



# Network Threat Hunter Training

Level 1

# Thanks to our sponsors!



# Continue Your Training

- ▷ Getting started with Packet Decoding
  - 8/20/21 - 16 hours over 4 days
  - Pay what you want

<https://wildwesthackinfest.com/antisyphon/getting-started-with-packet-decoding-w-chris-brenton/>

- ▷ Advanced Threat Hunting
  - 7/30/21 - 16 hours over 4 days
  - Live at WWHF in Sept

<https://wildwesthackinfest.com/antisyphon//advanced-network-threat-hunting-chris-brenton/>

# Before we get started

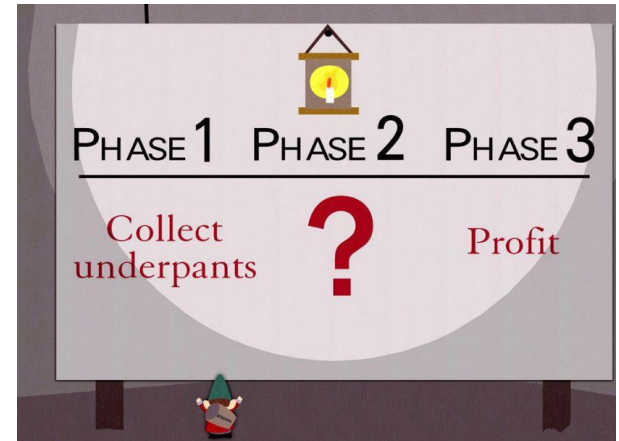
- ▷ You'll need the class VM to do the labs
- ▷ Login info:
  - Name: thunt
  - Pass: aybab2u
- ▷ Or run the install script
- ▷ Or deploy on DigitalOcean
- ▷ This should have been done before class :-)
- ▷ Slides are now available as well

# Logistics

- ▷ 10 minute break at top of each hour
- ▷ 30 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- ▷ The vision of a lone hacker in the basement is dangerously outdated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
- ▷ Focus is on avoiding signature matches

# 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
- ▷ Sometimes with pretty graphics
- ▷ Match known bad patterns
- ▷ Because adversaries have stopped innovating and we now know all of the possible bad patterns they can use
- ▷ Oh wait...
- ▷ This is the 1990's anti-virus model

# Lack of innovation

- ▷ Log review to detect attacks 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-f-or-an-average-of-800-days-before-discovery/>

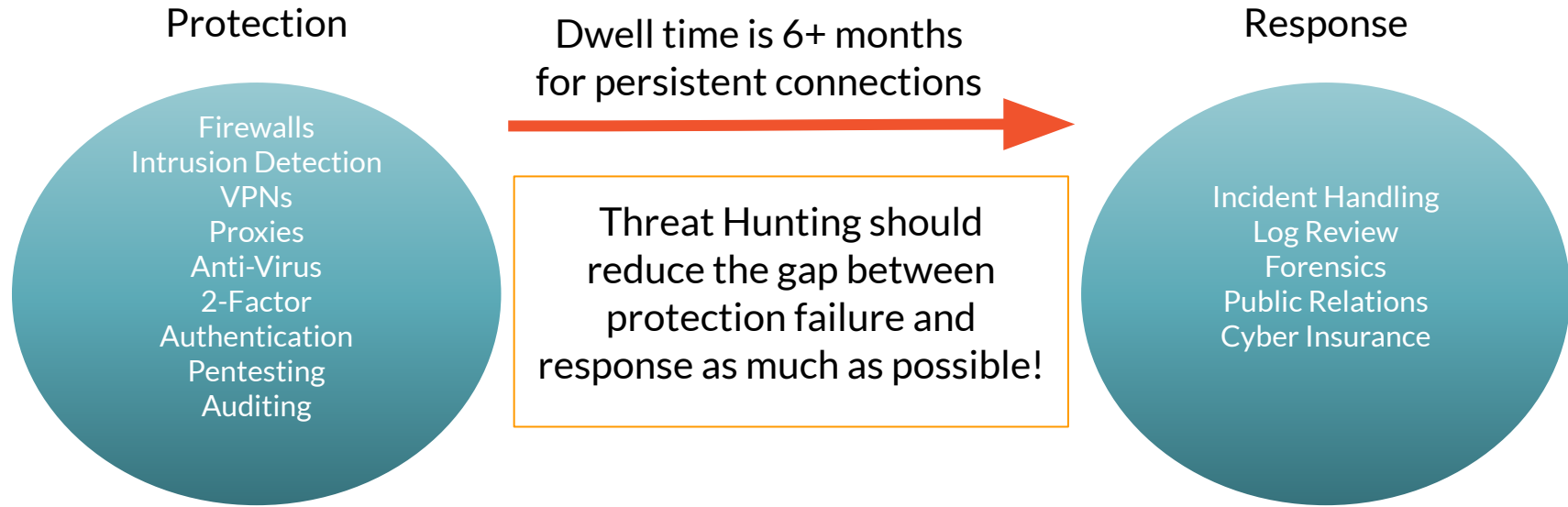
- ▷ We are getting worse at self detection

<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**

# The Purpose of Threat Hunting



# 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, even 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 ... <a href="#">show more</a>	Brute-Force
🇦🇺 Anonymous	25 Dec 2018	This ip showing as Microsoft Azure, location Chicago has been on all three of my blogs at Blogger an ... <a href="#">show more</a>	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
✓ <a href="#">Deny_IP</a>	18 Dec 2018	US bad_bot	Web App Attack
🇬🇧 Anonymous	16 Dec 2018	Runs all Javascript on page, showing up in Google Analytics and ad reporting as an individual unique ... <a href="#">show more</a>	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?

```
SmartTTY - 167.71.122.148
File Edit View SCP Settings Help
cbrenton@cbrenton-lab-testing:/var/log/suricata$ head -2 fast.log
01/30/2018-18:17:06.337205  [**] [1:2027390:2] ET USER_AGENTS Microsoft Device Metadata Retrieval Client Us
er-Agent [**] [Classification: Unknown Traffic] [Priority: 3] {TCP} 10.55.182.100:14314 -> 104.79.151.15:80
01/30/2018-18:17:07.017556  [**] [1:2027390:2] ET USER_AGENTS Microsoft Device Metadata Retrieval Client Us
er-Agent [**] [Classification: Unknown Traffic] [Priority: 3] {TCP} 10.55.182.100:14317 -> 104.79.151.15:80
cbrenton@cbrenton-lab-testing:/var/log/suricata$ grep -v 'Microsoft Device Metadata Retrieval' fast.log | h
ead -2
01/30/2018-18:17:06.662884  [**] [1:2025275:1] ET INFO Windows OS Submitting USB Metadata to Microsoft [**]
[Classification: Misc activity] [Priority: 3] {TCP} 10.55.182.100:14315 -> 40.80.145.38:80
01/30/2018-18:17:06.903781  [**] [1:2025275:1] ET INFO Windows OS Submitting USB Metadata to Microsoft [**]
[Classification: Misc activity] [Priority: 3] {TCP} 10.55.182.100:14315 -> 40.80.145.38:80
cbrenton@cbrenton-lab-testing:/var/log/suricata$ grep -v 'Microsoft Device Metadata Retrieval' fast.log | g
rep -v 'INFO Windows OS Submitting' | head -2
01/30/2018-21:12:15.378653  [**] [1:2027758:2] ET DNS Query for .cc TLD [**] [Classification: Potentially B
ad Traffic] [Priority: 2] {UDP} 10.55.200.10:53219 -> 172.16.200.11:53
01/30/2018-23:17:10.330756  [**] [1:2027758:2] ET DNS Query for .cc TLD [**] [Classification: Potentially B
ad Traffic] [Priority: 2] {UDP} 10.55.200.10:54451 -> 172.16.200.11:53
cbrenton@cbrenton-lab-testing:/var/log/suricata$ grep -v 'Microsoft Device Metadata Retrieval' fast.log | g
rep -v 'INFO Windows OS Submitting' | grep -v 'DNS Query for .cc' | head -2
cbrenton@cbrenton-lab-testing:/var/log/suricata$
```

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 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

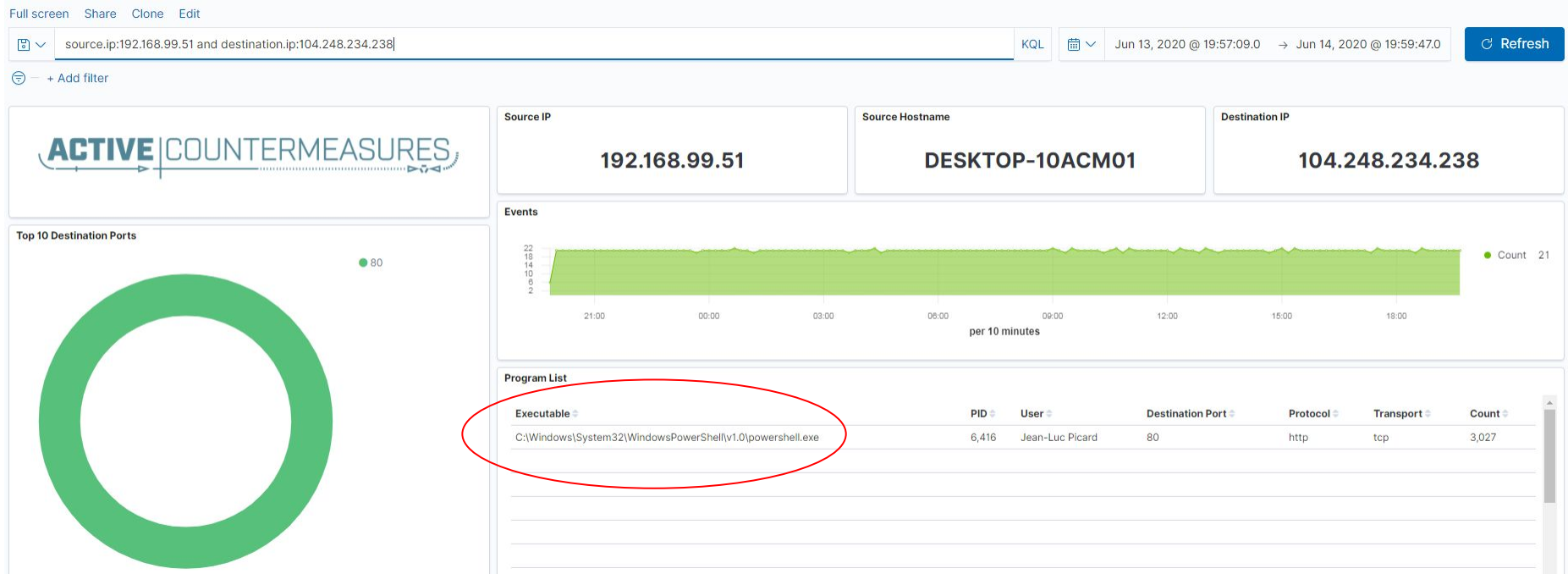
- ▷ The network is the great equalizer
  - You see everything, regardless of platform
  - High level assessment of the terrain
- ▷ You can hide processes but not packets
- ▷ Malware is usually controlled
  - Which makes targeting C2 extremely effective
  - Identify compromise when C2 "calls home"
  - 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 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

# 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)

# 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  $\leq 1/\text{minute}$
  - Red team on long term contract  $\leq 1/\text{week}$
- ▷ 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

24 Hours



# Wireshark-Statistics-Conversations

Wireshark · Conversations · dnsstat2.pcapng

Ethernet · 6   IPv4 · 14760   IPv6 · 1   TCP · 117498   UDP · 177088

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.55.100.100	49778	65.52.108.225	443	1,461	178 k	964	90 k	497	87 k	157.470908	86222.3654	8	8
10.55.100.107	56099	111.221.29.113	443	1,472	179 k	973	91 k	499	88 k	31.952281	86220.1262	8	8
10.55.100.110	60168	40.77.229.82	443	1,086	132 k	722	67 k	364	65 k	31.873157	86160.1197	6	6
10.55.182.100	1567	131.253.34.244	443	157	19 k	104	9718	53	9365	724.206990	84176.7114	0	0
10.55.100.109	53932	65.52.108.233	443	1,233	156 k	816	78 k	417	77 k	14127.883802	72176.1311	8	8
10.55.100.105	60214	65.52.108.195	443	1,691	212 k	1,123	107 k	568	105 k	19775.546806	66599.0023	12	12
10.55.100.103	49918	131.253.34.243	443	3,272	400 k	2,171	204 k	1,101	196 k	17.401930	64698.3708	25	24
10.55.100.104	63530	131.253.34.246	443	1,471	184 k	970	92 k	501	91 k	24194.544203	57413.2785	12	12
10.55.100.111	63029	111.221.29.114	443	836	102 k	543	51 k	293	51 k	109.981977	46663.4804	8	8
10.55.100.108	52989	65.52.108.220	443	755	92 k	502	47 k	253	44 k	18.024716	44615.1658	8	8
10.55.100.106	52918	40.77.229.91	443	717	92 k	473	46 k	244	46 k	25188.024496	41206.9130	8	8
10.55.100.111	62950	40.77.229.40	443	685	88 k	452	44 k	233	44 k	46752.784683	39602.5189	8	9
10.55.100.108	61842	131.253.34.249	443	513	67 k	337	33 k	176	34 k	56989.595829	29143.0082	9	9
10.55.100.106	49875	65.52.108.232	443	427	51 k	283	26 k	144	25 k	118.148709	25185.3859	8	8
10.55.100.103	58675	40.77.229.40	443	1,118	141 k	736	70 k	382	70 k	64721.060443	21661.6952	26	26
10.55.100.106	58537	65.52.108.183	443	299	41 k	194	19 k	105	21 k	67953.761919	16185.8018	9	10
10.55.100.104	60984	65.52.108.217	443	387	51 k	252	25 k	135	26 k	5252.986634	13965.8973	14	15
10.55.100.108	60066	111.221.29.107	443	271	37 k	173	18 k	98	19 k	44519.096770	12585.0109	11	12
10.55.100.105	51403	65.52.108.187	443	243	29 k	162	15 k	81	13 k	83.795249	9081.5345	13	17

86,400 seconds = 24 hours

# The same, but with tshark

```
cbrenton@cbrenton-lab-testing:~/pcaps$ tshark -q -z conv,ip -r dnscat2.pcap  
ng | tr -s ' ' | cut -d " " -f 1,2,3,10 | sort -k 4 -rn | head  
10.55.100.111 <-> 52.84.11.32 86287.865752146  
10.55.100.106 <-> 54.85.75.70 86284.765128713  
10.55.100.104 <-> 52.4.3.93 86284.727643346  
10.55.100.106 <-> 50.19.118.19 86284.543579127  
10.55.100.110 <-> 23.194.107.124 86283.678785025  
10.55.100.106 <-> 34.201.231.171 86283.560157895  
10.55.100.105 <-> 47.88.68.22 86283.461914723  
10.55.100.110 <-> 159.45.170.145 86282.584911853  
192.168.88.2 <-> 208.109.255.35 86178.716231261  
192.168.88.2 <-> 216.69.185.35 86178.696240942  
cbrenton@cbrenton-lab-testing:~/pcaps$
```

Not as accurate as Bro/Zeek!

# 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 CjZ8aQ2AoHDrshUAj 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$
```



# less -S conn.log

```
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path conn
#open 2019-09-09-17-29-18
#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
1552945200.461603 CzbzJEk6RuIxLbGn1 10.0.0.161 58980 239.255.255.250
1552945313.228065 CgwupV2awF4aatt9Bl 10.0.0.204 1900 239.255.255.250
1552945313.228482 CkbFqoRX6QZ0ITPX9 fe80::fd3a:9528:1967:3f5d 1900
1552945320.475783 CDS6Q22dwGlMtJuN4c 10.0.0.161 55745 239.255.255.250
1552945372.938043 CXUNhf1m4ocrnhWu0i 10.0.0.204 138 10.0.0.255
1552945424.133076 CFWt9N1hOWYjRGXsRd 10.0.0.204 53373 1.1.1.1 53
1552945424.268767 Cz131y1nzQSl9bJSA1 10.0.0.204 58048 1.1.1.1 53
1552945434.129470 CwYZRi2XtPjEbIVjQ7 10.0.0.204 55356 1.1.1.1 53
1552945434.149155 Clf1bM306g4t2Vc5d 10.0.0.204 58625 1.1.1.1 53
1552945440.489740 Cmr2ua30QNXZnM0qGj 10.0.0.161 53029 239.255.255.250
1552945476.411242 CW8fH52sNJIV9WSZZ9 fe80::fd3a:9528:1967:3f5d 135
1552945476.413299 C22eRO22JbnnxuqGje fe80::488f:bef:fec:aeaa 136
conn.log
```

# Cumulative talk time with Zeek

```
thunt@thunt-one-day:~/lab1$ cat conn.log | bro-cut id.orig_h id.resp_h duration |  
sort | grep -v '-' | datamash -g 1,2 sum 3 | sort -k 3 -rn | head -10  
10.55.100.100      65.52.108.225      86222.365445  
10.55.100.107      111.221.29.113      86220.126151  
10.55.100.110      40.77.229.82        86160.119664  
10.55.100.109      65.52.108.233      72176.131072  
10.55.100.105      65.52.108.195      66599.002312  
10.55.100.103      131.253.34.243      64698.370547  
10.55.100.104      131.253.34.246      57413.278323  
10.55.100.111      172.217.8.198       56057.255003  
10.55.100.111      111.221.29.114      46658.400629  
10.55.100.108      65.52.108.220      44615.165823  
thunt@thunt-one-day:~/lab1$
```

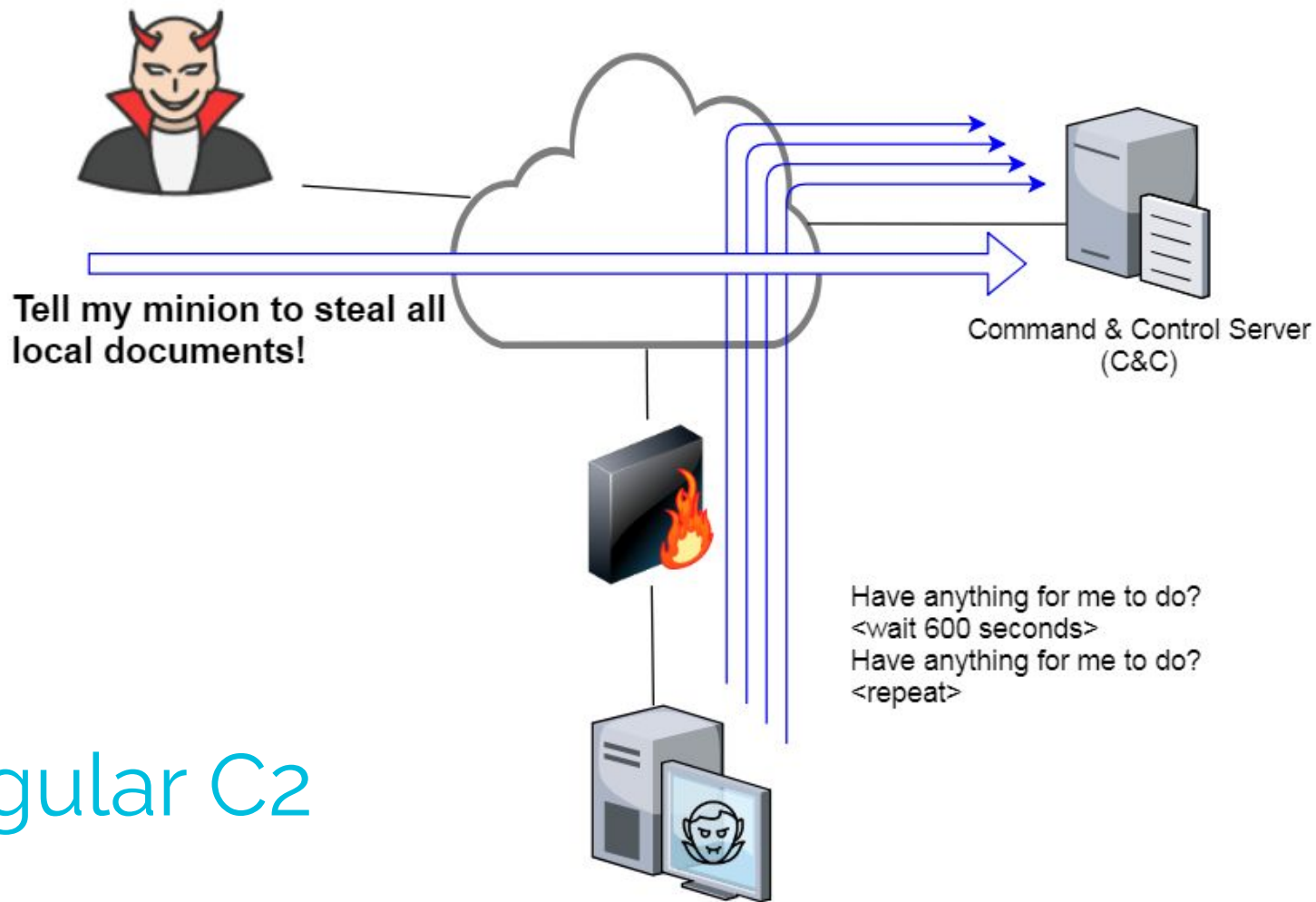
# What about firewalls?

- ▷ 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



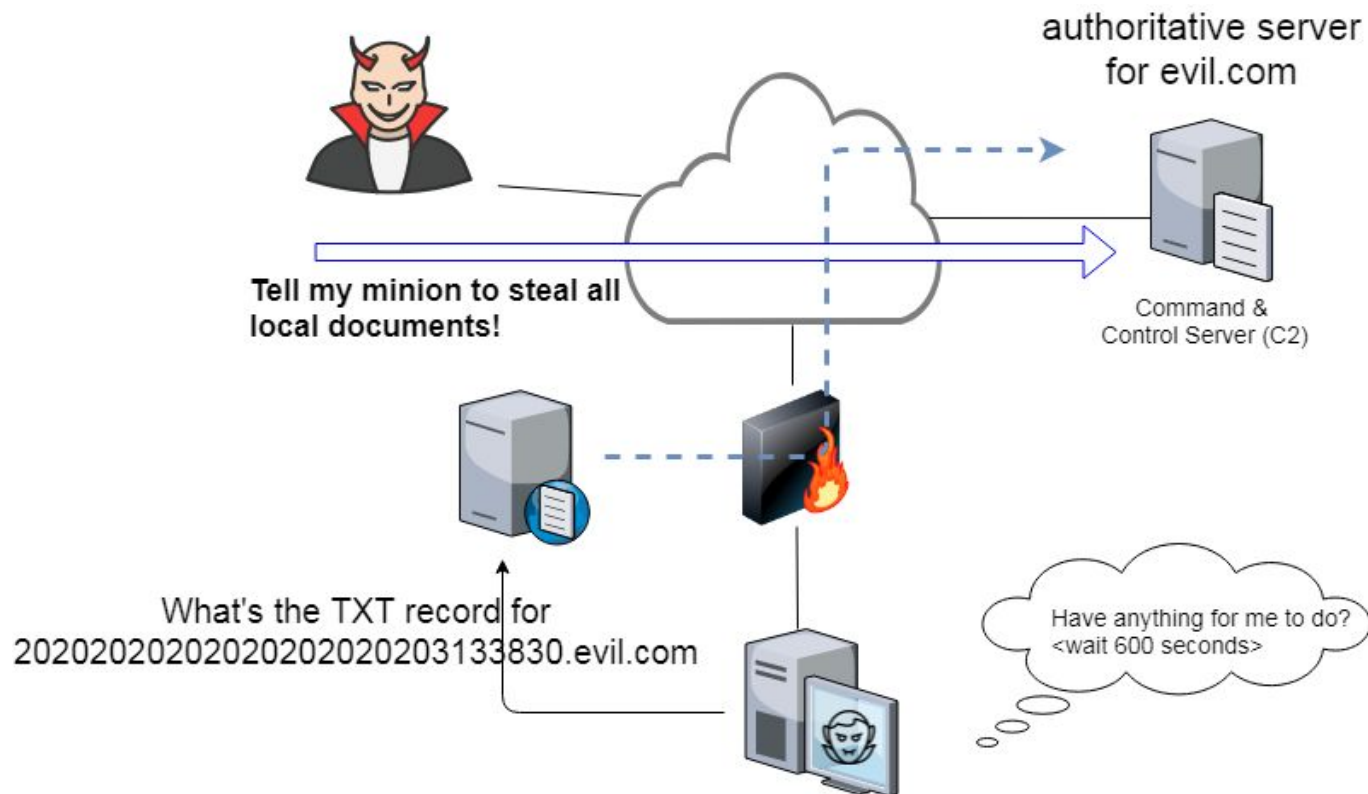
# What is a beacon?

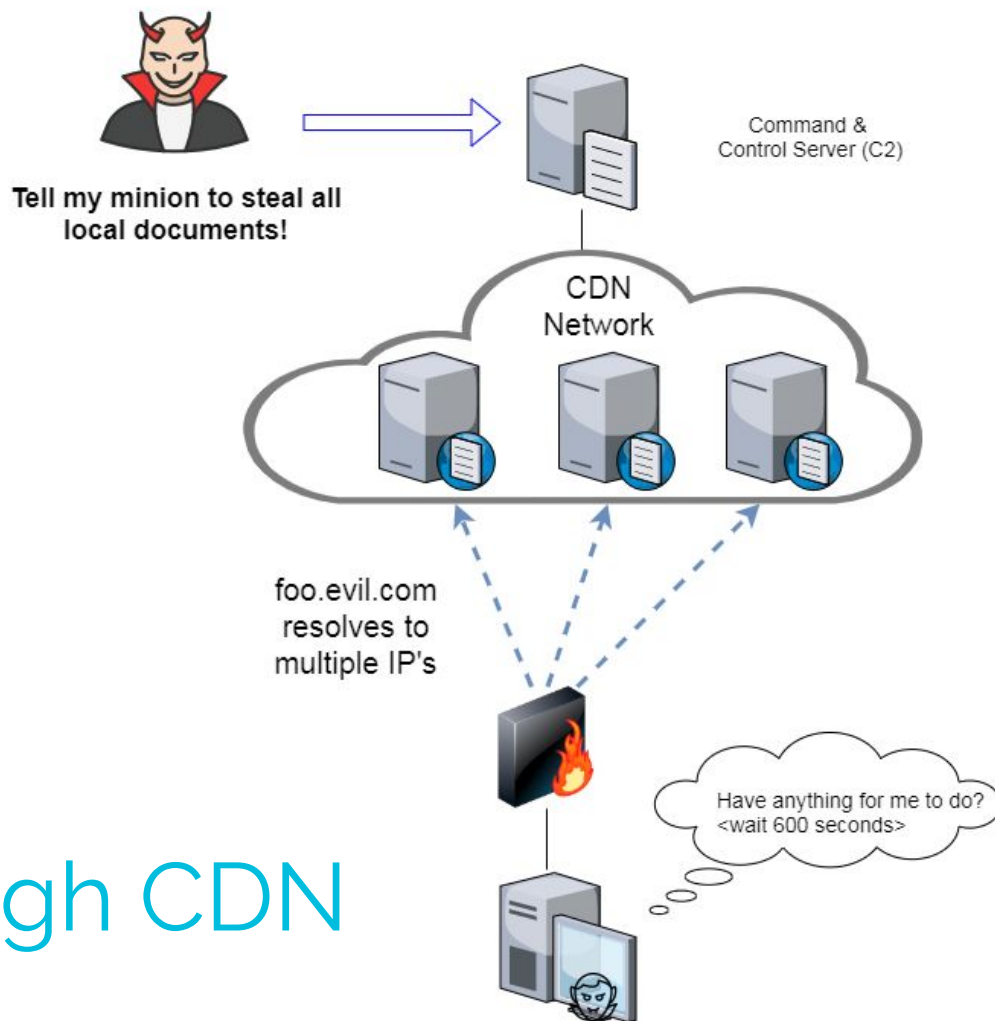
- ▷ Repetitive connection establishment between two IP addresses
  - Easiest to detect
- ▷ Repetitive connection establishment between internal IP and FQDN
  - Beacon broken up over multiple IP's
    - Usually a CDN provider
  - Target IPs also destination for legitimate traffic
  - Far more difficult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



Regular C2

# C2 over DNS



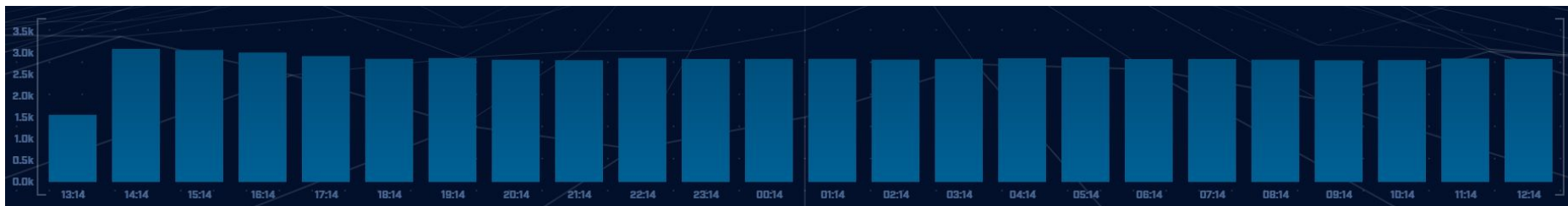


## C2 through CDN

# Beacon detection based on timing

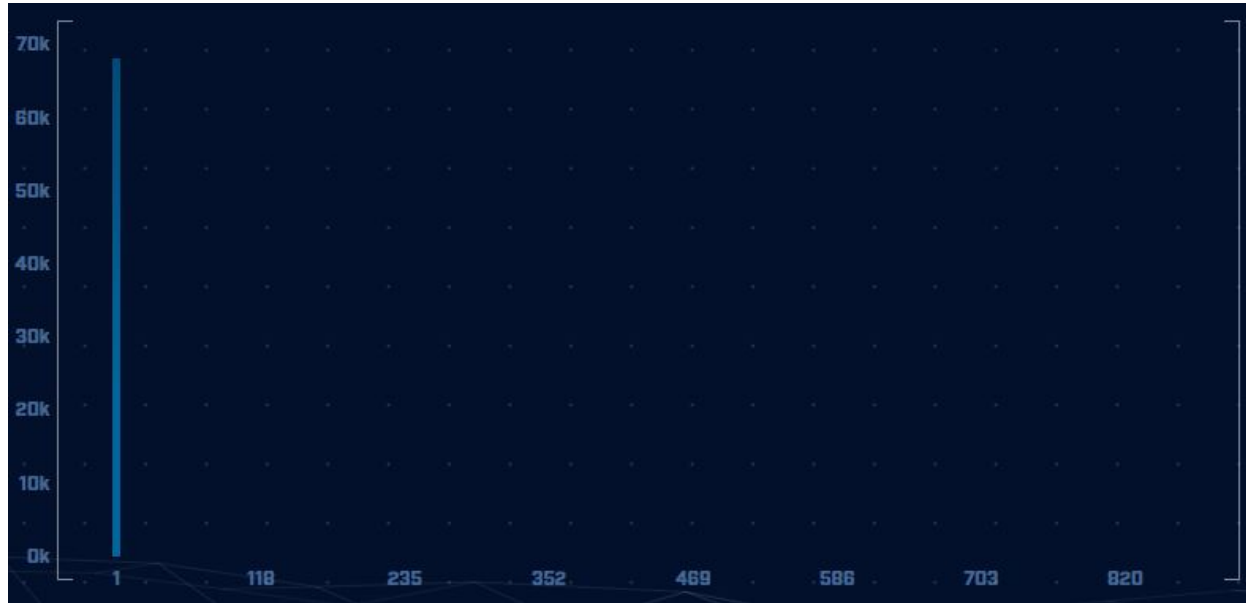
- ▷ May follow an exact time interval
  - Technique is less common today
  - Detectable by k-means
  - Potential false positives
- ▷ May introduce "jitter"
  - Vary connection sleep delta
  - Avoids k-means detection
  - False positives are extremely rare
- ▷ Short enough delta for terminal activities

# Connection quantity VS time



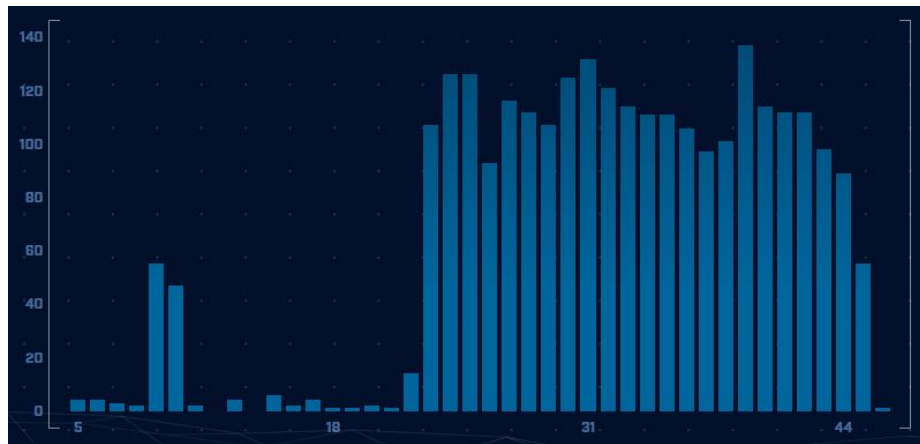
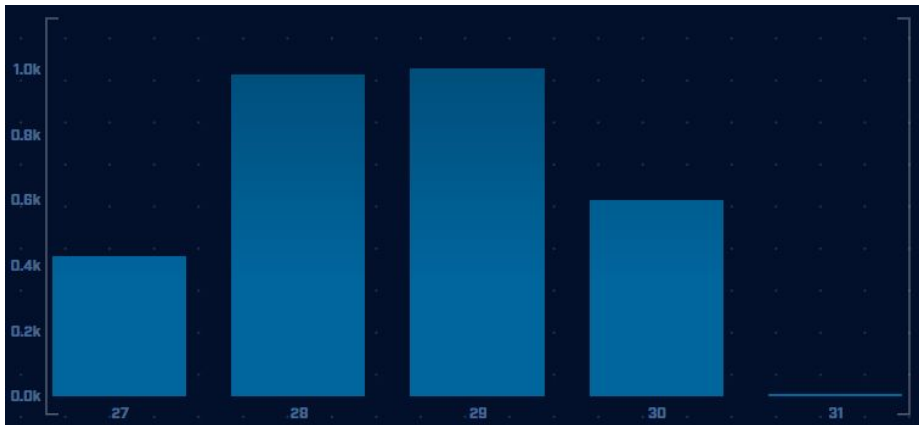
Each bar represents the number of times the source connected to the destination during that one hour time block

# Connect time deltas with no jitter



How often a specific time delta was observed

# Connection time deltas with jitter



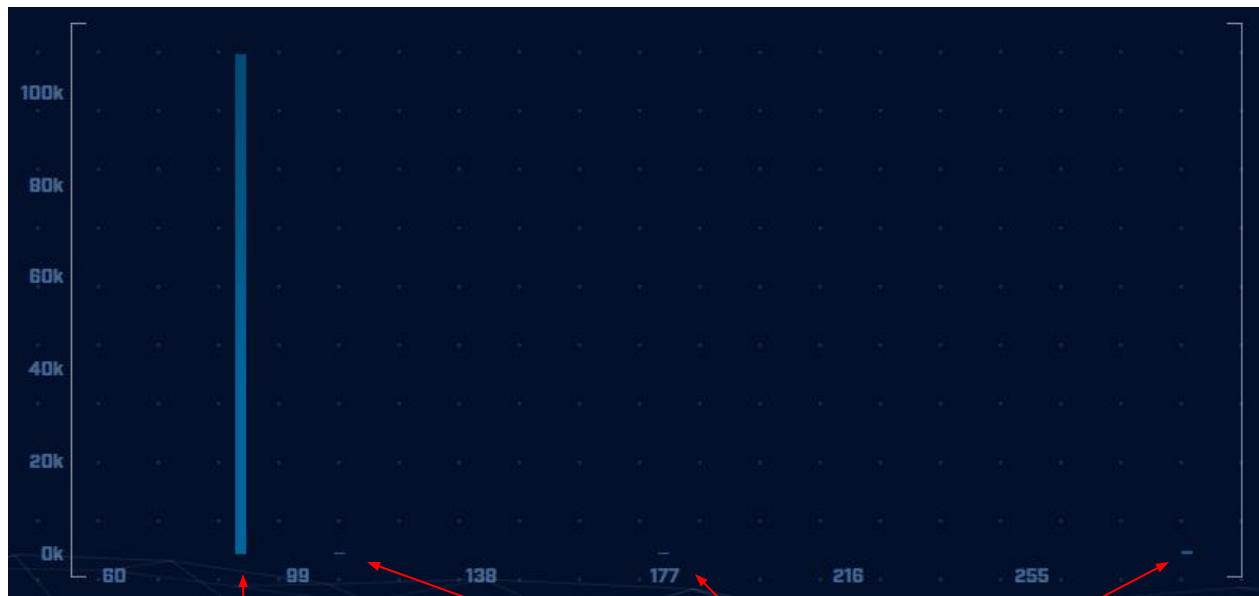
Cobalt Strike will typically produce a bell curve



# 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 68.183.138.51 | 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 miscategorization:
  - 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
```

Show all instances of unique FQDNs queried

```
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
```

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

```
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
```

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 of FQDN queries
  - But no one ever connects to these systems

# Normal DNS query patten

Subdomain Threshold: 0

AI HUNTER

— DATABASE: DNSCAT2-BEACON  
— MODULE: DNS  
— VIEW: DNS ANALYSIS

Subdomains	Lookups	Domain
62468	109227	r-1x.com
62466	108911	dnsc.r-1x.com
154	27381	akamaiedge.net
125	13907	akadns.net
121	7110	edgekey.net
101	13297	amazonaws.com
90	13259	elb.amazonaws.com

DNS Queries [ 3 ]

Direct Connections [ 13 ]

Host	Count
10.55.100.111	889
10.55.100.108	532
10.55.100.109	489
10.55.100.100	477
10.55.100.103	462
10.55.100.104	446
10.55.100.110	443
10.55.100.107	443
10.55.100.106	442

1 / 9880

# Things that make you go "hummm"

Subdomain Threshold  
0

AI HUNTER  
-- DATABASE: DNSCAT2-BEACON  
-- MODULE: DNS  
-- VIEW: DNS ANALYSIS

Subdomains	Lookups	Domain
62468	109227	r-1x.com
62466	108911	dnsc.r-1x.com
154	27381	akamaiedge.net
125	13907	akadns.net
121	7110	edgekey.net
101	13297	amazonaws.com
90	13259	elb.amazonaws.com

DNS Queries [1]  
Direct Connections [1]

Host	Count
192.168.88.2	108858

1 / 9680


# 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) -  
-      -      -      -      -      FhGf5d4pejzo7Ob31l      -      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

# Invalid certificate check



Host	Seen	Invalid Certificate Code	Port:Protocol:Service	Sources
104.40.53.192	1	unable to get local issuer certificate	443:tcp:ssl	10.55.100.108
134.170.51.190	1	unable to get local issuer certificate	443:tcp:ssl	10.55.100.109
65.55.252.190	1	unable to get local issuer certificate	443:tcp:ssl	10.55.200.10
52.224.179.205	1	unable to get local issuer certificate	443:tcp:ssl	10.55.100.103
65.55.252.93	1	unable to get local issuer certificate	443:tcp:ssl	10.55.200.10

Extended Validation (EV) certs are more trusted and can assign negative threat points

# 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://apility.io/search/<IP address>>

<https://dnslytics.com/ip/<IP address>>

<https://transparencyreport.google.com/safe-browsing/search?url=<IP, FQDN or URL>>

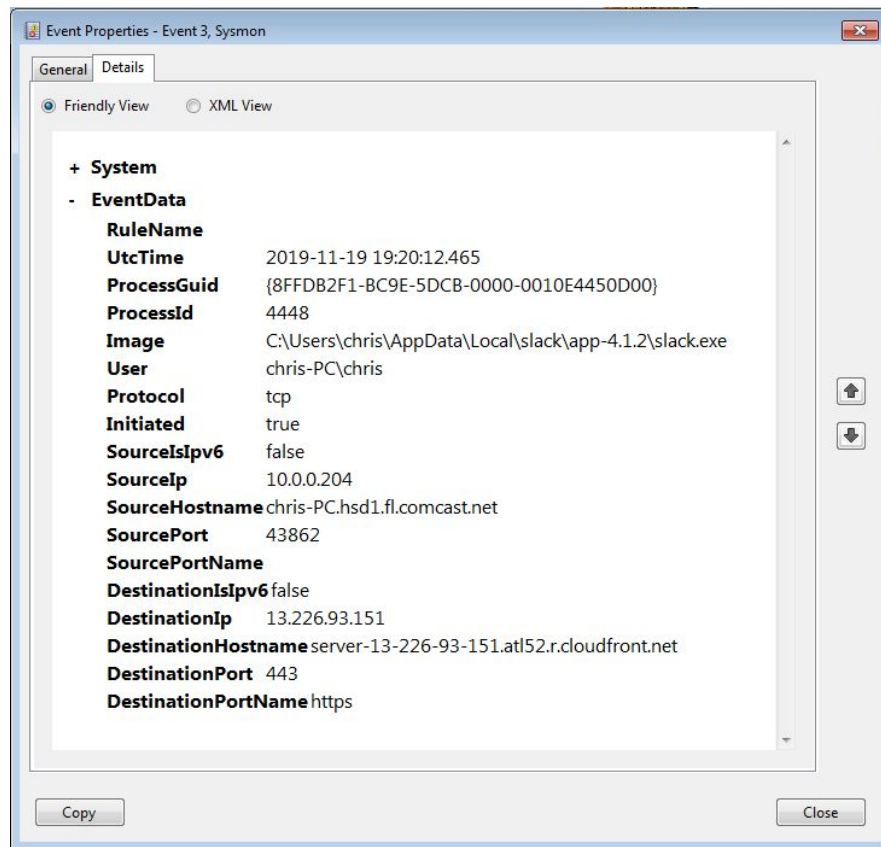
# Internal system

- ▷ 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 -i eth0 -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

The image shows a Wireshark packet capture analysis of a file named `perimeter_class.cap`. The main packet list pane displays a series of network packets. A filter is applied: `ip.addr == 148.78.247.10`. The selected packet (No. 98594) is a TCP SYN packet from source 12.33.247.4 to destination 148.78.247.10, port 80. The packet details pane shows the following information:

- Frame 98594: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
- Ethernet II, Src: HewlettP\_ea:20:ab (00:50:8b:ea:20:ab), Dst: Computer\_20:7d:e3 (00:b0:d0:20:7d:e3)
- Internet Protocol Version 4, Src: 148.78.247.10, Dst: 12.33.247.4
- Transmission Control Protocol, Src Port: 26268, Dst Port: 80, Seq: 0, Len: 0
  - Source Port: 26268
  - Destination Port: 80
  - [Stream index: 648]
  - [TCP Segment Len: 0]
  - Sequence number: 0 (relative sequence number)
  - [Next sequence number: 0 (relative sequence number)]
  - Acknowledgment number: 0
  - 1010 .... = Header Length: 40 bytes (10)
  - Flags: 0x002 (SYN)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000 00 b0 d0 20 7d e3 00 50 8b ea 20 ab 08 00 45 00  ...}..P...E-
0010 00 3c f7 29 00 00 31 06 04 14 94 4e f7 0a 0c 21  (<...)..1. ...N...!
0020 f7 04 66 9c 00 50 64 37 ff 9d 00 00 00 a0 02  --f--Pd7-----
0030 ff ff a8 97 00 00 02 04 05 b4 01 03 03 00 01 01  ....HD-- ..addr
0040 08 0a 00 ec 48 44 00 00 00 00 61 64 64 72
```

The status bar at the bottom indicates: `perimeter_class.cap`, `Packets: 197147 · Displayed: 5462 (2.8%)`, and `Profile: Default`.

# 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>.<P>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 | sort -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 | sort | 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

```
ubuntu@ip-172-31-26-215:~/working/beacon$ rita show-beacons beacon | head -10
Score,Source,Destination,Connections,Avg Bytes,TS Range,DS Range,TS Mode,DS Mode,TS Mode Count,
DS Mode Count,TS Skew,DS Skew,TS Dispersion,DS Dispersion,TS Duration
0.999774,192.168.88.2,165.227.88.15,108858,199.578,980,201,1,89,53341,108319,0,0,0,0,1
0.99182,192.168.88.2,13.107.3.1,57,190.07,5902,3,3154,73,9,35,0,0,1,0,0.98537
0.99182,192.168.88.2,13.107.3.2,60,193.833,7576,3,3154,73,10,43,0,0,1,0,0.98537
0.958989,192.168.88.2,205.233.73.201,163,152,31,0,542,76,11,163,0,0,7,0,0.988426
0.955013,192.168.88.2,216.229.4.69,164,148.756,32,0,519,76,9,164,0,0,8,0,0.997905
0.949074,192.168.88.2,216.218.220.101,164,152,32,0,518,76,12,164,0,0,9,0,0.995602
0.939216,192.168.88.2,66.228.58.20,164,151.537,31,0,519,76,11,164,-0.0588235,0,9,0,0.995278
0.93308,192.168.88.2,108.61.56.35,163,152,31,0,543,76,11,163,-0.125,0,8,0,0.991308
0.92634,192.168.88.2,45.33.48.4,164,152,31,0,514,76,12,164,-0.2,0,7,0,0.992535
ubuntu@ip-172-31-26-215:~/working/beacon$ _
```

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
  - Destination IP
  - Duration of each connection
- ▷ 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
```

# 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 this connection?



# AbuseIPDB data on 2nd IP

 **AbuseIPDB**

[Home](#) [Report IP](#) [Bulk Reporter](#) [Pricing](#) [About](#) [FAQ](#) [Documentation](#) [Statistics](#) [IP Tools](#) [Contact](#) [LOGIN](#) [SIGN UP](#)


## AbuseIPDB » 52.179.219.14

Check an IP Address, Domain Name, or Subnet  
e.g. 67.7.80.242, microsoft.com, or 5.188.10.0/24

67.7.80.242


CHECK

**52.179.219.14** was not found in our database

ISP	Microsoft Corporation
Usage Type	Data Center/Web Hosting/Transit
Domain Name	microsoft.com
Country	 United States of America
City	Boydton, Virginia

IP info including ISP, Usage Type, and Location provided by [IP2Location](#).  
Updated monthly.

[REPORT 52.179.219.14](#) [WHOIS 52.179.219.14](#)



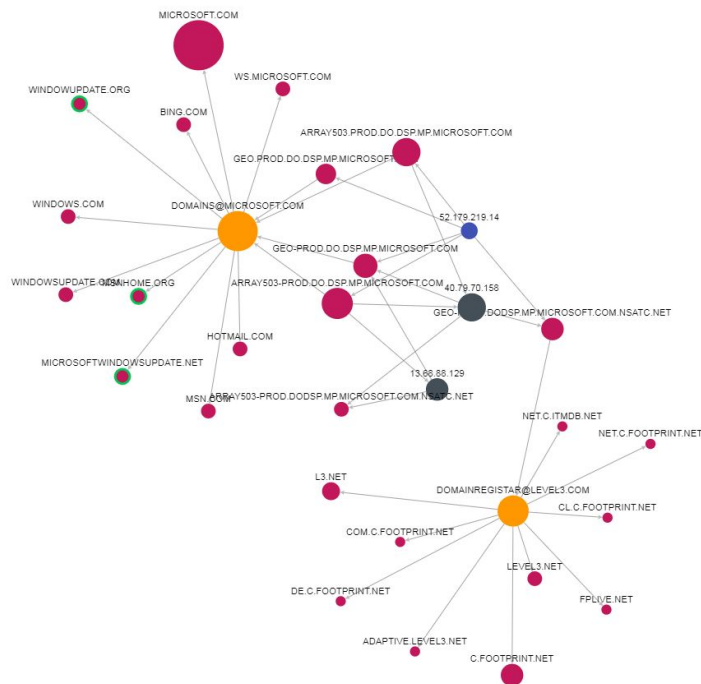
Navigate code across all repos and code hosts with Sourcegraph universal code search. Try it now.

ADS VIA CARBON

IP Abuse Reports for 52.179.219.14:

This IP address has not been reported. [File Report](#)

# ThreatCrowd data on 2nd IP



IP > 52.179.219.14

Welcome! Right click nodes and scroll the mouse to navigate the graph.

More information on this IP is in [AlienVault OTX](#)

## IS THIS MALICIOUS?

Yes No

## IP WHOIS

Property	Value
Location	Wilmington, United States
Country	United States

## REVERSE DNS

Domain	Date
<a href="#">array503.prod.do.dsp.mp.microsoft.com</a>	2021-02-11
<a href="#">geo.prod.do.dsp.mp.microsoft.com</a>	2020-12-18
<a href="#">array503-prod.do.dsp.mp.microsoft.com</a>	2020-12-14
<a href="#">geo-prod.dodsp.mp.microsoft.com.nsatc.net</a>	2020-09-23
<a href="#">geo-prod.do.dsp.mp.microsoft.com</a>	2020-07-26

## IP CLASSES

52.179.219.x=[Browse](#) , 52.179.219.x.x=[Browse](#) | [View on ThreatMiner](#)



# 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 "hummm"

```
thunt@thunt:~/lab1$ ngrep -q -I trace1.pcap host 192.168.99.51 and host 104.248.234.238 | 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/eghmbblnphlaefbmmnoenohhoncmcepapefjjekpleokhjfmnmijghedkienpli
dbbcmgdjldbegpeemiboacnfcnpbnhlmjbpcejfpecdioiddklfegefcbjbcnagjclnoiipajlpkk
egakmpdddojnlphegeehaacmofggdfkagpbighfkndllaamndepdanhnogedkaodhgakiigohehin
oolnaobdiiokpebghapnghbebkpiffooljden;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/eghmabblnphlaefbmmnoenohhoncmcepap
efjjekpleokhjfmnmijghedkienplidbbcmgdjldbegpeemiboacnfcpcbnnhlmjbpcejfpecdioiddkl
fegefcjbcbnagjclnoijpajlpkkegakmpdddojnlphegeehaacmofggdfkagpbighfkndllaamndepdanh
ogedkaodhgakiigoheminoalnaobdiiokepbghapnghbeebkepiffooljden;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

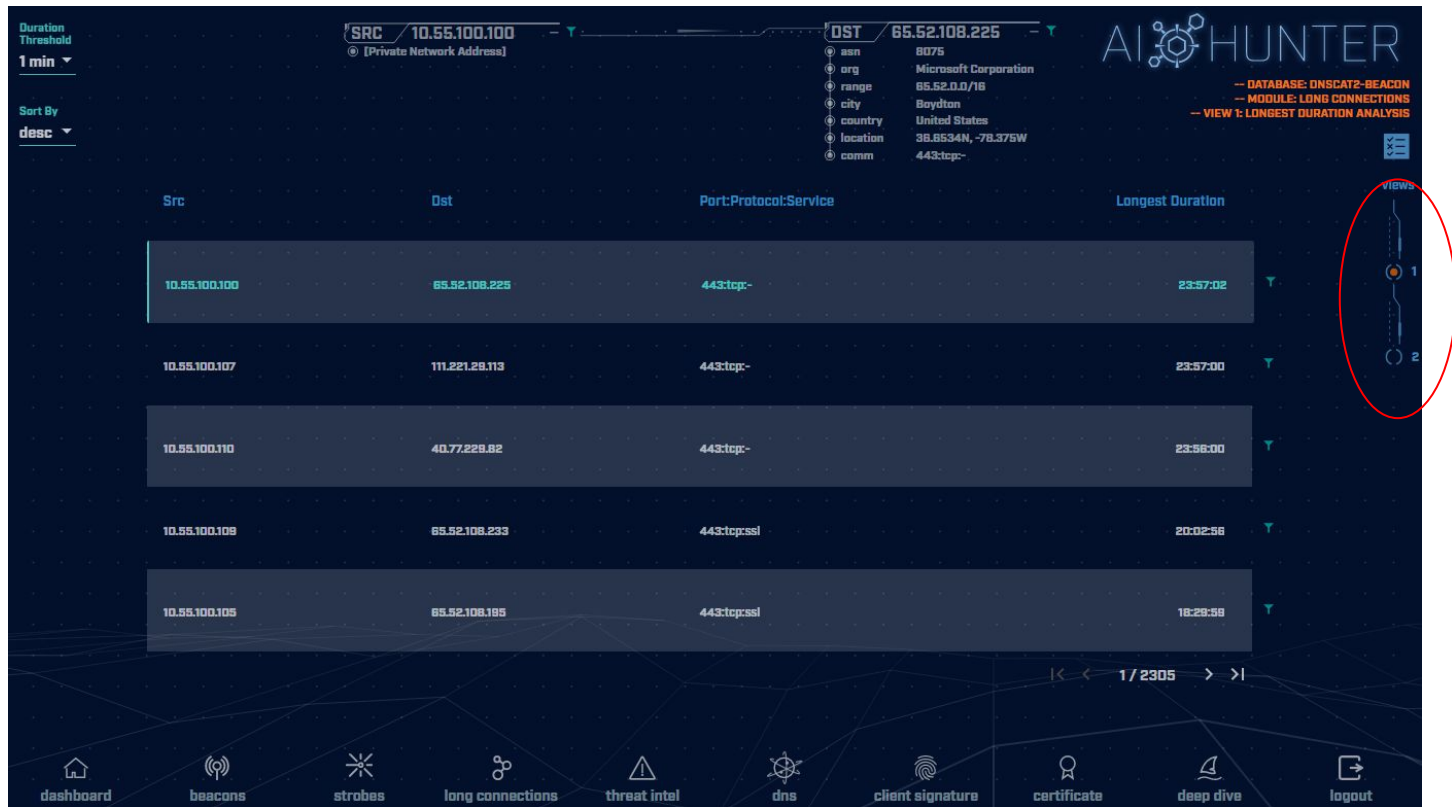
- ▷ Do we feel confident in flagging anything we have seen as requiring incident handling?
- ▷ Are there any connections that need more research?
  - What should this research be?
  - Do we need to involve any other teams?
  - If we need more data collection, for how long?

# Quick demo

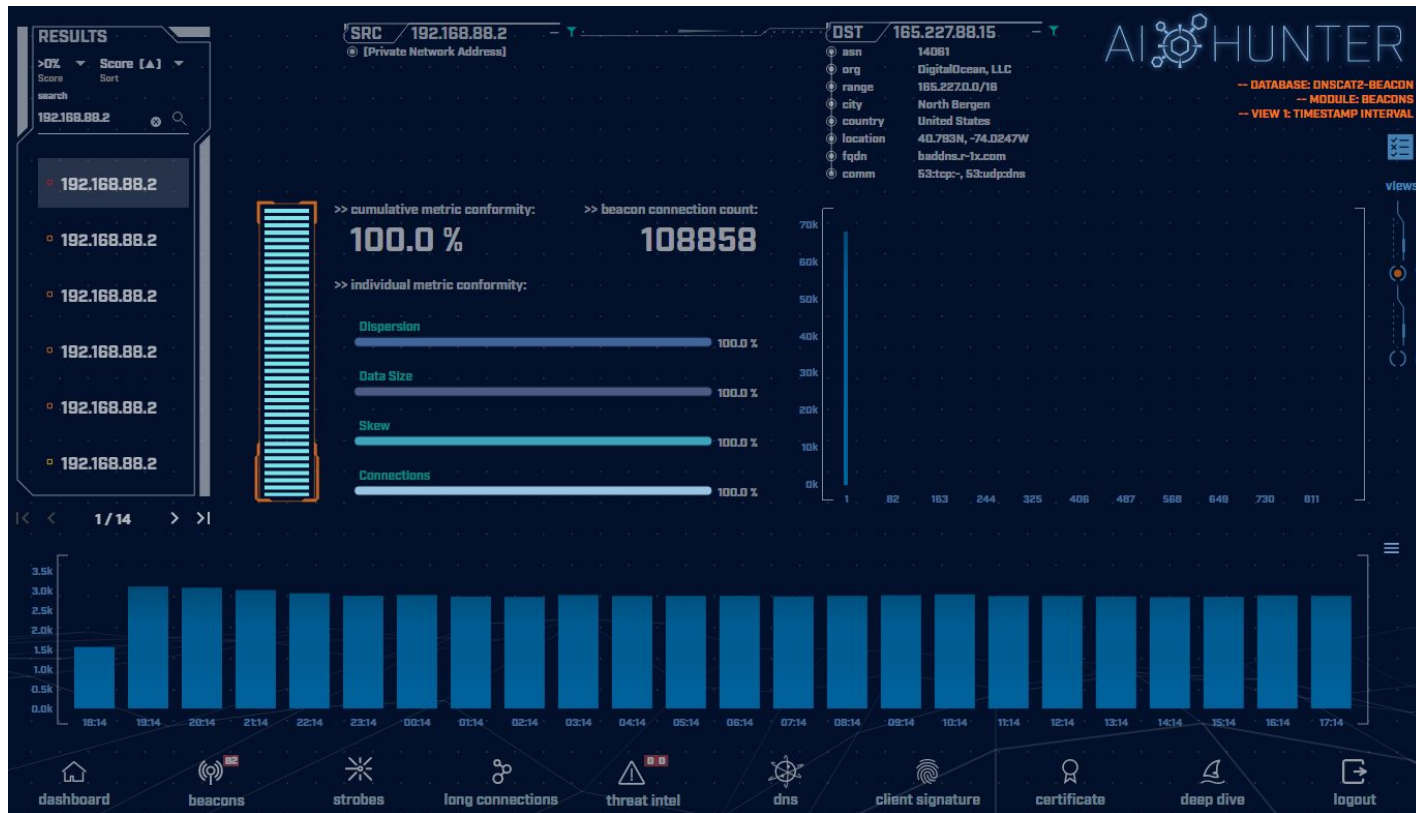
- ▷ Similar data, seen through AI-Hunter
- ▷ Inexpensive commercial solution
- ▷ Automates much of the hunting process



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

The screenshot shows a network tool interface with a dark blue background. At the top, a tab labeled 'DST' is selected, showing the IP address '165.227.88.15'. Below this, a list of attributes and their values is displayed:

asn	14061
org	DigitalOcean, LLC
range	165.227.0.0/16
city	North Bergen
country	United States
location	40.793N, -74.024W
fqdn	baddns.r-1x.com
comm	53:tcp:-, 53:udp:0

To the right of this table, a list of resources to dig deeper is shown in a white box:

- deep dive
- AbuseIPDB
- AlienVault
- apility.io
- ThreatCrowd
- Shodan
- Google
- Google DNS
- VirusTotal
- SecurityTrails

In the bottom left corner, there is a vertical bar chart with labels '70k' and '60k'.

Subdomain Threshold 0

AI HUNTER

-- DATABASE: DNSCAT2-BEACON  
-- MODULE: DNS  
-- VIEW: DNS ANALYSIS

Subdomains	Lookups	Domain
62468	109227	r-1x.com
62466	108911	dnsc.r-1x.com
154	27381	akamaiedge.net
125	13907	akadns.net
121	7110	edgekey.net
101	13287	amazonaws.com
90	13259	elb.amazonaws.com

DNS Queries [1]

Direct Connections [1]

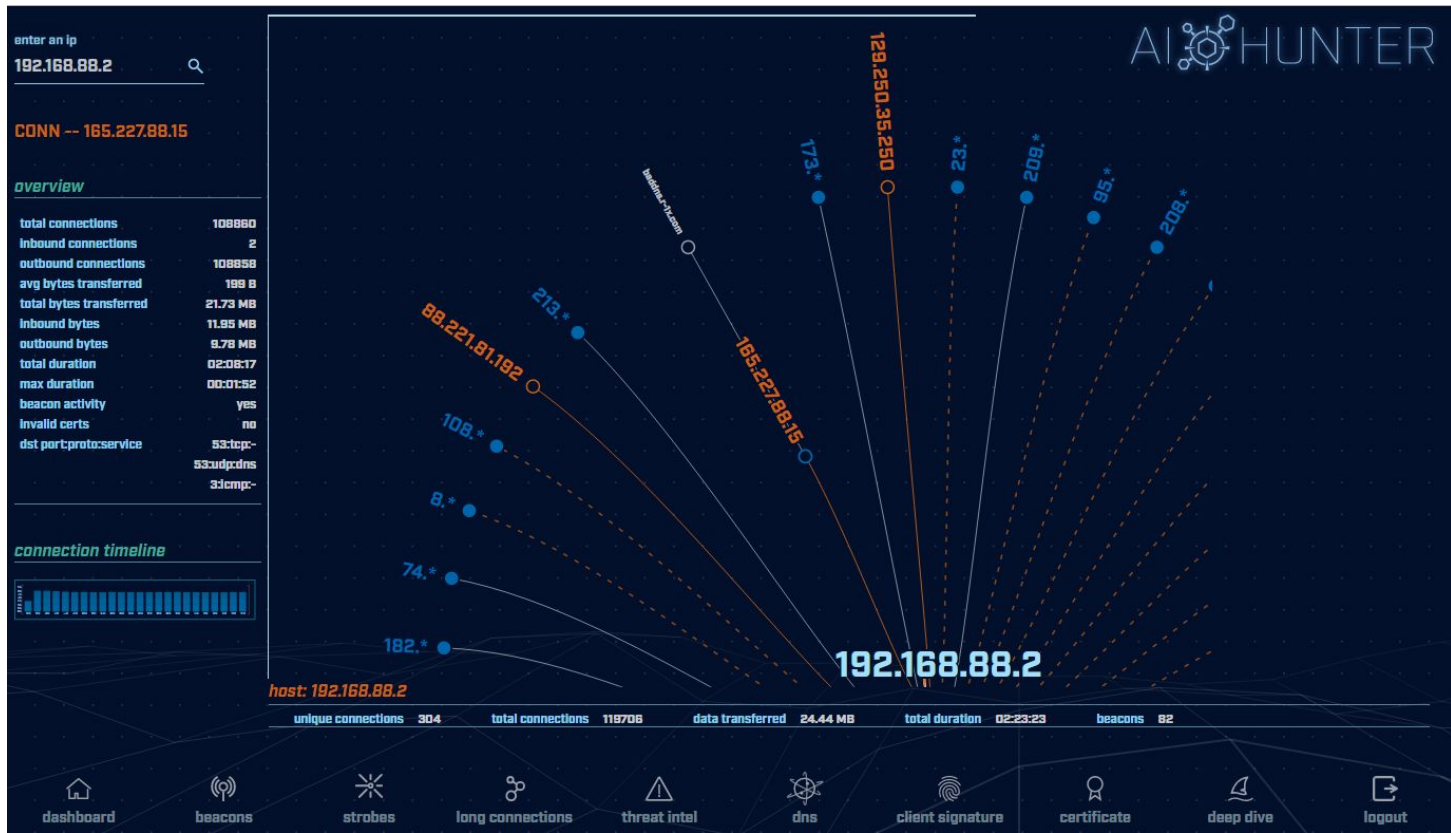
Host	Count
192.168.88.2	108858

1 / 9680

dashboard beacons strobos long connections threat intel dns client signature certificate deep dive logout

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

- ▷ Getting started with Packet Decoding
  - 8/20/21 - 16 hours over 4 days
  - Pay what you want

<https://wildwesthackinfest.com/antisyphon/getting-started-with-packet-decoding-w-chris-brenton/>

- ▷ Advanced Threat Hunting
  - 7/30/21 - 16 hours over 4 days
  - Live at WWHF in Sept

<https://wildwesthackinfest.com/antisyphon//advanced-network-threat-hunting-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](mailto:chris@activecountermeasures.com)

# 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
- ▷ Use "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?