



Network Cyber Threat Hunter Training

Level 1

Thanks to our sponsors!



Check out the ACM calendar!

ACTIVE COUNTERMEASURES

HOME AI-HUNTER™ FREE TOOLS ABOUT EDUCATION CONTACT

Search for events Find Events List Month

< > Today Now onwards ▾

October 2020

WED 7

October 7 @ 2:00 pm - 3:00 pm EDT

Webcast – Passer – Effortless Network Knowledge! w/ Bill Stearns

GoToWebinar

Passer is a network sniffer that teaches you about your network - with no extra work for you! By watching network traffic it [...]

SAT 17



October 17 @ 12:00 pm - 4:00 pm EDT

Training – Cyber Threat Hunting w/ Chris Brenton

GoToWebinar

Chris Brenton with Active Countermeasures is conducting a free, one-day, Cyber Threat Hunting Training online course. This will be a live online course [...]

< Previous Events Next Events >



<https://www.activecountermeasures.com/events/>

Classes I'll be teaching

- Advanced network threat hunting
 - November 2nd - 5th
 - WWHF @ Secure WV

<https://wildwesthackinfest.com/>

Before we get started

- ▷ You'll need the class VM to do the labs

<https://www.activecountermeasures.com/cyber-threat-hunting-training-course/>

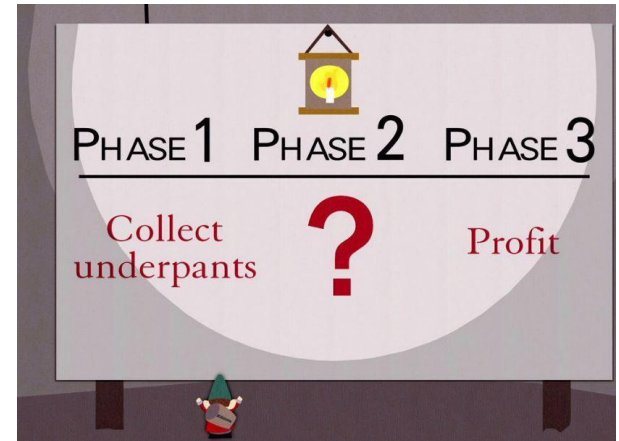
- ▷ VMWare or VirtualBox version
- ▷ Login info:
 - Name: thunt
 - Pass: aybab2u
- ▷ This should have been done before class :-)
- ▷ Slides are now available here as well

Logistics

- ▷ 10 minute break at top of 1st and 3rd hour
- ▷ 30 minute break at midpoint
- ▷ 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
- ▷ Please keep an open mind
- ▷ Prime cognitive bias fodder



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

Anatomy of a Syslog message

Header	Part 1	Part 2
Jun 17 14:48:27 fs1	sshd(pam_unix)[10633]:	session opened for user cbrenton by (uid=0)
15:28:46.968346	IP 172.30.1.7.514 > 172.30.2.10.514:	UDP, length 71
0x0000:	4500 0063 b4d1 4000 3f11 2b6b ac1e 0107	E..c..@.?.+k....
0x0010:	ac1e 020a 0202 0202 004f 812c 3c33 383eO.<38>
0x0020:	7373 6864 2870 616d 5f75 6e69 7829 5b33	sshd(pam_unix)[3
0x0030:	3131 3430 5d3a 2073 6573 7369 6f6e 206f	1140]:.session.o
0x0040:	7065 6e65 6420 666f 7220 7573 6572 2063	pened.for.user.c
0x0050:	6272 656e 746f 6e20 6279 2028 7569 643d	brenton.by.(uid=
0x0060:	3029 0a	0).

(Facility X 8) + Severity = Priority (sed for sorting)

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



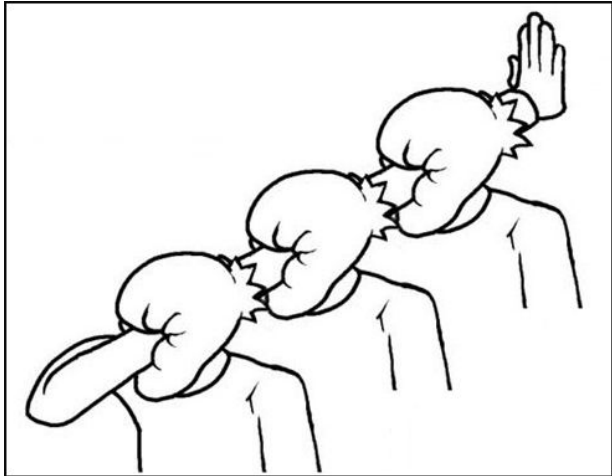
OK to still wear
parachute pants?

Is there data showing its broken?

- ▷ Out of 1,200 orgs surveyed, 71% breached
 - [451 Group 2018 Global Threat Report](#)
- ▷ Less than 50% of breaches get detected internally
 - [2019 Verizon Breach Report](#)
- ▷ 40% of orgs that think they are threat hunting, are not
 - [SANS 2019 Threat Hunting Survey](#)
- ▷ **On average, 191-197 days to ID a breach & 66-69 days to contain it**
 - [Ponemon Institute 2017 study](#)
 - [2018 IBM Global Breach Study](#)

2019 Verizon Breach Report

But we review
our SIEM logs!



POS devices 100% external detection

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, 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 ... show more	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 ... 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 Analytics 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?

```
SmartTTY - 167.71.123.148
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$
```

cbrenton@cbrenton-lab-testing /var/log/suricata

SCP: No transfers

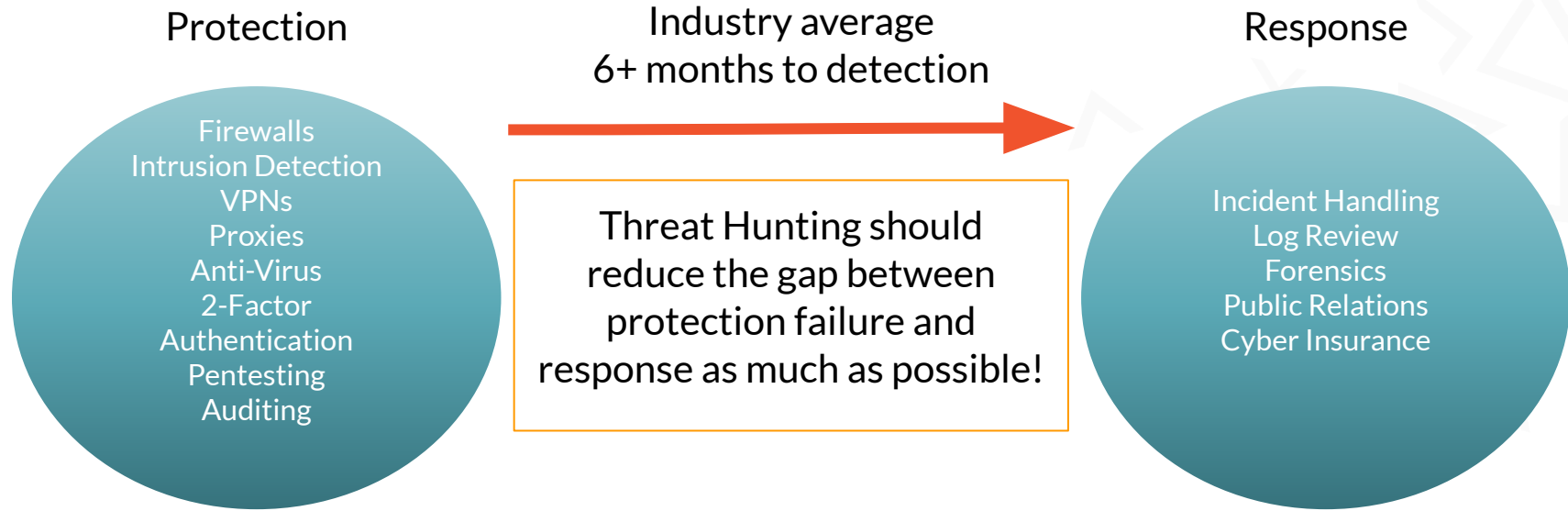
364KB sent, 1301KB received

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

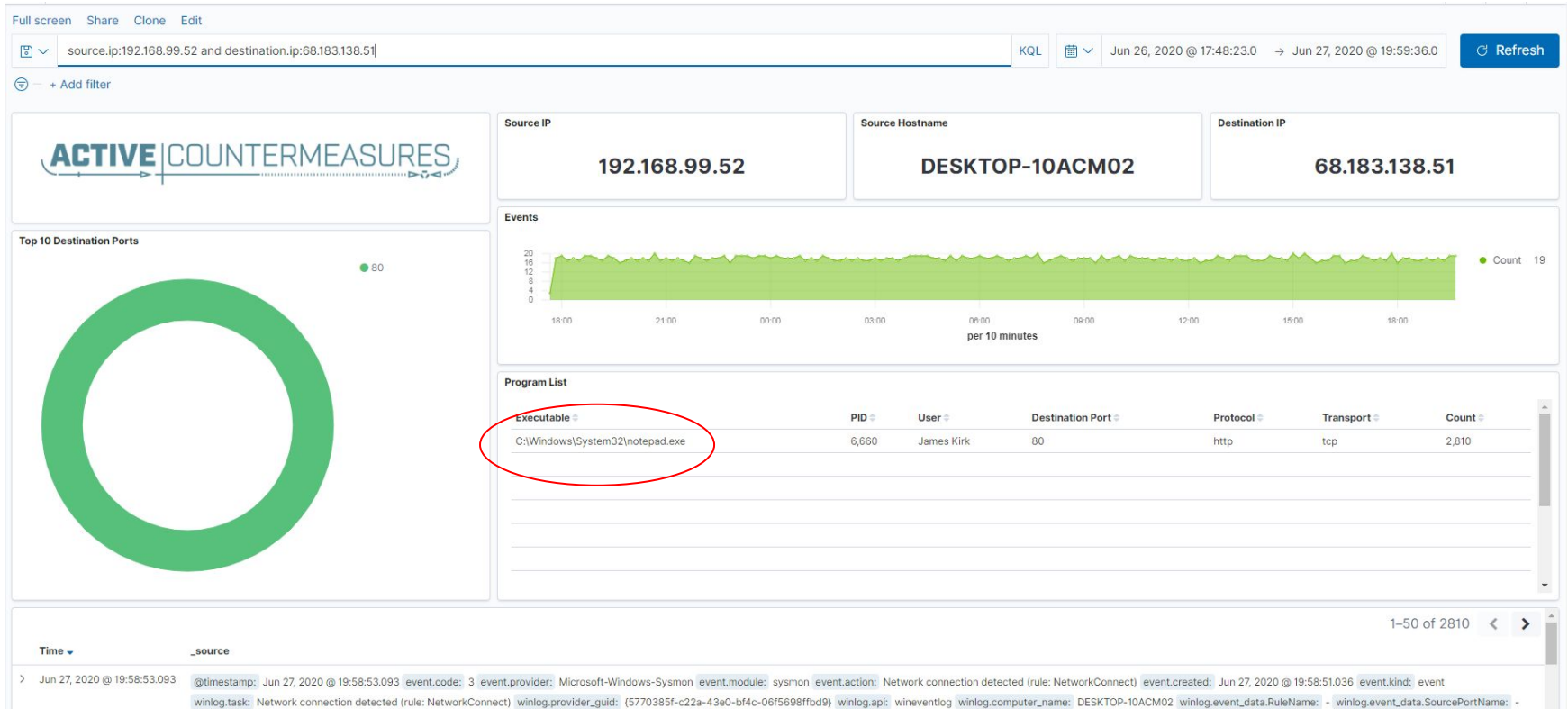
Proposal - Start with the network

- ▷ 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



Where to Start

- ▷ Traffic in and out of the Internet link(s)
 - 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
 - For every internal IP, each external IP connected to
 - 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



C2 Detection Techniques

C2 Detection - What is the goal?

- ▷ Identify persistent connections
- ▷ Internal to external
- ▷ Are the hosts in constant communication?
- ▷ Does connection appear automated?
- ▷ Can the connection be explained?
 - Valid business need
 - These will be whitelisted/exception listed
- ▷ Results screened for potential C2

Why start with persistent conns?

- ▷ C2 can look like normal traffic
 - Attackers are better at being protocol compliant
 - May not be any "signatures" to match
 - But sometimes you get lucky
- ▷ NIDS useful for detects of old attacks
 - Faster than doing a full hunt
 - Don't expect this to catch everything/anything

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 ≤ 60 seconds
 - Red team on long term contract \leq week
- ▷ Focus will be on the bad guys

Where to Start

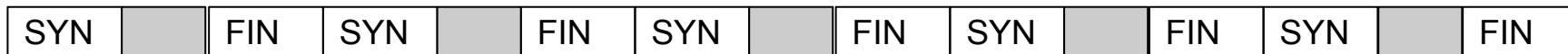
- ▷ Packet captures or Bro/Zeek data
- ▷ Analyze in large time blocks
 - More data = better fidelity
 - Minimum of 12 hours, 24 is ideal
- ▷ Analyze communications in pairs
 - For every internal IP, each external IP it connected to
 - Ignore internal to internal (high false positive)

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



Long connections challenges

▷ TCP

- Time between first SYN and last SYN/ACK

▷ UDP

- Stateless so no flags to work with
- Identify a timeout for session reset
- Most firewalls ≤ 30 seconds same "session"

▷ Other transports

- Similar to UDP
- ICMP can be extra problematic

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$
```

What about UDP?

Wireshark · Conversations · dnscat2.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
10.55.100.197	5353	224.0.0.251	5353	24	4392	24	4392	0	0	388.52853	82801.5640	
10.55.182.100	63547	96.45.33.73	8888	938	86 k	469	49 k	469	36 k	82.168722	86168.7296	
10.55.182.100	63547	173.243.138.201	8888	938	86 k	469	49 k	469	36 k	82.168731	86168.7522	
10.55.182.100	63547	173.243.138.200	8888	941	86 k	471	49 k	470	36 k	82.168732	86168.7489	
10.55.182.100	63548	96.45.33.73	8888	938	86 k	469	49 k	469	36 k	82.178726	86173.4218	
10.55.182.100	63548	173.243.138.201	8888	938	86 k	469	49 k	469	36 k	82.178736	86173.4426	
10.55.182.100	63548	173.243.138.200	8888	939	86 k	470	49 k	469	36 k	82.178840	86173.4375	
10.55.182.100	63546	96.45.33.73	8888	1,078	102 k	546	61 k	532	41 k	82.773213	86164.6093	
10.55.182.100	63546	173.243.138.201	8888	955	88 k	479	51 k	476	37 k	82.773247	86164.6382	
10.55.182.100	63546	173.243.138.200	8888	1,012	94 k	506	55 k	506	39 k	82.773372	86164.6318	
10.55.182.100	60772	172.217.8.202	443	12	7112	6	2563	6	4549	84.339691	15.0852	
10.55.182.100	61272	172.217.8.163	443	6	3090	3	1553	3	1537	648.209168	0.1545	
10.55.182.100	59685	172.217.8.196	443	309	206 k	121	16 k	188	189 k	648.217702	59.5390	
10.55.182.100	59686	172.217.8.195	443	6	3090	3	1553	3	1537	648.226462	0.1467	

☐ Name resolution ☐ Limit to display filter ☐ Absolute start time Conversation Types ▼

Copy Follow Stream... Graph... Close Help

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$
```


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$
```

Zeek's 5 min TCP timeout problem

- ▷ Zeek uses a TCP state timeout of 5 minutes
- ▷ Purges entries that have gone quiet
- ▷ Designed to keep the state table small
- ▷ Can cause bursty long connections to get broken up into smaller segments
- ▷ This breaks long conn detection
- ▷ Cumulative calculations will be off as well
- ▷ Proper fix is to extend TCP timeout

Zeek's 5 min TCP timeout problem

```
ritabeakerlab@ritabeakerlab:~/lab1b$ bro -C -r athlab-1.pcap
ritabeakerlab@ritabeakerlab:~/lab1b$ cat conn.log | bro-cut ts id.orig_h id.orig_p id.resp_h id.res
p_p conn_state duration | grep 52.177.166.224 | head -5
1580950130.985146      10.0.2.15      50543  52.177.166.224  443      S1      0.295939
1580951340.641297      10.0.2.15      50543  52.177.166.224  443      OTH      0.093155
1580953020.637633      10.0.2.15      50543  52.177.166.224  443      OTH      0.091489
1580954646.564385      10.0.2.15      50543  52.177.166.224  443      OTH      0.092423
1580956326.593464      10.0.2.15      50543  52.177.166.224  443      OTH      0.089237
ritabeakerlab@ritabeakerlab:~/lab1b$ rm *.log
ritabeakerlab@ritabeakerlab:~/lab1b$ bro -C -r athlab-1.pcap "tcp_inactivity_timeout = 60 min;"
ritabeakerlab@ritabeakerlab:~/lab1b$ cat conn.log | bro-cut ts id.orig_h id.orig_p id.resp_h id.res
p_p conn_state duration | grep 52.177.166.224 | head -5
1580950130.985146      10.0.2.15      50543  52.177.166.224  443      S1      68236.602521
ritabeakerlab@ritabeakerlab:~/lab1b$
```

How to fix Bro/Zeek's timeout

Add the following to the end of local.bro or local.zeek file:

```
redef tcp_inactivity_timeout = 60 min;
```

Run Bro/Zeek specifying to use these alternate settings:

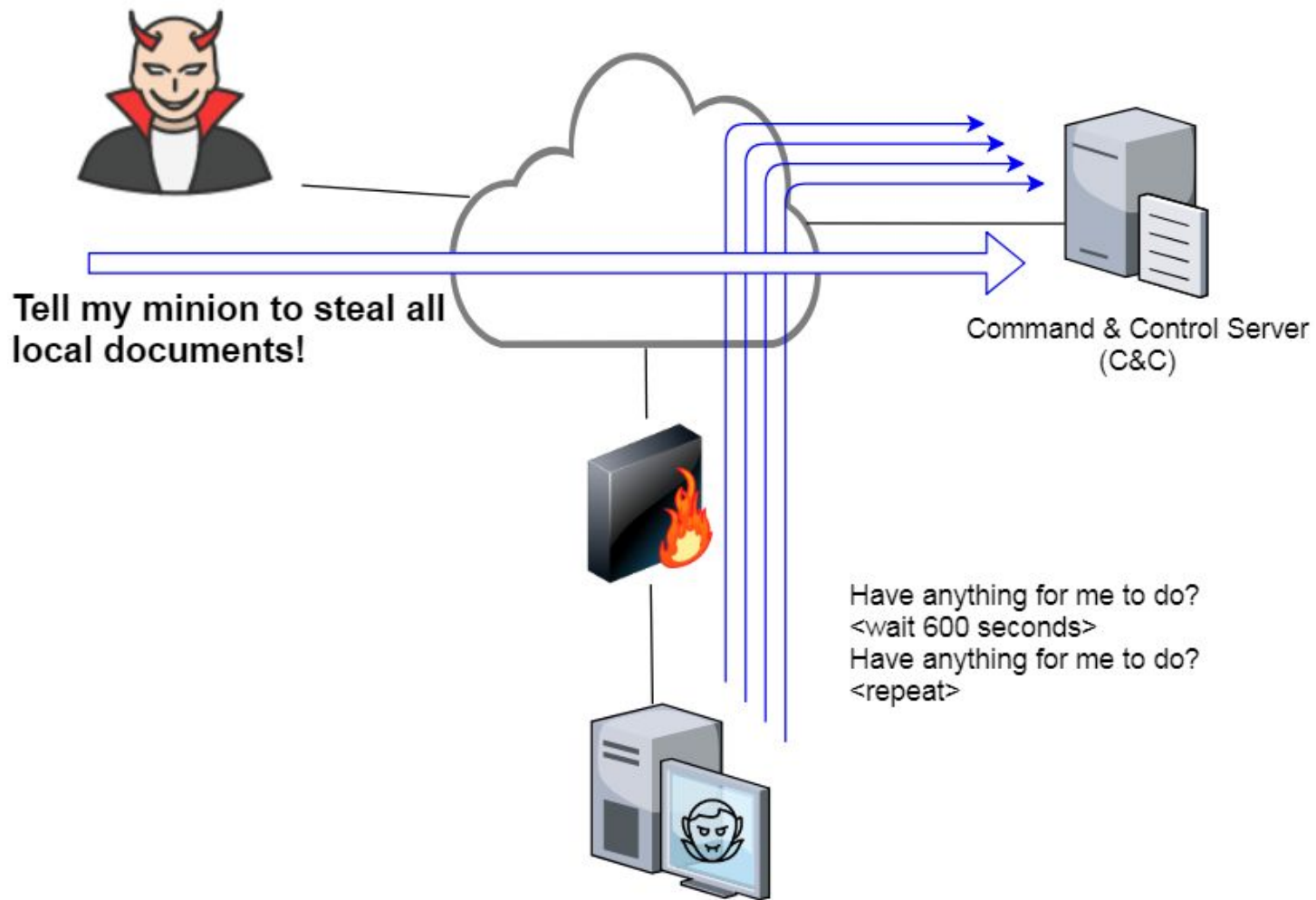
```
zeek -r <pcap file> local
```


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

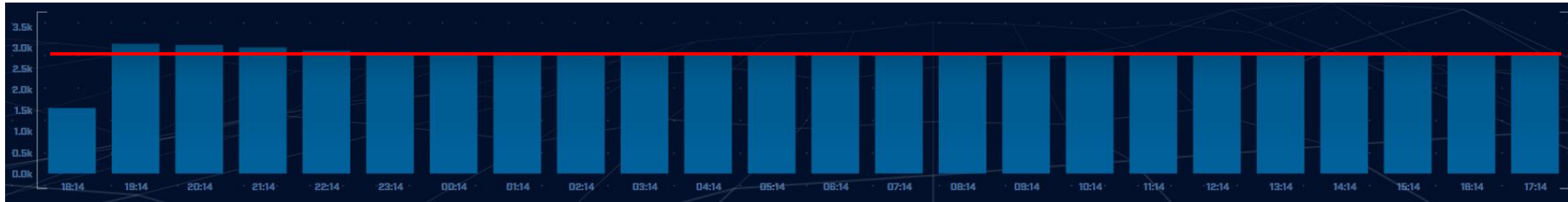
Beacons

- ▷ Same rules - internal to external
- ▷ Can be more challenging than long conns
- ▷ Looking for persistent outbound signal
 - Is there consistency in timing?
 - Is there consistency in session size?



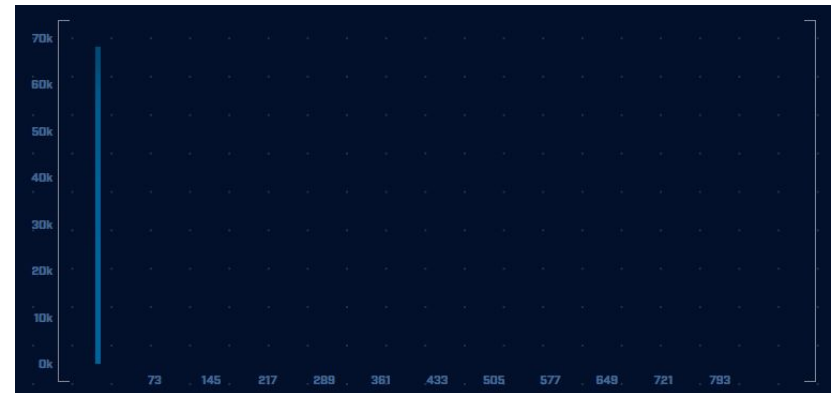
A beacon timing example

Flat line = beacon!



Connections per hour

Dispersion of connection deltas



Beacon session size example



Heartbeat
(check in, nothing to do)

Activation

Tshark timing & size

```
cbrenton@cbrenton-3:~/testing/dnscat$ tshark -r suspect-dns.pcap -T fields -E separator=, -e ip.src -e ip.dst -e ip.proto -e udp.dstport -e ip.len -e frame.time_delta_displayed ip.src==192.168.88.2 | head -10
192.168.88.2,165.227.88.15,17,53,89,0.000000000
192.168.88.2,165.227.88.15,17,53,89,1.074819000
192.168.88.2,165.227.88.15,17,53,89,1.084472000
192.168.88.2,165.227.88.15,17,53,89,1.078729000
192.168.88.2,165.227.88.15,17,53,89,1.069749000
192.168.88.2,165.227.88.15,17,53,89,1.077715000
192.168.88.2,165.227.88.15,17,53,89,1.076643000
192.168.88.2,165.227.88.15,17,53,89,1.070790000
192.168.88.2,165.227.88.15,17,53,89,1.071049000
192.168.88.2,165.227.88.15,17,53,89,1.064914000
tshark: An error occurred while printing packets: Broken pipe.
cbrenton@cbrenton-3:~/testing/dnscat$ _
```

Stats are per packet, not per session!

Beacon jitter

- ▷ The introduction of variance into timing
- ▷ Designed to make signal harder to detect
- ▷ Cobalt Strike:
 - `bsleep($1, 60, 50);`
 - For all beacon IDs (\$1)
 - Sleep 60 seconds
 - Jitter timing +/- 50%
- ▷ This become a challenge statistically
- ▷ Easier to detect in "time buckets"

A jitter example



Working around 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

Potential false positives

- ▷ Long connections
 - BGP
 - Chat programs
 - Dynamic web pages (weather)
- ▷ Beacons
 - NTP
 - Internal DNS forwarder to external DNS resolvers
 - Chat programs
 - Security software
- ▷ Whitelist to remove from future hunts

What next?

- ▷ Investigate all persistent connections
- ▷ Time to perform a deeper analysis on each
- ▷ Look at:
 - Protocol usage
 - Source IP
 - Destination IP
- ▷ Will cover this in the next section



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>>

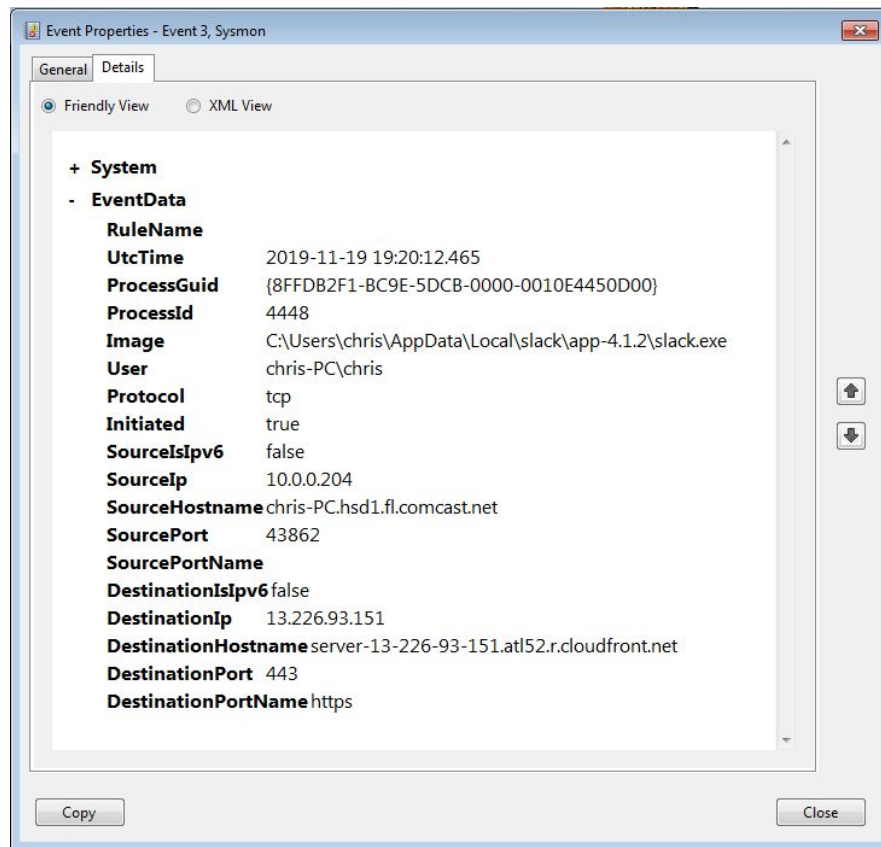
Internal system

- ▷ 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

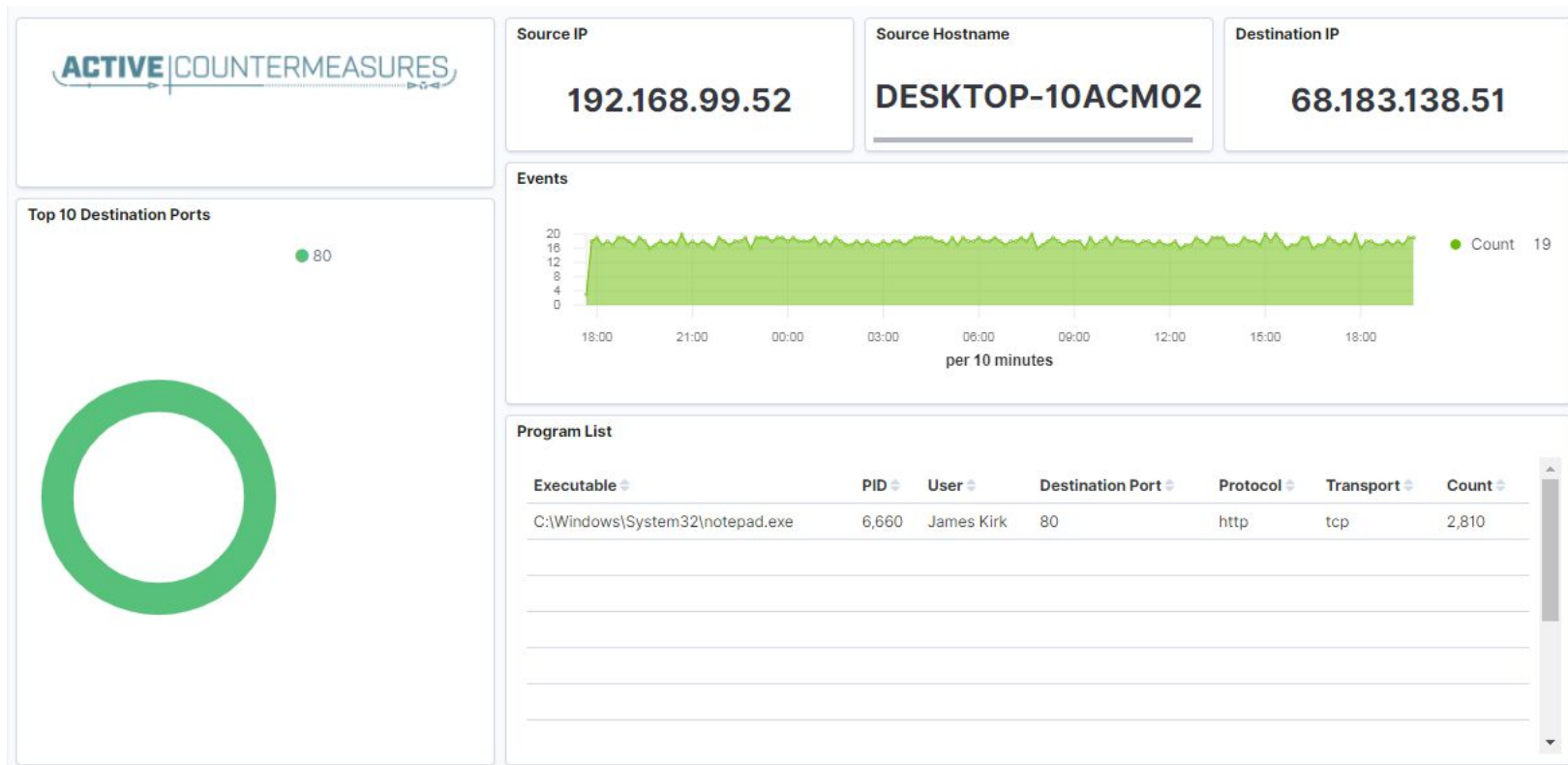
- ▷ Network shows suspicious traffic patterns
- ▷ Use this data to pivot to host logs
- ▷ Filter your logs based on:
 - Suspect internal host
 - Timeframe being analyzed
- ▷ Anything stand out as unique or odd?

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

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
```

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/>

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

Useful when I have a target

The image shows a Wireshark packet capture analysis of a file named `perimeter_class.cap`. The top pane displays a list of network packets. The bottom pane shows the detailed view of a selected packet (Frame 98594), which is an Ethernet II frame containing an Internet Protocol Version 4 packet and a Transmission Control Protocol (TCP) segment.

Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
98594	678.865000	148.78.247.10	12.33.247.4	TCP	78	78 80 → 26268 [SYN, Seq=0 Win=0 Len=0]
98595	678.865219	12.33.247.4	148.78.247.10	TCP	78	78 80 → 26268 [SYN, ACK] Seq=0 Ack=1566666666 Win=0 Len=0
98597	678.894523	148.78.247.10	12.33.247.4	TCP	70	70 26268 → 80 [ACK] Seq=1 Ack=1566666666 Win=0 Len=0
98599	678.896451	148.78.247.10	12.33.247.4	HTTP	225	HEAD / HTTP/1.0 [ETHERNET FRAME]
98600	678.896515	12.33.247.4	148.78.247.10	TCP	70	70 80 → 26268 [ACK] Seq=1 Ack=1566666666 Win=0 Len=0
98601	678.899778	12.33.247.4	148.78.247.10	HTTP	211	HTTP/1.1 200 OK [ETHERNET FRAME]
98602	678.899881	12.33.247.4	148.78.247.10	TCP	70	70 80 → 26268 [FIN, ACK] Seq=142 Ack=1566666666 Win=0 Len=0
98608	678.929234	148.78.247.10	12.33.247.4	TCP	70	[TCP Dup ACK 98597#1] 26268 → 80 [ACK] Seq=1566666666 Win=0 Len=0
98609	678.933213	148.78.247.10	12.33.247.4	TCP	70	70 26268 → 80 [ACK] Seq=1566666666 Ack=1566666666 Win=0 Len=0
98610	678.933475	148.78.247.10	12.33.247.4	TCP	70	70 26268 → 80 [FIN, ACK] Seq=1566666666 Ack=1566666666 Win=0 Len=0
98611	678.933517	12.33.247.4	148.78.247.10	TCP	70	70 80 → 26268 [ACK] Seq=143 Ack=1566666666 Win=0 Len=0
98716	679.708532	148.78.247.10	12.33.247.4	TCP	78	78 26460 → 80 [SYN] Seq=0 Win=65535 Len=0

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)

Packet Bytes:

```
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  ....PD...addr
0040 08 0a 00 ec 48 44 00 00 00 00 61 64 64 72
```

perimeter_class.cap | Packets: 197147 · Displayed: 5462 (2.8%) | 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
```

R statistic tools

- ▷ What is it good for?
 - Generating statistics from a set of numbers
 - Producing repeatable results
- ▷ When to use it
 - Beacon detection
- ▷ Where to get it

<https://www.r-project.org/>
`sudo apt install r-base`

R example

```
cbrenton@cbrenton-3:~/testing/dnscat$  
cut -d ',' -f 5 beacon-test.txt | Rscr  
ipt -e 'y <-scan("stdin", quiet=TRUE)'  
-e 'cat(min(y), max(y), mean(y), sd(y)  
, sep="\n")'  
89  
290  
89.83496  
12.75772  
cbrenton@cbrenton-3:~/testing/dnscat$
```

Min sessions size

Max sessions size

Mean very close to min could
indicate a heartbeat

Standard deviation is small and
close in value to "mean minus
min". Indicator this could be a
heartbeat

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
```

Majestic Top 1 Million

- ▷ List of busiest sites on the Internet
- ▷ Busiest sites less likely to be evil
 - But not a guarantee
 - Absolutely a minor modifier

http://downloads.majesticseo.com/majestic_million.csv

Open source threat feeds

- ▷ What's it good for?
 - Identifying reputation of external IPs
 - Can assist with attribution
 - Need to qualify any matches
 - How old is the entry?
 - Why was it flagged?
 - Is it within dynamic space?
- ▷ When to use it
 - Minor modifier only

Threat feeds

Spamhaus & DShield

<http://rules.emergingthreats.net/fwrules/emerging-Block-IPs.txt>

Known C2 servers

<http://osint.bambenekconsulting.com/feeds/c2-ipmasterlist.txt>

Consolidation of multiple feeds

<https://raw.githubusercontent.com/stamparm/ipsum/master/ipsum.txt>

ISC top reported sites

https://isc.sans.edu/feeds/suspiciousdomains_High.txt

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
```



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
- ▷ Bro-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

Identify time window being audited

```
thunt@thunt:~/lab1$ capinfos -aeu thunt-lab.pcapng
File name:          thunt-lab.pcapng
Capture duration:    86398.290132521 seco
First packet time:   2018-01-30 18:14:02.059662093
Last packet time:    2018-01-31 18:14:00.349794614
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$  
thunt@thunt:~/lab1$ cat conn.log | zeek-cut id.orig_h id.resp_h duration | sort  
-k 3 -rn | head  
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      111.221.29.114      46638.510373  
10.55.100.108      65.52.108.220      44615.165823  
10.55.100.106      40.77.229.91        41206.913035  
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  
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:~/lab1$ _
```

Well this is new...

Run down the longest talkers

- ▷ Let's investigate the external IP on the three longest sessions
 - 65.52.108.225
 - 111.221.29.113
 - 40.77.229.82
- ▷ We'll use two common web tools
 - ThreatCrowd
 - <https://www.threatcrowd.org/>
 - AbuseIPDB
 - <https://www.abuseipdb.com/>

Useful commands

- ▷ Both sites let you send a URL with the IP address at the end
- ▷ Useful for automation

<https://www.threatcrowd.org/ip.php?ip=65.52.108.225>

<https://www.threatcrowd.org/ip.php?ip=111.221.29.113>

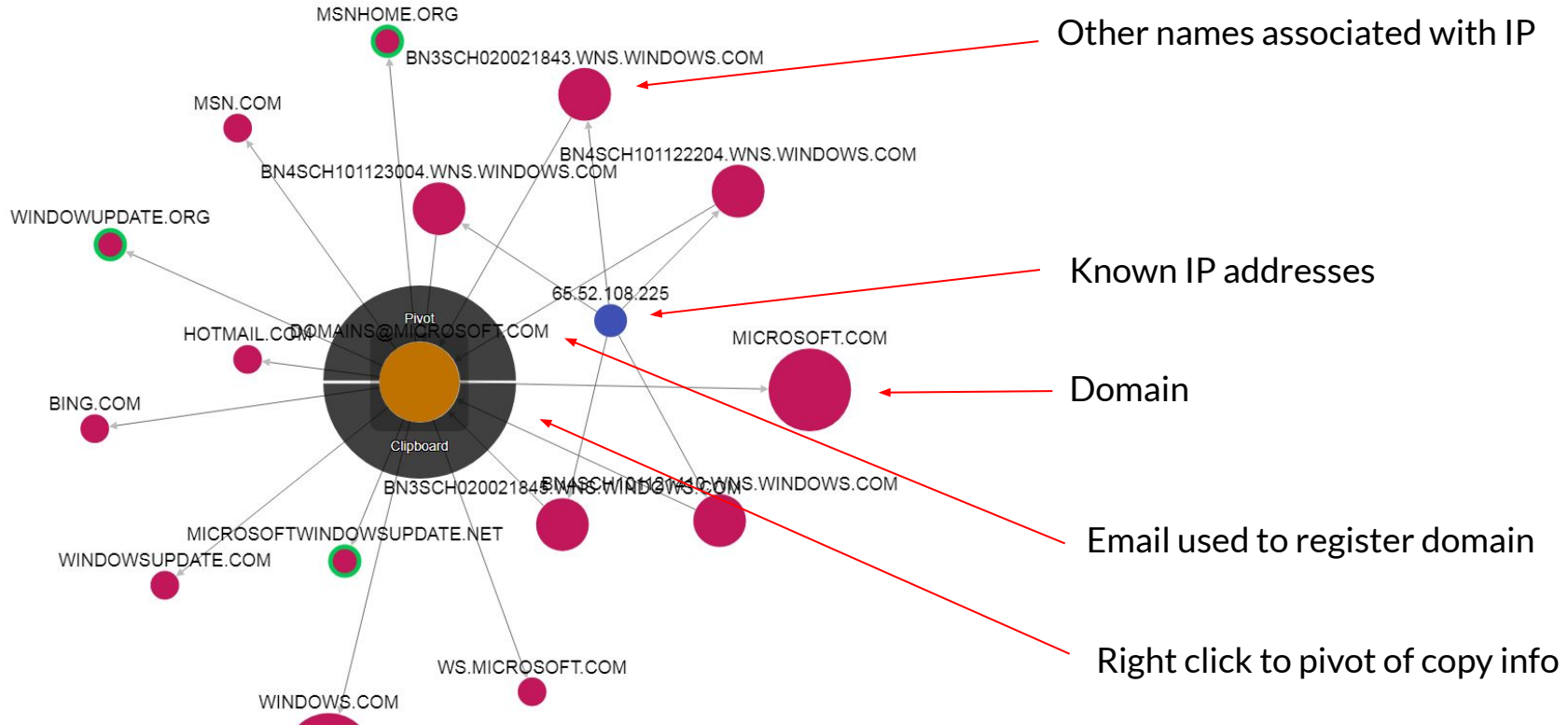
<https://www.threatcrowd.org/ip.php?ip=40.77.229.82>

<https://www.abuseipdb.com/check/65.52.108.225>

<https://www.abuseipdb.com/check/111.221.29.113>

<https://www.abuseipdb.com/check/40.77.229.82>

Research - Threat Crowd Results



Research - Threat Crowd FQDN info

REVERSE DNS

Domain	Date
bn3sch020021845.wns.windows.com	2020-04-28
bn4sch101123004.wns.windows.com	2020-04-23
bn4sch101122204.wns.windows.com	2020-04-01
bn3sch020021843.wns.windows.com	2020-03-14
bn4sch101121410.wns.windows.com	2020-03-14

DNS RESOLUTIONS

SSL CERTIFICATE

SSL MD5	83f4ff001d33c9efc0f967ca2163b319
SSL SHA1	2ac66742b18c305c1924511049dea272b7a5dd91

Subject: commonName=*.wns.windows.com
Issuer: commonName=Microsoft IT SSL SHA2/organizationName=Microsoft Corporation/stateOrProvinceName=Washington/countryName=US
Not valid before: 2014-10-23 17:55

Known FQDNs

Digital certificate info
(when applicable)

Research - AbuseIPDB

65.52.108.225 was found in our database!

This IP was reported **2** times. Confidence of Abuse is **0%** ?

0%

ISP	Microsoft Corporation
Usage Type	Search Engine Spider
Domain Name	microsoft.com
Country	United States
City	Boydton, Virginia

Spot an error? IP info including ISP, Usage Type, and Location provided by [IP2Location](#).

[REPORT 65.52.108.225](#) [WHOIS 65.52.108.225](#)

MongoDB Atlas is the most reliable cloud database service available. Try now!

ADS VIA CARBON

SPONSOR

Microsoft Azure Get 750 hours of virtual machine compute time, free each month for a year.

IP Abuse Reports for 65.52.108.225:

This IP address has been reported a total of **2** times from 2 distinct sources. 65.52.108.225 was first reported on July 17th 2017, and the most recent report was **2 years ago**.

Old Reports: The most recent abuse report for this IP address is from **2 years ago**. It is possible that this IP is no longer involved in abusive activities.

Reporter	Date	Comment	Categories
Anonymous	15 Jan 2018		Phishing Hacking Brute-Force Exploited Host
Anonymous	17 Jul 2017	[DoS Attack: ACK Scan] from source: 65.52.108.225, port 443, Sunday, July 16,2017 08:52:04	DDoS Attack

Research - Answers - Final

- ▷ Windows push notification service
 - https://en.wikipedia.org/wiki/Windows_Push_Notification_Service
- ▷ Not malware (in pure sense of the term)
- ▷ Is there a business need for this?
 - If yes, hunt it down and kill it
 - If no, 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
- ▷ Run down the top three by connection quantity
- ▷ Is there consistency in session size?
 - Possible beacon?

Find beacons - hints

- ▷ You need to be able to clearly identify:
 - When a new session starts
 - 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 192.168.88.2 | grep 165.227.88.15  
| sort | uniq -c | sort -rn | head
```

```
cat conn*.log | zeek-cut id.orig_h id.resp_h  
orig_bytes | grep 192.168.88.2 | grep 165.227.88.15  
| sort | grep -v -e '^$' | grep -v '-' | datamash  
-g 1,2 count 3 min 3 mean 3 mode 3 max 3
```


Answers - most connections

```
thunt@thunt:~/lab1$ cat conn*.log | zeek-cut id.orig_h id.resp_h | sort | uniq
-c | sort -rn | head
108858 192.168.88.2      165.227.88.15
 64285 10.55.200.10         172.16.200.11
 20054 10.55.100.111        165.227.216.194
  4190 10.55.182.100        10.233.233.5
  3856 10.55.200.10         216.239.34.10
  3660 10.55.200.11         193.108.88.128
  2742 10.55.200.11         88.221.81.192
  2289 10.55.200.11         205.251.195.166
  2265 10.55.200.11         216.239.34.10
  1931 10.55.200.10         205.251.195.166
thunt@thunt:~/lab1$ _
```

Evaluate top 3 as possible beacons

Others could be beacons as well but need a cut off

Session size analysis

```
thunt@thunt:~/lab1$ cat conn*.log | zeek-cut id.orig_h id.resp_h orig_bytes | g  
rep 192.168.88.2 | grep 165.227.88.15 | sort | uniq -c | sort -rn | head  
108250 192.168.88.2 165.227.88.15 61  
429 192.168.88.2 165.227.88.15 262  
117 192.168.88.2 165.227.88.15 -  
13 192.168.88.2 165.227.88.15 122  
12 192.168.88.2 165.227.88.15 81  
3 192.168.88.2 165.227.88.15 260  
3 192.168.88.2 165.227.88.15 197  
3 192.168.88.2 165.227.88.15 118  
2 192.168.88.2 165.227.88.15 93  
2 192.168.88.2 165.227.88.15 73  
thunt@thunt:~/lab1$ _
```

Possible beacon with activation
Repeat for other two IP pairs

Beacons - answers

```
thunt@thunt:~/lab1$ cat conn*.log | zeek-cut id.orig_h id.resp_h orig_bytes | g  
rep 192.168.88.2 | grep 165.227.88.15 | sort | grep -v -e '^$' | grep -v '-' |  
datamash -g 1,2 count 3 min 3 mean 3 mode 3 max 3  
165.227.88.15    192.168.88.2    1          3673      3673      3673      3673  
192.168.88.2    165.227.88.15    108741     0          61.832602238346 61          262  
thunt@thunt:~/lab1$ _
```

Mean and mode are close, so potential beacon...
but we knew that by eyeballing it on the last slide

Look for C2 over DNS

- ▷ Continue with the same data files
- ▷ 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

- ▶ Can solve with both tshark and Zeek
- ▶ Tshark
 - '-e dns.query.name'
 - Note: Processor and memory intensive!
- ▶ Zeek
 - Use dns.log
 - 'query'
- ▶ Zeek option runs faster and uses fewer system resources

Useful commands to try

```
tshark -r thunt-lab.pcapng -T fields -e  
dns.qry.name | sort | uniq | rev | cut -d '.'  
-f 1-2 | rev | sort | uniq -c | sort -rn | head
```

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

Answers - C2 over DNS - tshark

```
thunt@thunt-one-day:~/lab1$ tshark -r thunt-lab.pcapng -T fields -e dns.qry.  
name | sort | uniq | rev | cut -d . -f 1-2 | rev | sort | uniq -c | sort -rn  
| head -10  
48812 r-1x.com  
146 akamaiedge.net  
118 akadns.net  
114 edgekey.net  
91 amazonaws.com  
60 microsoft.com  
41 cloudfront.net  
37 dynect.net  
33 akamai.net  
32 akam.net  
thunt@thunt-one-day:~/lab1$
```

Answers - C2 over DNS - Zeek

```
thunt@thunt:~/lab1$ cat dns*.log | zeek-cut query | sort | uniq | rev | cut -d  
. -f 1-2 | rev | sort | uniq -c | sort -rn | head  
62468 r-1x.com  
154 akamaiedge.net  
125 akadns.net  
121 edgekey.net  
101 amazonaws.com  
67 microsoft.com  
51 dynect.net  
45 parsely.com  
44 akam.net  
43 cloudfront.net  
thunt@thunt:~/lab1$
```

Note: Nearly identical to tshark equivalent

Repeat the labs with RITA

- ▷ With RITA, identify:
 - Long connections
 - Beacons
 - C2 over DNS
- ▷ Data is already loaded into RITA
 - Database = lab1
- ▷ 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

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 - RITA long connections

```
thunt@thunt-one-day:~/lab1$ rita show-long-connections lab1 | head
Source IP, Destination IP, Port: Protocol: Service, Duration
10.55.100.100, 65.52.108.225, 443: tcp: -, 86222.4
10.55.100.107, 111.221.29.113, 443: tcp: -, 86220.1
10.55.100.110, 40.77.229.82, 443: tcp: -, 86160.1
10.55.100.109, 65.52.108.233, 443: tcp: ssl, 72176.1
10.55.100.105, 65.52.108.195, 443: tcp: ssl, 66599
10.55.100.103, 131.253.34.243, 443: tcp: -, 64698.4
10.55.100.104, 131.253.34.246, 443: tcp: ssl, 57413.3
10.55.100.111, 111.221.29.114, 443: tcp: -, 46638.5
10.55.100.108, 65.52.108.220, 443: tcp: -, 44615.2
thunt@thunt-one-day:~/lab1$ _
```

Answers - Cumulative talk time

```
thunt@thunt:~/lab1$ 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  
datamash: invalid numeric value in line 1001 field 3: 'Duration'  
10.55.100.100,65.52.108.225,86222.4  
10.55.100.107,111.221.29.113,86220.1  
10.55.100.110,40.77.229.82,86160.1  
10.55.100.109,65.52.108.233,72176.1  
10.55.100.105,65.52.108.195,66599  
10.55.100.103,131.253.34.243,64698.4  
10.55.100.104,131.253.34.246,57413.3  
10.55.100.111,111.221.29.114,46638.5  
10.55.100.108,65.52.108.220,44615.2  
10.55.100.106,40.77.229.91,41206.9  
thunt@thunt:~/lab1$ _
```

Results are identical. Will not always be the case.

Answers - Beacons

```
thunt@thunt-one-day:~/lab1$ rita show-beacons lab1 | head
Score,Source IP,Destination IP,Connections,Avg Bytes,Intvl Range,Size Range,
Top Intvl,Top Size,Top Intvl Count,Top Size Count,Intvl Skew,Size Skew,Intvl
Dispersion,Size Dispersion
1,192.168.88.2,165.227.88.15,108858,199,860,230,1,89,53341,108319,0,0,0,0
1,10.55.100.111,165.227.216.194,20054,92,29,52,1,52,7774,20053,0,0,0,0
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
0.834,192.168.88.2,13.107.5.2,27,198,2,33,12601,73,4,15,0,0,0,0
0.834,10.55.100.111,34.239.169.214,34,704,5,4517,1,156,15,30,0,0,0,0
0.833,10.55.100.106,23.52.161.212,27,940,38031,52,1800,505,19,19,0,0,0,0
0.833,10.55.100.111,23.52.162.184,27,2246,37828,52,1800,467,23,25,0,0,0,0
0.833,10.55.100.100,23.52.161.212,26,797,36042,52,1800,505,16,25,0,0,0,0
thunt@thunt-one-day:~/lab1$
```

Score scale = 0 → 1 with "1" being a perfect beacon score

Score based on session timing and size

Answers - C2 over DNS

```
thunt@thunt-one-day:~/lab1$ rita show-exploded-dns lab1 | head
Domain,Unique Subdomains,Times Looked Up
r-1x.com,62468,109227
dnsc.r-1x.com,62466,108911
akamaiedge.net,154,27381
akadns.net,125,13907
edgekey.net,121,7110
amazonaws.com,101,13297
elb.amazonaws.com,90,13259
com.edgekey.net,88,6075
microsoft.com,67,1687
thunt@thunt-one-day:~/lab1$
```


Answers - Investigate top results

```
thunt@thunt-one-day:~/lab1$ rita show-exploded-dns lab1 | grep 'r-1x.com' | head
```

```
r-1x.com,62468,109227  
dnsc.r-1x.com,62466,108911  
20202020202020202020202020202020.dnsc.r-1x.com,12,12  
2e202e202e202e202e203a20666538303a3a.dnsc.r-1x.com,8,8  
0a2020205375626e6574204d61736b202e202e202e202e202e202e202e202e202e20.2e202e202e202e20  
03a203235352e3235352e.dnsc.r-1x.com,8,8  
2e202e202e202e203a203235352e3235352e.dnsc.r-1x.com,8,8  
6f73742e65786520202020202020202020202020.dnsc.r-1x.com,7,7  
20536572766963657320202020202020202020202020.dnsc.r-1x.com,7,7  
6e736f6c65202020202020202020202020202020202020.dnsc.r-1x.com,6,6  
6365732020202020202020202020202020202020202020202020.dnsc.r-1x.com,6,6  
thunt@thunt-one-day:~/lab1$
```


Next steps

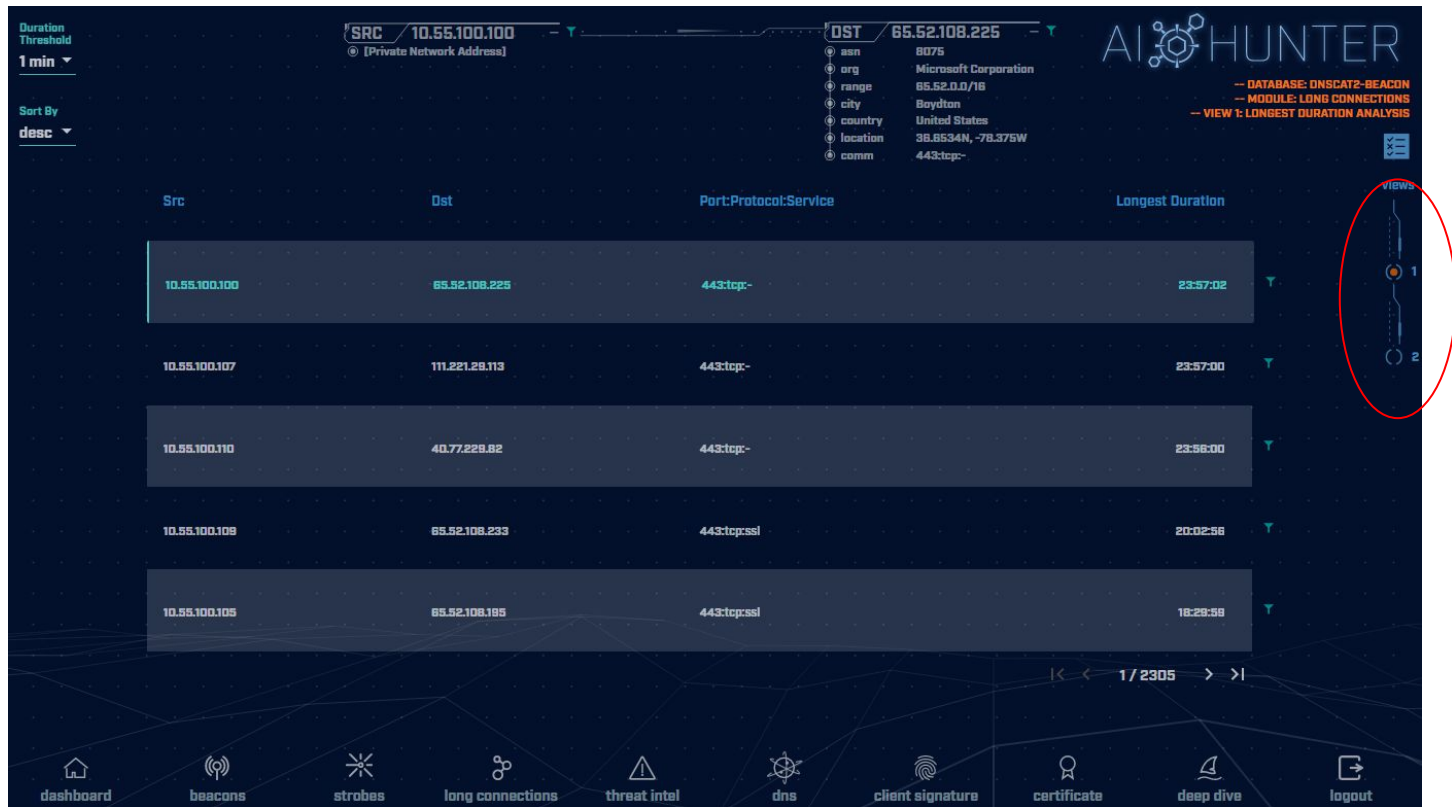
- ▷ Do we feel confident in flagging anything we have seen as requiring incident handling?
- ▷ Are there any connections that need more research?
 - What should this research be?
 - Do we need to involve any other teams?
 - If we need more data collection, for how long?

Quick demo

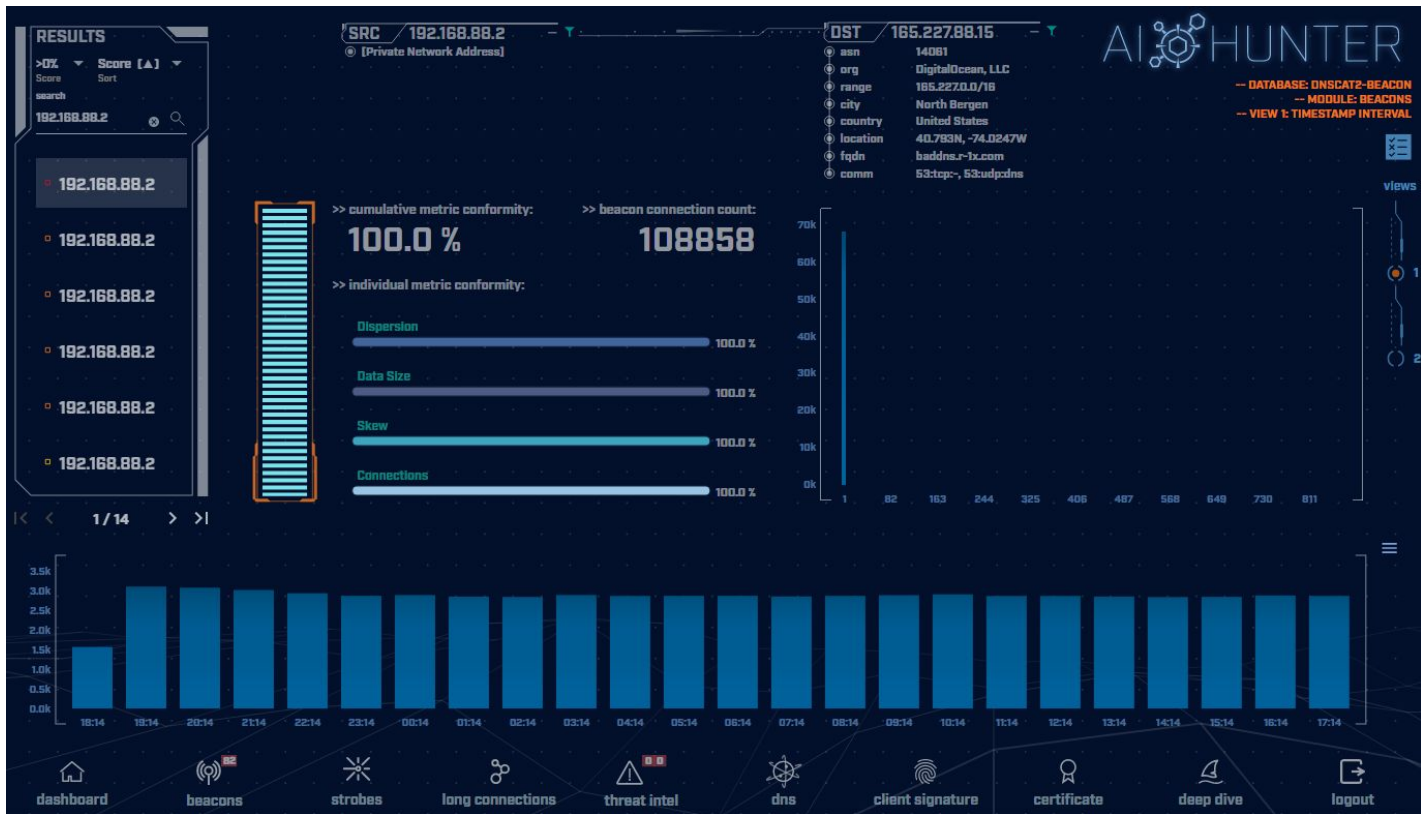
- ▷ Same data, seen through AI-Hunter
- ▷ Inexpensive commercial solution
- ▷ Automates much of the hunting process



12 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, displaying the IP address '165.227.88.15'. Below this, a list of attributes and their corresponding values is shown:

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 displayed 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 a scale from 60k to 70k. The bar is blue and reaches approximately 65k.

Subdomain Threshold 0

AI HUNTER

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

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

DNS Queries [1]

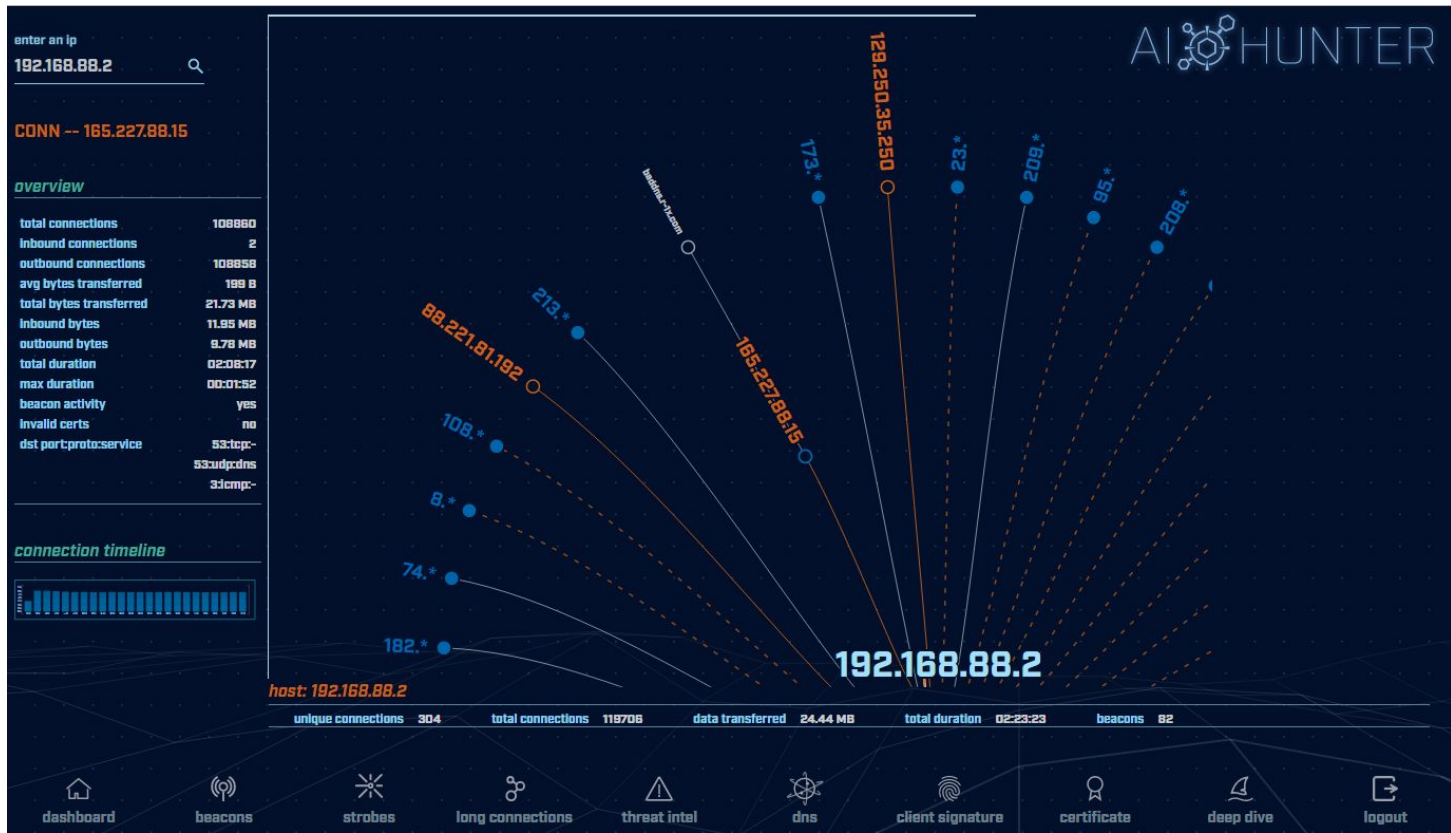
Direct Connections [1]

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

Wrap Up

- ▷ Thanks for attending!
- ▷ Very special thank you to the folks behind the scenes
- ▷ Content feedback?
 - Please email: chris@activecountermeasures.com