Threat Hunting
  - 12/13 - 12/16 (4 hours per day)
  - $495

https://www.antisyphontraining.com/advanced-network-threat-hunting-w-chris-brenton/

- Getting Started with Packet Decoding
  - 10/19 - 10/22
  - 12/07 - 12/10
  - Pay what you want, $25+

https://www.antisyphontraining.com/event/getting-started-with-packet-decoding-w-chris-brenton/

Wrap Up

- Thanks for attending!
- Very special thank you to the folks behind the scenes
  - They give up their free time to help us all out
- Content feedback?
  - Please email: chris@activecountermeasures.com

Take home lab

- Move to the "lab3" directory
- Check for long connections and beacons
- Investigate any suspect external IP's
- Do you see anything of concern?

Hints for the take home lab

- Repeat what we did with "lab1"
  - Look for long connections
  - Look for cumulative communication time
  - Look for beacons
  - You can choose to jump right into RITA
- Us "up arrow" key to scroll through previous commands to find what you used earlier
- You've got this! :-)

Useful commands to try (1 of 2)

```
cat conn.log | zeek-cut id.orig_h id.resp_h
duration | sort -k 3 -rn | head

cat conn.log | zeek-cut id.orig_h id.resp_h
duration | sort | grep -v -e '^$' | grep -v '-'
| datamash -g 1,2 sum 3 | sort -k 3 -rn | head

cat conn.log | zeek-cut id.orig_h id.resp_h |
sort | uniq -c | sort -rn | head

host <IP address to investigate>
```

Useful commands to try (2/2)

```
rita show-databases
rita show-long-connections lab3 | head

rita show-long-connections lab3 | cut -d ,
-f 1,2,4 | sort | datamash -H -t , -g 1,2
sum 3 | sort -t , -k 3 -rn | head

rita show-beacons lab1 | head

rita show-exploded-dns lab1 | head
```

Answers - Long connections

```
thunt@thunt:~/lab3$ cat conn.log | zeek-cut id.orig_h id.resp_h duration | sort -k
 3 -rn | head
192.168.99.52   167.71.97.235   86387.734233
192.168.99.52   162.250.5.77    86347.153666
192.168.99.52   52.117.209.74   9868.617938
192.168.99.52   162.250.2.168   6735.118200
192.168.99.52   52.184.217.56   129.924272
192.168.99.52   52.184.212.181  129.754188
192.168.99.52   52.184.213.21   129.130822
192.168.99.52   52.184.212.181  129.123714
192.168.99.52   52.167.17.97    129.057349
192.168.99.52   52.167.17.97    128.896376
thunt@thunt:~/lab3$
```

Answers - Cumulative comm time

```
thunt@thunt:~/lab3$ cat conn.log | zeek-cut id.orig_h id.resp_h duration | sort |
grep -v -e '^$' | grep -v '-' | datamash -g 1,2 sum 3 | sort -k 3 -rn | head
192.168.99.52   167.71.97.235   86387.734233
192.168.99.52   162.250.5.77    86347.153666
192.168.99.52   52.117.209.74   9868.617938
192.168.99.52   52.184.217.56   7065.516309
192.168.99.52   52.184.213.21   7056.53546
192.168.99.52   162.250.2.168   6735.1182
192.168.99.52   52.184.212.181  6646.856637
192.168.99.52   239.255.255.250 2294.038962
fe80::d048:42e0:8448:187c       ff02::c 2281.05815
fe80::2126:bcd7:16f4:8cdb       ff02::c 2242.310744
thunt@thunt:~/lab3$
```

Same two top IPs

Answers - Beacons

```
thunt@thunt:~/lab3$ cat conn.log | zeek-cut id.orig_h id.resp_h | sort | uniq -c |
 sort -rn | head
    339 192.168.99.52   224.0.0.251
    319 192.168.99.52   208.67.222.222
    288 fe80::fd16:6e8:118e:81cd        ff02::fb
    288 fe80::fd16:6e8:118e:81cd        ff02::16
    288 fe80::d048:42e0:8448:187c       ff02::fb
    288 fe80::d048:42e0:8448:187c       ff02::16
    288 fe80::b8d7:3773:ab6e:7fc9       ff02::fb
    288 fe80::b8d7:3773:ab6e:7fc9       ff02::16
    288 fe80::5d7e:4fb3:8fbc:d59        ff02::fb
    288 fe80::5d7e:4fb3:8fbc:d59        ff02::16
thunt@thunt:~/lab3$
```

Nothing of note

Answers - RITA

```
thunt@thunt:~/lab1$ rita show-long-connections lab3 | head -5
Source IP,Destination IP,Port:Protocol:Service,Duration
192.168.99.52,167.71.97.235,9200:tcp:-,86387.7
192.168.99.52,162.250.5.77,5938:tcp:-,86347.2
192.168.99.52,52.117.209.74,5938:tcp:-,9868.62
192.168.99.52,162.250.2.168,5938:tcp:-,6735.12
thunt@thunt:~/lab1$ rita show-beacons lab3 | head -5
Score,Source IP,Destination IP,Connections,Avg. Bytes,Intvl Range,Size Range,Top I
ntvl,Top Size,Top Intvl Count,Top Size Count,Intvl Skew,Size Skew,Intvl Dispersion
,Size Dispersion
0.835,192.168.99.52,52.230.222.68,59,546,31350,2696,840,181,46,48,0,0,0,0
0.834,192.168.99.52,52.242.211.89,21,826,1651,2696,1680,181,14,11,0,0,0,0
0.833,192.168.99.52,104.71.255.238,24,5429,21721,40,1800,505,16,22,0,0,0,0
0.658,192.168.99.52,52.184.213.21,65,5392,2199,120,900,1883,28,33,0.99757,0,1,0
thunt@thunt:~/lab1$ rita show-exploded-dns lab3 | head -5
Domain,Unique Subdomains,Times Looked Up
microsoft.com,10,237
teamviewer.com,6,36
mp.microsoft.com,5,111
8.e.f.ip6.arpa,4,20
thunt@thunt:~/lab1$
```

Answers - Investigate IPs

```
thunt@thunt:~/lab3$ host 167.71.97.235
235.97.71.167.in-addr.arpa domain name pointer demo1.aihhosted.com.
thunt@thunt:~/lab3$ host 162.250.5.77
77.5.250.162.in-addr.arpa domain name pointer US-NJC-ANX-R010.teamviewer.com.
thunt@thunt:~/lab3$ _
```

Business need?

Answers - Final

- Two long connections found
- Unlikely (but not impossible) we have any beacons
- For the two long connections
  - First was discussed earlier (business partner)
  - The second is TeamViewer
- Is there a business need to run TeamViewer on this system?