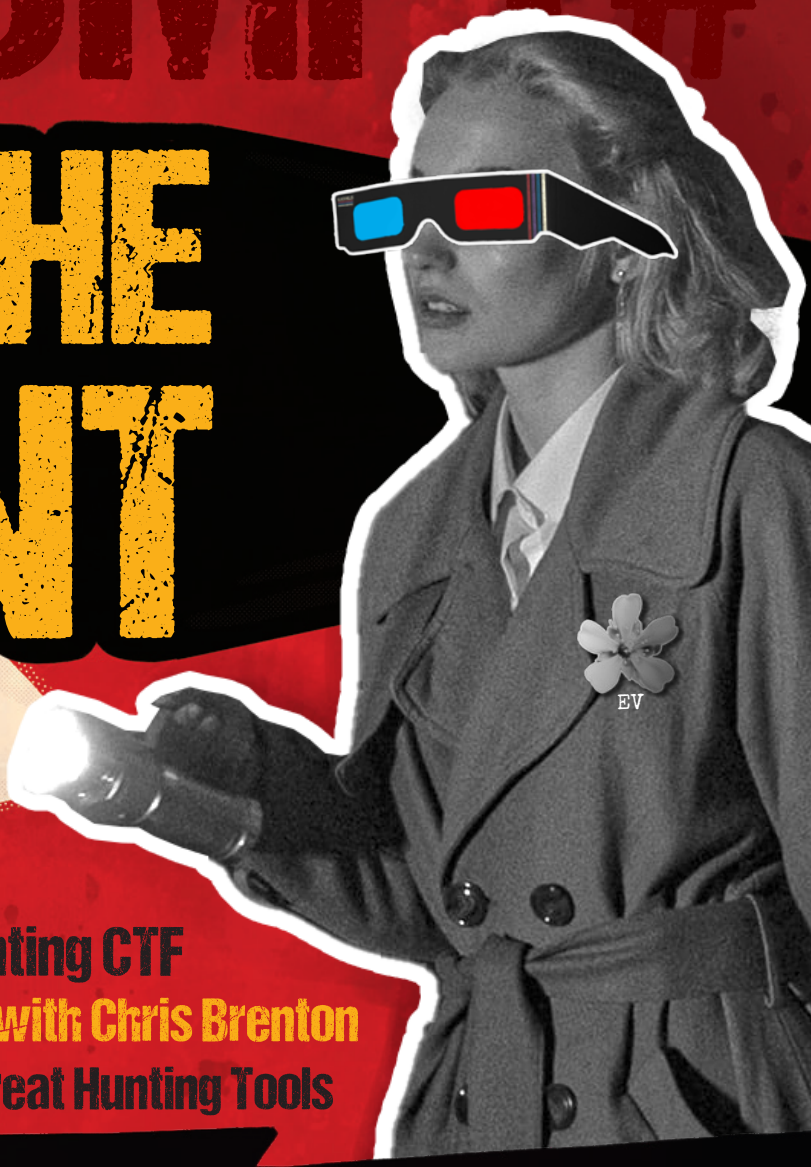
PROMPT#

ON THE HUNT



- Online Threat Hunting CTF
- Threat Hunter Q&A with Chris Brenton
- Tips & Tricks for Threat Hunting Tools

---FREE---

AC Hunter
Community Edition
(and how to use it)

Pull-Out Poster

TShark Cheat Sheet

- THT Commands Cheat Sheet
- Other Cool Stuff

Solve the puzzle inside and get
a free enamel pin!

BY BRADLEY SAMUELSON

Attackers Taunt You

The attackers call out the security team, via a message to every system on the network.

NOTES

The attackers use msg command to send a pop-up to incident responders, "Shall we play a game?"

MU HA HA HA HA HAAA!

Contributed by @davehull

EXPV1_1_0821

If you check Google Trends for 2017,

"threat hunting"

was barely a thing.

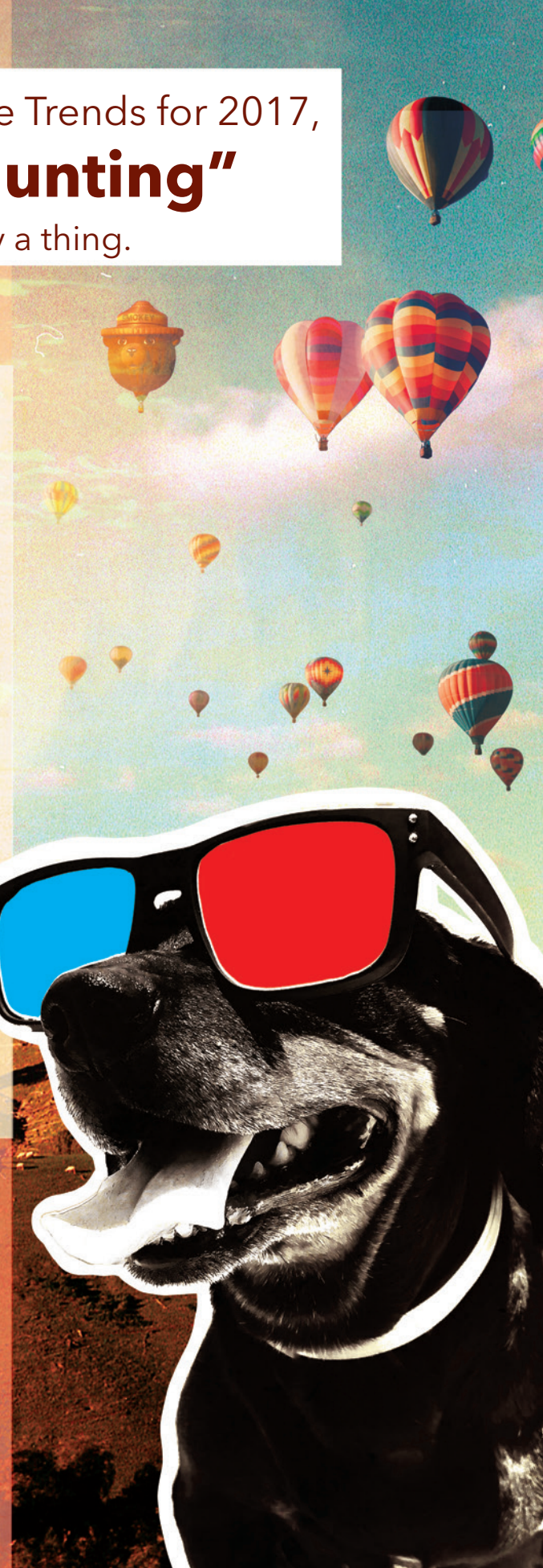
The little bit of information available on the topic focused on EDR and signature matching on centralized logs. When I went through the current network breach data, it was clear that these processes were *not working*. The chances of catching an adversary that successfully made it past your layers of protection was minimal at best.

So that year, when John Strand came to me and said, **"I think we have a pretty cool tool for finding adversaries on the network"** and ran me through a demo of a very early alpha version of AC-Hunter, to say I was blown away would be an understatement. In fact, I was captivated to the point that I don't think I slept for the next 36 hours.

I feel like we've finally turned a corner as threat hunting is starting to go mainstream. We've had over 30,000 people attend our threat hunting training class. The threat hunting Discord server now has over 23,000 members with engaging and frequent conversations. NIST special publication 800-53 now calls out threat hunting as a requirement.

In short, the future of threat hunting is *so bright* **we've included a set of shades in this issue.**

Best,
Chris Brenton



?s FROM A BEGINNER

THREAT HUNTER

ANSWERED BY CHRIS BRENTON

QUESTIONS COMPILED FROM THE INFOSEC COMMUNITY BY SHELBY PERRY

What is the primary difference between threat *detection* and threat *hunting*?

Threat *detection* is the act of detecting malicious activity on the network. This may occur through a number of different means, like an alert being triggered or during a forensic analysis. Threat detection is a very generic term and the process can be passive or active.

Threat *hunting* is specific to actively searching through network and host data, looking for indicators of compromise. This activity is performed regardless of whether any alerts have been triggered.

What's required to start threat hunting?

Process-wise: The first step is to **identify what checks you wish to perform and what data is needed to perform that check.** For example, if you want to hunt for C2 communications, you need a way to analyze all traffic passing between the internal network and the internet. This is usually accomplished by capturing traffic at the internal interface of the firewall. This may be done with a network tap or by leveraging a switch SPAN port. Once the data is collected, you need tools and processes to distinguish between C2 communications and normal traffic patterns. C2 can be pretty stealthy, so you may need the ability to analyze the traffic in 12-hour(+) "chunks" of time in order to be able to distinguish it from normal patterns.

Knowledge-wise: For network threat hunting, it's extremely helpful to **have a good knowledge of networking and protocol communications.** For example, HTTPS communications typically use the SSL/TLS protocols over TCP port 443. Many C2 tools pass their traffic over TCP/443, but simply obfuscate it (they don't use SSL/TLS). So if you are network savvy and see traffic using TCP/443 that does not include the SSL/TLS handshake, you know that's something that needs to be investigated further.

If you plan to do your hunts on the endpoints, you need to have a strong knowledge of every operating system and the applications they are using. For example, PowerShell is a powerful scripting language built into the Windows operating system. It is rare that anyone outside of the IT or security teams would have a legitimate reason for using it. So as a threat hunter, you would need to know that *Nancy-in-accounting running PowerShell is extremely suspicious behavior.*

What does C2 over DNS mean?

C2 over DNS is the practice of an attacker embedding the C2 traffic inside legitimate DNS queries. This causes your DNS forwarders to happily send the C2 traffic out to the internet. The attackers will then register a remote domain and set up their C2 servers as the authoritative DNS servers for the domain. This means that your DNS servers will send the C2 traffic to the remote C2 server. What makes this C2 activity so stealthy is that the compromised system does not generate any new traffic headed to the internet. Instead, you simply see an increase in the number of DNS queries.

There are so many tools out there... How do I know which to use for what?

Try them out! See which works best in your environment and matches your workflow. Also, don't expect one tool to always be a perfect fit for every need. For example, if I'm analyzing overall traffic flow, I'll use a tool like Zeek. If I'm looking for specific patterns, I'll use Suricata. For specific traffic flows, I'll use TShark. For a deep analysis on a single session, I'll turn to Wireshark. My best advice: pick one tool and stick with it. When it doesn't help with a specific challenge, check out other tools.

I suspect a system has been compromised, what is the best/easiest way to detect lateral movement?

Identify the command and control (C2) server being used and see if any other internal systems are communicating with that server. Be patient, as it's not uncommon for secondary systems to only call home every 4-8 hours. A packet capture running for 24 hours should be sufficient in most cases.

Acronyms To Remember?

C2 – Command and Control
DNS – Domain Name System
FQDN – Fully Qualified Domain Name
SIEM – Security Information and Event Management

ADVANCED THREAT HUNTING

WITH CHRIS BRENTON

AN ANTISYPHON CLASS

ACTIVE COUNTERMEASURES

is a group of like-minded geeks that are passionate about giving back to the security community. We do this through free training, thought leadership, and both open-source and affordable commercial tools.

THE FIRST FEW MINUTES
OF A THREAT INVESTIGATION OR INCIDENT RESPONSE
ARE THE TOUGHEST

You know something appears wrong, but you don't have the details to prove it. The following sites take a piece of information you have (like the remote IP address to which one of your systems is talking) and give back more detail on what it is and whether it's benign or malicious.

THESE SITES HELP MAKE THOSE MINUTES EASIER

For a more comprehensive list, visit
acm.re/threat-hunting-resources/

VISIT US FOR EVEN MORE RESOURCES
ACM.RE

PORT LOOKUPS
speedguide.net/ports.php
Google search for "tcp port portnumber"
or "udp port portnumber"

DOMAINS
ipvoid.com/domain-reputation-check/

IP ADDRESSES
ipinfo.io

WHAT IS MY EXTERNAL IP ADDRESS?
(Handy if you're on a LAN that shares a single IP address!)
icanhazip.com

RESERVED IP ADDRESS BLOCKS
en.wikipedia.org/wiki/Reserved_IP_addresses

HOSTNAMES
virustotal.com/gui/home/search

Gathered by Bill Stearns

ASNs OR AUTONOMOUS SYSTEM NUMBERS
(which are blocks of addresses owned by an organization)
team-cymru.com/ip-asn-mapping

POSSIBLE MALWARE
(files and URLs)
virustotal.com/gui/home/upload

USER AGENT STRINGS
whatismyip.net/tools/user-agent-lookup.php

JA3 HASHES
sslbl.abuse.ch/ja3-fingerprints/

TOR SERVERS
dan.me.uk/tornodes

RITA

-Real Intelligence Threat Analytics-

Written by Liza Tsibur

What is RITA?

RITA is an open-source network threat hunting tool designed to identify malicious command and control (C2) activity. It ingests Zeek connection logs and uses behavioral analytics to identify potentially compromised systems.

Why do we need threat hunting?

Perimeter defensive tools like firewalls and IDS protect your valuable systems. A solid incident response plan allows your team to contain and eliminate a discovered threat. But how do we identify when your protective layers have failed and incident response is needed? What happens when an attacker is already inside? It is clear from the evidence that preventative and reactive measures are woefully insufficient. The average time to identify and contain a data breach has remained virtually unchanged for the past seven years — a shocking figure considering the average cost. And needless to say that while the financial consequences are significant, the final impact surpasses it. Loss of personal information has impacted millions of people, and rising breaches in the healthcare and critical infrastructure industries have placed vital services at risk of failure.

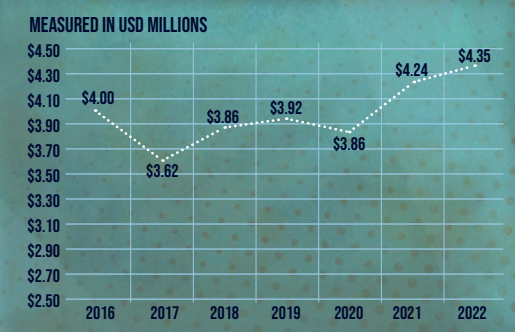
Why RITA?

The one absolute constant we, as threat hunters, can rely on is that **malware must call home.** By analyzing network traffic, we can detect C2 calling home regardless of platform and without the need of endpoint agents. Unfortunately, scouring through log data by hand and reading individual connection entries is unlikely to alert you to potential C2 activity. For better fidelity, behavior and pattern analysis needs to look at the communication between an internal and external host over time. Persistency is the key attribute we look for when searching for compromised systems. RITA looks for the primary indicators of this persistency, allowing you to focus on vetting the activities flagged by its findings.

AVERAGE DAYS TO IDENTIFY AND CONTAIN A DATA BREACH

2022 COST OF A DATA BREACH REPORT BY IBM AND THE PONEMON INSTITUTE

AVERAGE TOTAL COST OF A DATA BREACH



2022 COST OF A DATA BREACH REPORT BY IBM AND THE PONEMON INSTITUTE

Assumption of compromise
Aggregation of network traffic logs
Analysis of behaviors and patterns

RITA's analysis stage splits these primary indicators of persistency into separate modules, allowing results to be viewed based on a specific indicator. Let's look at some of the main modules that make up this analysis.

Beacons

Beacons are repeating "heartbeat" communications between a pair of hosts. While some beacons are innocuous, a compromised system will use beaconing to continuously request instructions or exfiltrate data, allowing an attacker to maintain a persistent presence on the network. RITA identifies and scores four different types of beacons. We recommend investigating hosts with scores over 85% to verify that the associated network traffic fulfills a legitimate business need.

IP

An IP beacon may indicate that a compromised internal system is communicating with a C2 server at a specific IP address. This module analyzes connections between an internal source IP and an external destination IP pair.

$ rita show-beacons

WEB

Web beacons are cases where an internal host communicates with a C2 server through a CDN. The CDN will spread out the C2 traffic over multiple IPs and mix it in with legitimate traffic. RITA reverses this process to make C2 connections clearly visible.

$ rita show-beacons-sni

PROXY

Environments that use one or more proxy servers for external communication may struggle to identify beacons due to the proxy server appearing as the destination of an HTTP/HTTPS request. This module uses Proxy CONNECT header information to determine the originally requested destination for its beacon analysis.

$ rita show-beacons-proxy

STROBES

Internal to external host pairs that trigger a new connection one or more times per second are called strobes. Since these are indisputable beacons based on the frequency of communication alone, they are not scored and instead presented as a list sorted on connection count.

$ rita show-strobes

Scoring Breakdown

The score of each beacon is determined by four factors

TIMESTAMP ANALYSIS

Time interval consistency between sessions

HISTOGRAM ANALYSIS

Frequency of sessions over time

DATASIZE ANALYSIS

Size consistency between sessions

DURATION ANALYSIS

Persistency of channel within timeframe

Exploded DNS

A compromised system may leverage DNS to set up a C2 channel by encoding data in the FQDN or query portion of a DNS request. To avoid DNS caching and ensure that the local resolver forwards the request, the malware will use a unique query for every request by varying the FQDN. This results in thousands of separate resource requests to a single parent domain. This module displays the unique FQDN count and total DNS lookups for each domain.

```
$ rita show-exploded-dns
```

Long Connections

Long connections can be an indicator of well-established malware, allowing a compromised system to receive commands and exfiltrate data without constantly checking in with the C2 server. Longer sessions also create fewer log entries, making them difficult to detect. This module displays a sorted list of the longest connections and their source and destination hosts.

```
$ rita show-long-connections
```

User Agent

User agent strings can also function as indicators of compromise. Malware might use a weird or uncommon user agent string or alter one to make it appear as if it was coming from a browser or client other than the one infected. Detecting and vetting such irregularities can assist you in determining whether a communication is malicious or benign. This module displays a list of unique user agent strings found in the dataset.

```
$ rita show-useragents
```

Threat Intel

Threat intelligence feeds contain information on potentially malicious hosts based on attack information accumulated through various sources. You can customize which feeds this module uses in its analysis. Results display a list of the potentially malicious matches split into three categories: hostnames, IPs contacted via an outbound connection, and IPs that initiated an inbound connection.

```
$ rita show-bl-dest-ips
$ rita show-bl-source-ips
$ rita show-bl-hostnames
```

Getting Started

RITA runs on Linux and can be installed either manually or with our automated install script on Ubuntu 20.04 LTS, Debian 11, Security Onion, and CentOS 7.

You can find instructions for both of these on the RITA GitHub repository.

`github.com/activecm/rita`

Check out the Active Countermeasures website for some great webcasts to help you get started.

`activecountermeasures.com`

Import and Analysis

The install script will install Zeek and walk you through setting it up to monitor your network. You can also generate Zeek logs from existing PCAPs. The import step requires a 24-hour block of logs, giving RITA the fidelity it needs to look for persistent connections. Run the import command to create a dataset containing the analysis results. Another option is to create a "rolling" dataset to progressively analyze log data as it comes in, typically in 1-hour increments. This type of dataset keeps a rolling 24 hours of data, deleting older data and removing it from the analysis. A cron job or another task scheduler can help you automate this process. Once the import and analysis steps are complete, you can try out the module commands we looked at in the previous pages.

Tips and Tricks

Format module results into easy-to-read tables by adding -H to the command:

```
$ rita show-strobes <dataset> -H
```

Create a simple HTML summary of all module results:

```
$ rita html-report <dataset>
```

You can filter results of any module by using grep. One example is to create a text file with the IP addresses to exclude (one per line) and use it when piping into grep:

```
$ rita show-beacons <dataset> | grep -v -w -F -f <filename>
```

If you find a suspicious result, you can use your **Zeek logs** to gather more context clues about the hosts or connections. Check out the **Useful Threat Hunting Scripts** article for some examples.

Most modules have thresholds that can be adjusted via RITA's configuration file. The beacon modules also allow you to customize the weight each subscore *(see Score Breakdown in Beacons section)* has on the final score.

IN MEMORY OF RITA STRAND

To catch a C2 channel, you can use the tools taught in this zine to see when it calls home, and catch the malicious activity. Sometimes you can catch people this way, too. Hunt through the zine using your 3D glasses to find all the clues and figure out the names and numbers of each suspect. One clue might even help identify which number belongs to the top threat.

Once you have all the names and numbers, figure out who sits where at the table of thieves. Only then will you be able to reveal the secret code you can use to enter and win a prize.

Now go hunt those threats!

- The dairy specialist is in the top left seat
- Due to an unresolvable disagreement over breakfast drink options, the citrus grower and the dairy specialist refuse to sit next to or across from one another
- The magician wants a center seat so he can ask everyone to pick a card
- The dot-obsessed wants to sit next to the citrus grower (he smells the best)
- Cows scare the nature lover, so those two must be as far apart as possible
- The math teacher and the magician would like to sit next to each other
- The magician and the citrus grower are across from each other

SECRET CODE:

_____ _____ _____

TOP THREAT

Enter this SECRET CODE at the Spearphish General Store to get a free Threat Hunter Puzzle Champion enamel pin (yes, even the shipping is free). Hurry, those pins are limited!

AC-Hunter
Community Edition

with Chris Brenton

*For the last five years, **RITA has been the premier open-source tool** for identifying covert command and control (C2) channels. While RITA is extremely accurate, the command line interface can be challenging when you are used to working with GUI-based tools. This can add additional layers of complexity when you are trying to threat hunt your network.*

*With this in mind, we've decided to release a freeware version of our graphical C2 threat hunting tool, AC-Hunter. We will still maintain RITA, but the **free AC-Hunter Community Edition** will be an additional tool in your arsenal for finding adversaries on your network.*

```
cbrenton@cb-lab:~$ 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,Total Bytes
1,10.55.100.111,165.227.216.194,20054,92,29,52,1,5
2,7774,20053,0,0,0,0,1845020
0.838,10.55.200.10,205.251.194.64,210,308,29398,4,
300,70,109,205,0,0,0,0,64850
0.835,10.55.200.11,205.251.197.77,69,308,1197,4,30
0,70,38,68,0,0,0,0,21313
0.834,10.55.100.111,34.239.169.214,34,1259,5,14388
,1,156,15,30,0,0,0,0,42831
```

Prioritizing Systems With Threat Scoring

One of the benefits of AC-Hunter is that it assigns a **threat score** to all of your internal systems. The higher the threat score, the more likely the **system has been compromised.** This can be a huge time saver if you think an adversary has compromised multiple systems on your network.

On the right side of the screen, AC-Hunter breaks down which attributes were observed when assigning the threat score. **All of these items are clickable,** which makes it easy to dig into the data.

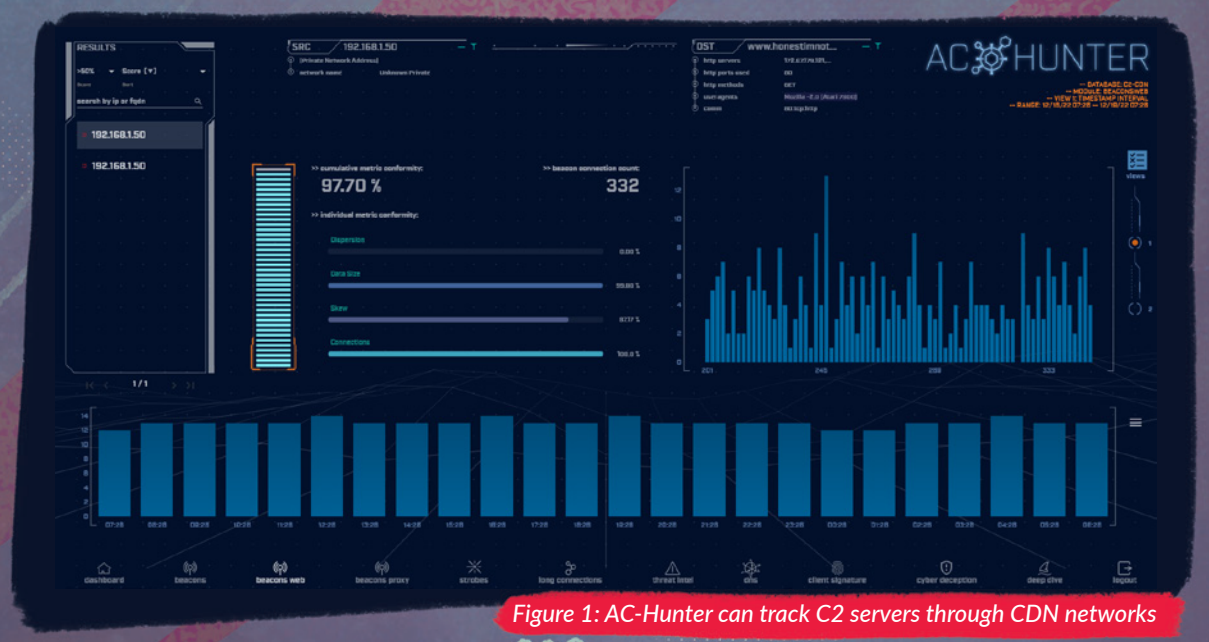


Figure 1: AC-Hunter can track C2 servers through CDN networks

We've focused on a

highly simplified interface

so that analysts of any skill level can be effective threat hunters.

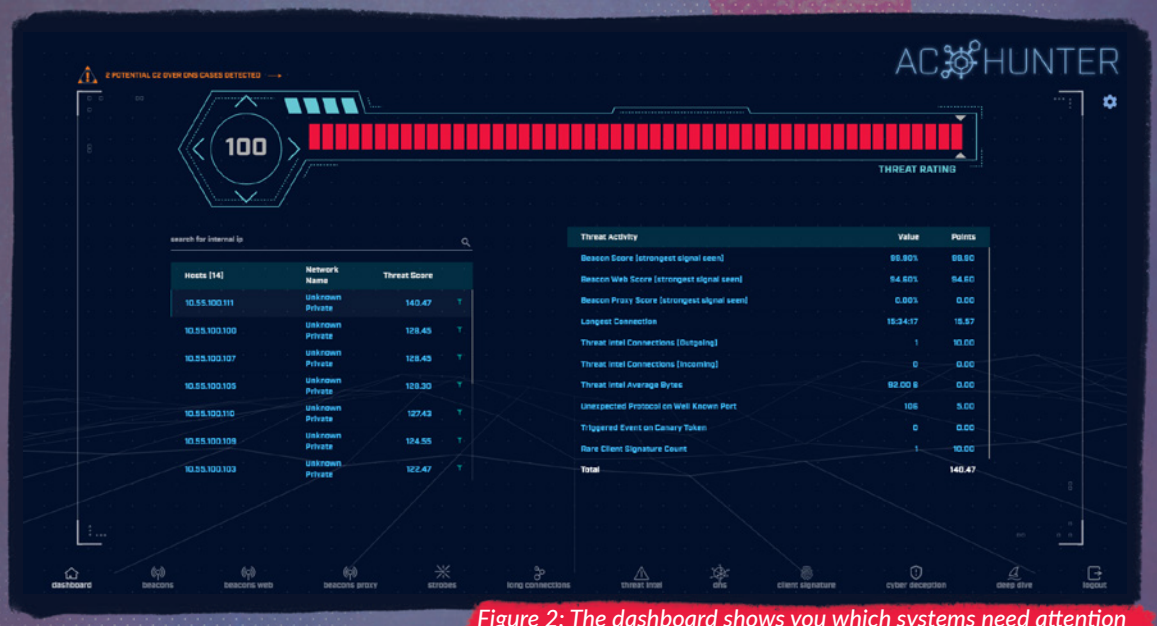


Figure 2: The dashboard shows you which systems need attention

Cyber Deception Capability

The cyber deception capability built into AC-Hunter permits you to leave **"tripwires"** in various locations around your network in order to detect lateral movement by rogue employees or potential adversaries. Deception tokens can be user accounts or resources that look like tempting targets.

For example, you can create a fake Administrator account that alerts whenever anyone tries to log in, or tantalizing private files that trigger when someone tries to view their contents. **Cyber deception provides an additional layer** of low false positive detection of potential adversaries on your network.

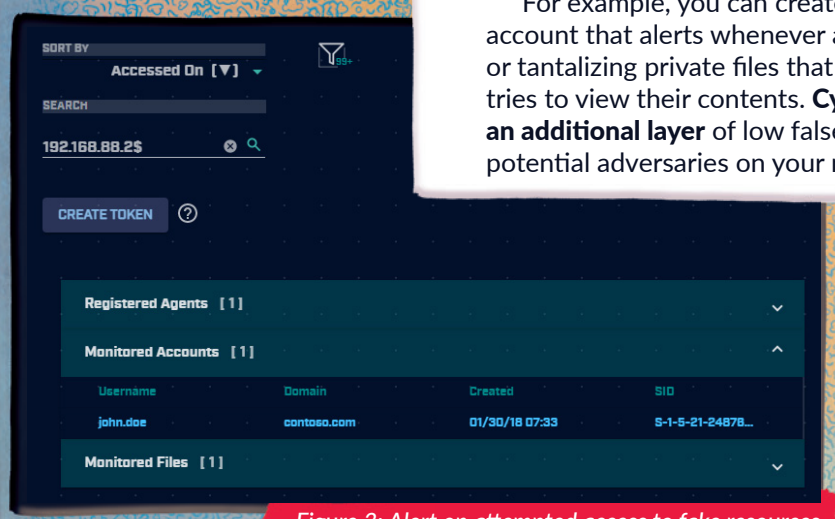


Figure 3: Alert on attempted access to fake resources

Community vs Enterprise Edition

Both the Community and the Enterprise editions of AC-Hunter use the same **patented C2 detection code** and the same simplified interface. So **both are just as capable of helping you find adversaries on your network.** The big difference comes down to day-to-day administration. The Enterprise Edition has more features designed to support daily SOC operations.

	COMMUNITY EDITION	ENTERPRISE EDITION
Threat Hunting - Beacons	✓	✓
Threat Hunting - Long Connections	✓	✓
Threat Hunting - Proxy Analysis	✓	✓
Threat Hunting - Threat Intel	✓	✓
Threat Hunting - DNS	✓	✓
Threat Hunting - Client Signature	✓	✓
Threat Hunting - Cyber Deception	✓	✓
Threat Hunting - Deep Dive	✓	✓
Number of Sensors Supported	1	Unlimited
Daily Snapshots	X	✓
Datasets	10	Unlimited
Reporting	X	✓
Customizable Menus	X	✓
Safelist Entries	50	Unlimited
Safelist Sharing	X	✓
LDAP Login Support	X	✓
Alerting	X	Syslog and Slack
Scoring Customization	X	✓
Support	Discord Community	Live Chat, Email, Videoconferencing
Cost	Free	$

Deploying AC-Hunter

One of the benefits of AC-Hunter is that network data is used to collect C2 telemetry. This means that there are **no endpoint agents to install.** Further, AC-Hunter is capable of protecting any operating system, IoT, IIoT, or network device. **All devices connected to the network are automatically protected.** AC-Hunter can be deployed on-prem or into IaaS public clouds like EC2 and Azure.

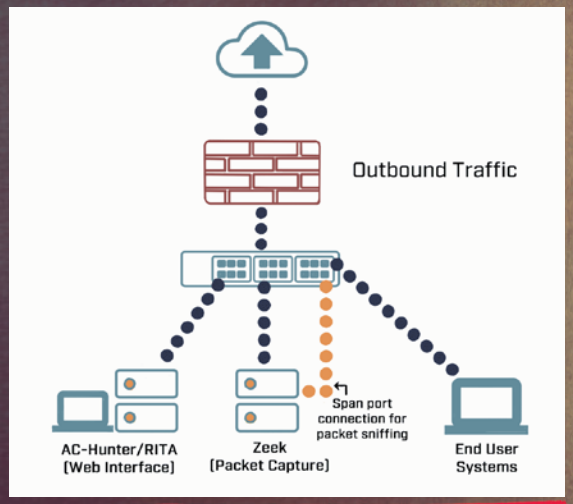
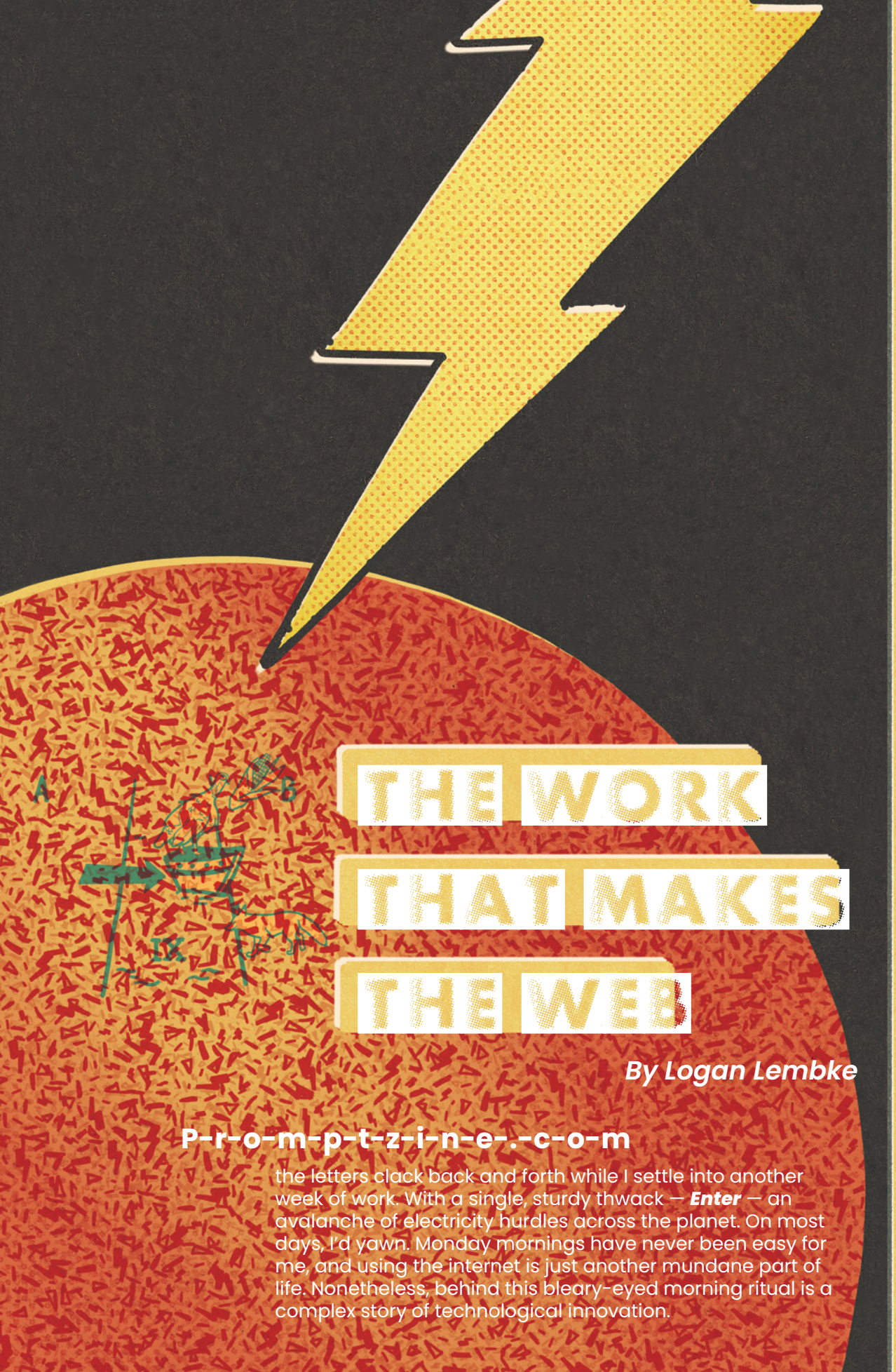


Figure 4: AC-Hunter's Zeek sensor monitors all traffic going to and from the internet

FREE DOWNLOAD

acm.re/ac-hunter-community-edition/



THE WORK THAT MAKES THE WEB

By Logan Lembke

P-r-o-m-p-t-z-i-n-e-.-c-o-m

the letters clack back and forth while I settle into another week of work. With a single, sturdy thwack — ***Enter*** — an avalanche of electricity hurdles across the planet. On most days, I'd yawn. Monday mornings have never been easy for me, and using the internet is just another mundane part of life. Nonetheless, behind this bleary-eyed morning ritual is a complex story of technological innovation.

First, the web browser has to take the link and stash it into the device's memory.

This **memory** could be a *mix of capacitors, transistors, and magnetic devices,* but each kind stores data as sequences of *on-and-off signals* known as **bits.** As a result, the web browser has to use an **encoding scheme** to store the link as a *sequence of bits.*

Unicode - an international standard *which assigns a number to* roughly 150,000 *letters and symbols.*

Unicode
Transformation
Format

The most common **encoding schemes** for *mapping **Unicode** numbers to sequences of bits* are **UTF-16** and **UTF-8.**

UTF-16 is commonly *used by Microsoft Windows*

UTF-8 is used in *most other spaces*

Importantly, **UTF-8 is backwards compatible with ASCII,** an older encoding scheme dating back to 1967.

Once the link has been stored into memory, the web browser needs to find the **IP address** of the machine responsible for hosting the webpage.

Internet
Protocol

Domain
Name
System

To accomplish this, the web browser queries the **DNS** via a **resolver.**

A **DNS resolver** is *a server which takes in DNS requests, contacts other DNS servers, and responds back to the requests with the results it has found.*

In order to contact the DNS resolver, the operating system needs to sandwich ***the domain name with a new set of bits*** to form a **DNS request.**

For example, the field called "query type" is always set to 00000001 when searching for a website's IPv4 address.

The first publication which laid out the format for DNS requests was released in 1987 under the title, Request For Comments (RFC) 1035.

Now, the DNS request needs to be packaged up for delivery.

User
Datagram
Protocol

A **UDP header** is added to the front of the newly formed DNS request.

This header includes the port numbers *(which help each machine route the data to the correct applications)* and a field which lets the receiver know if the data has been transmitted correctly.

Internet Experiment Note (IEN) 88 was the first document to define the UDP header in 1979.

Similarly, an **IP header** is tacked on to the front of the UDP header.
This new header ***contains the IP addresses of the DNS resolver and the device making the request.*** While the IP header contains a handful of other details, one important field is the **"differentiated services"** field. This field ***helps machines prioritize transmitting certain types of traffic over others*** *(like video conferencing).*

The layout for the IPv4 header can first be found in IEN 54 from 1978, but it wasn't formalized until 1981 with RFC 791.

[9x5]+[77x2] - [3x8] + 226

If the system is connected via a gigabit Ethernet cable, the IP packet must be wrapped up inside an Ethernet frame.

The most popular format for Ethernet frames dates back to 1982 in a joint publication from DEC, Intel, and Xerox.

The **Ethernet frame** adds its own fields, including ***source and destination addresses,*** signals to help each receiver understand ***the rate at which the data is being transmitted,*** and optional tags which help ***segregate and prioritize communications.***

5x111

The data is now ready to be sent.

In 1999, this method of sending data was standardized as 802.3ab by IEEE.

Gigabit Ethernet works by sending ***eight bits of data at a time.*** Each set of eight bits is mapped to a set of four voltages, one per pair of wires. In turn, each of these voltages are set to take on one of five predetermined levels.

However, which set of voltages corresponds to which set of bit patterns ***changes over time.***

While this makes determining what is happening on the wire more difficult, it ***aids in preventing radio interference and error correction.***

Next, the browser waits to hear back from the DNS resolver.

Hyper
Text
Transfer
Protocol

Transport
Layer
Security

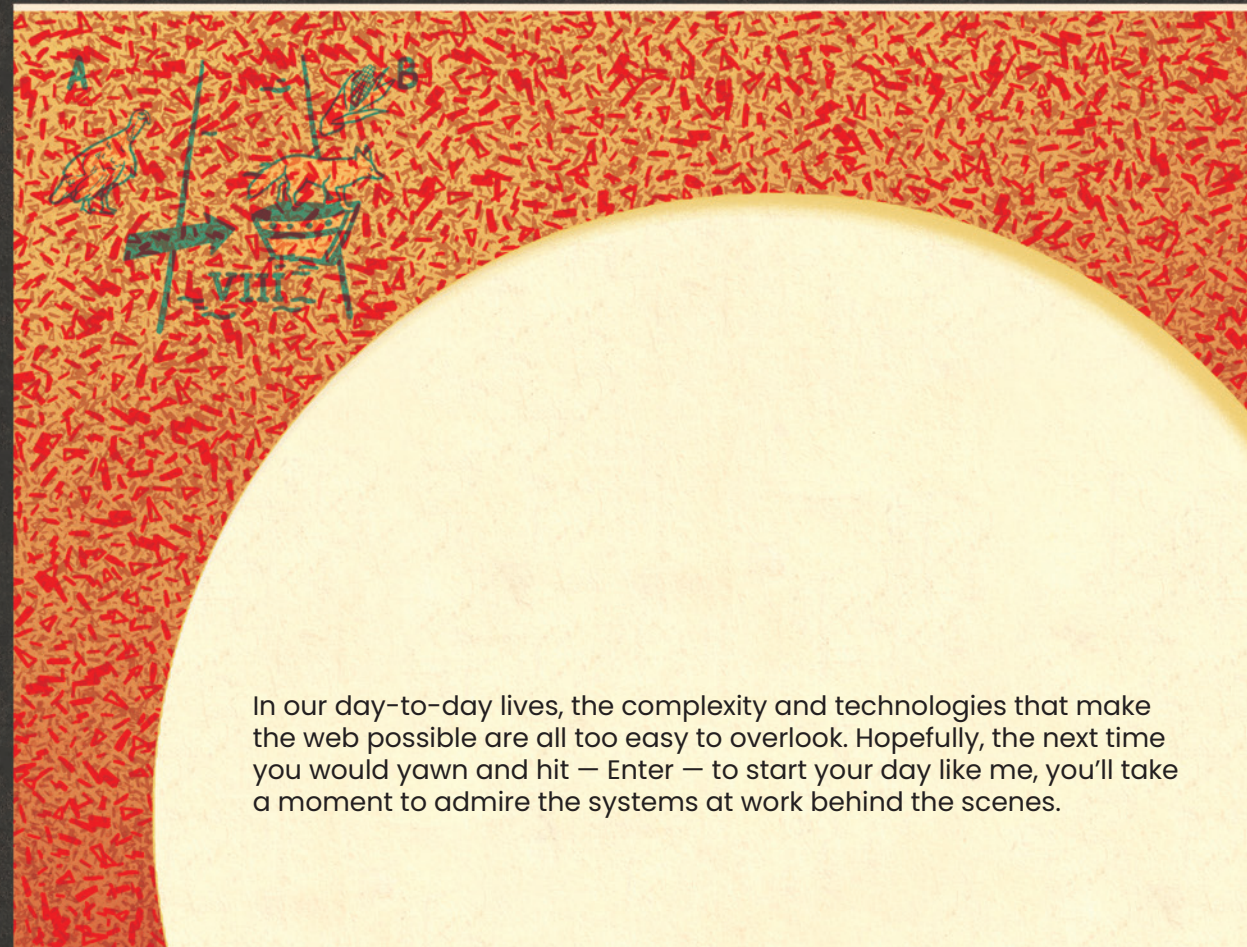
Transmission
Control
Protocol

Once it obtains an IP address for the machine serving the webpage, it will create an **HTTP request** for the webpage and wrap it up in yet another set of protocols.
This time around, instead of UDP, the request will be wrapped up with **TLS** and **TCP.**

[8^2]+3

√324

Finally, after transmitting the web request and waiting to hear back, the browser will begin displaying the page.



In our day-to-day lives, the complexity and technologies that make the web possible are all too easy to overlook. Hopefully, the next time you would yawn and hit — Enter — to start your day like me, you'll take a moment to admire the systems at work behind the scenes.

Advanced TShark Output Manipulation

sort - Sort the output data from left to right alphanumerically. Use the "-n" switch to sort numerically. Use "-r" to sort in reverse order. Use "-k" to offset where to start sorting.

uniq - Collapse multiple repeating lines into a single line. Use "-c" to count the number of lines that have been collapsed.

datamash - Perform basic statistical operations on a range of numbers. This can include counting or adding up a series of numbers, identifying the min or max value, or calculating the range or standard deviation of a series of numbers.

Example, count the number of TCP connections between two IP addresses:

```
tshark -r decode1.pcap -T fields -e ip.src -e ip.dst tcp.flags==2 and
ip.src==192.168.1.10 and ip.dst==1.2.3.4 | datamash count 1
```

Example, for a specified pair of IP addresses, print the delta time between each session:

```
tshark -r decode1.pcap -T fields -e ip.src -e ip.dst -e frame.
time_delta_displayed tcp.flags==2 and ip.src==192.168.1.10 and
ip.dst==1.2.3.4 | less
```

Example, for a specified pair of IP addresses that have connected multiple times, identify the minimum and maximum delta time between connections. Further, calculate the mean and the standard deviation (good for hunting beacons):

```
tshark -r decode1.pcap -T fields -e ip.src -e ip.dst -e frame.
time_delta_displayed tcp.flags==2 and ip.src==192.168.1.10 and
ip.dst==1.2.3.4 | datamash -g 1,2 min 3 max 3 mean 3 sstdev 3
```

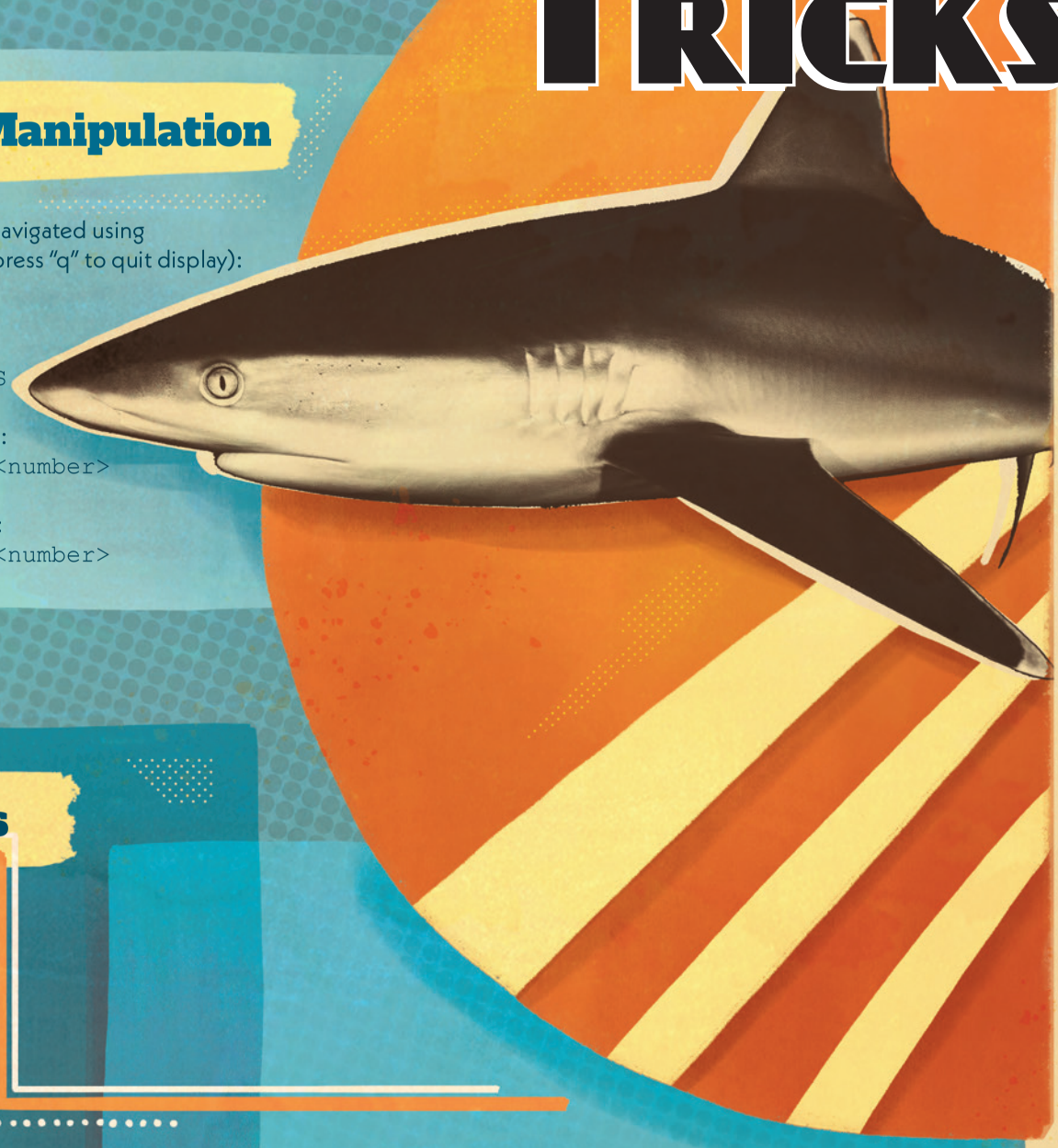
Identify which 10 internal systems are sending the most data out to the internet, and to which destination IP address:

```
tshark -r <pcap file name> -T fields -e ip.src -e ip.dst -e ip.len
ip.proto==6 and ip.src == 192.168.0.0/16 or ip.src == 10.0.0.0/8 or
ip.src == 172.16.0.0/12 | sort | datamash -g 1,2 sum 3 | sort -k 3 -rn
| head
```

Disclaimer: This is not intended to be a full description of TShark's functionality. The commands listed are the ones most commonly used.

CHRIS BRENTON'S

Cool TSHARK Tricks



Simple TShark Output Manipulation

Pipe the data through "less" so that the data can be navigated using the spacebar, page up, page down, and arrow keys (press "q" to quit display):

```
tshark -r <pcap file name> | less
```

Make output easier to read by removing line wrap:

```
tshark -r <pcap file name> | less -S
```

Print the first <number> lines of output (default is 10):

```
tshark -r <pcap file name> | head -<number>
```

Print the last <number> lines of output (default is 10):

```
tshark -r <pcap file name> | tail -<number>
```

Basic TShark Commands

Print a summary of each frame in a PCAP file:

```
tshark -r <pcap file name>
```

Disable name resolution:

```
tshark -n -r <pcap file name>
```

Print all frame fields (verbose output):

```
tshark -V -r <pcap file name>
```

Print absolute (instead of relative) timestamps:

```
tshark -t a -r <pcap file name>
```

Display frames that meet a specified criteria:

```
tshark -r <pcap file name> <display filter>
```

Example, display frames sent/received from a specified IP address:

```
tshark -r decode1.pcap ip.addr==10.0.0.204
```

Example, print frames where the time to live value is equal to 64:

```
tshark -n -r decode1.pcap ip.ttl==64
```

Example, print TCP frames where the Reset flag is turned on:

```
tshark -r decode1.pcap tcp.flags.reset!=0
```

Print frames with a display field that contains a case sensitive value:

```
tshark -r <pcap file name> <display filter> contains <value>
```

Example, print URIs that contain the string "windowsupdate":

```
tshark -r decode1.pcap http.request.uri contains "windowsupdate"
```

List of all possible display filters:

wireshark.org/docs/dfref/

Combine multiple display filters using logical "and" and "or" to further refine filtering. Example:

```
tshark -r decode1.pcap ip.src==10.0.0.204 and tcp.dstport==443
```

Send frames that match a specified display filter to a new PCAP file:

```
tshark -r <pcap file name> -w <new pcap file> <display filter>
```

Disclaimer: This is not intended to be a full description of TShark's functionality. The commands listed are the ones most commonly used.

Advanced TShark Commands

Print a summary of DNS information within a PCAP file:

```
tshark -r <pcap file name> -q -z dns,tree
```

Print a list of all endpoints that appear within a PCAP file, along with a communication summary:

```
tshark -r <pcap file name> -q -z endpoints,ip
```

Print a list of endpoints communicating with each other, along with a communication summary:

```
tshark -r <pcap file name> -q -z conv,ip
```

Display only specific frame fields:

```
tshark -r <pcap file name> -T fields -e <field 1> -e
<field 2> ...
```

Example, print the source and destination IP address, but only when the SYN flag is set:

```
tshark -n -r decode1.pcap -T fields -e ip.src -e
ip.dst tcp.flags==2
```

Extract the TCP payloads being exchanged between two IP addresses:

```
tshark -r <pcap file> -T fields -e tcp.payload
ip.addr==<1st IP> and ip.addr==<2nd IP>
```

Example, extract the HTTP traffic between two specified IP addresses, convert the Hex output to ASCII, and pause output:

```
tshark -r perimeter_class.pcap -T fields -e tcp.
payload ip.addr==192.168.1.10 and ip.addr==1.2.3.4
and tcp.port==80 | xxd -r -p | less -S
```

Example, similar to above, but extract ICMP payloads and convert to ASCII:

```
tshark -r weird-icmp.pcap -T fields -e data.data
ip.addr==192.168.1.10 and ip.addr==1.2.3.4 and icmp
| xxd -r -p | less -S
```

Advanced TShark Output Manipulation

sort - Sort the output data from left to right alphanumerically. Use the "-n" switch to sort numerically. Use "-r" to sort in reverse order. Use "-k" to offset where to start sorting.

uniq - Collapse multiple repeating lines into a single line. Use "-c" to count the number of lines that have been collapsed.

datamash - Perform basic statistical operations on a range of numbers. This can include counting or adding up a series of numbers, identifying the min or max value, or calculating the range or standard deviation of a series of numbers.

Example, count the number of TCP connections between two IP addresses:

```
tshark -r decode1.pcap -T fields -e ip.src -e ip.dst tcp.flags==2 and
ip.src==192.168.1.10 and ip.dst==1.2.3.4 | datamash count 1
```

Example, for a specified pair of IP addresses, print the delta time between each session:

```
tshark -r decode1.pcap -T fields -e ip.src -e ip.dst -e frame.
time_delta_displayed tcp.flags==2 and ip.src==192.168.1.10 and
ip.dst==1.2.3.4 | less
```

Example, for a specified pair of IP addresses that have connected multiple times, identify the minimum and maximum delta time between connections. Further, calculate the mean and the standard deviation (good for hunting beacons):

```
tshark -r decode1.pcap -T fields -e ip.src -e ip.dst -e frame.
time_delta_displayed tcp.flags==2 and ip.src==192.168.1.10 and
ip.dst==1.2.3.4 | datamash -g 1,2 min 3 max 3 mean 3 sstdev 3
```

Identify which 10 internal systems are sending the most data out to the internet, and to which destination IP address:

```
tshark -r <pcap file name> -T fields -e ip.src -e ip.dst -e ip.len
ip.proto==6 and ip.src == 192.168.0.0/16 or ip.src == 10.0.0.0/8 or
ip.src == 172.16.0.0/12 | sort | datamash -g 1,2 sum 3 | sort -k 3 -rn
| head
```

CHRIS BRENTON'S

Cool TShark Tricks

Disclaimer: This is not intended to be a full description of TShark's functionality. The commands listed are the ones most commonly used.

Advanced TShark Commands

Print a summary of DNS information within a PCAP file:

```
tshark -r <pcap file name> -q -z dns,tree
```

Print a list of all endpoints that appear within a PCAP file, along with a communication summary:

```
tshark -r <pcap file name> -q -z endpoints,ip
```

Print a list of endpoints communicating with each other, along with a communication summary:

```
tshark -r <pcap file name> -q -z conv,ip
```

Display only specific frame fields:

```
tshark -r <pcap file name> -T fields -e <field 1> -e
<field 2> ...
```

Example, print the source and destination IP address, but only when the SYN flag is set:

```
tshark -n -r decode1.pcap -T fields -e ip.src -e
ip.dst tcp.flags==2
```

Extract the TCP payloads being exchanged between two IP addresses:

```
tshark -r <pcap file> -T fields -e tcp.payload
ip.addr==<1st IP> and ip.addr==<2nd IP>
```

Example, extract the HTTP traffic between two specified IP addresses, convert the Hex output to ASCII, and pause output:

```
tshark -r perimeter_class.pcap -T fields -e tcp.
payload ip.addr==192.168.1.10 and ip.addr==1.2.3.4
and tcp.port==80 | xxd -r -p | less -S
```

Example, similar to above, but extract ICMP payloads and convert to ASCII:

```
tshark -r weird-icmp.pcap -T fields -e data.data
ip.addr==192.168.1.10 and ip.addr==1.2.3.4 and icmp
| xxd -r -p | less -S
```

CHRIS BRENTON'S

Cool TShark Tricks

You've finally gotten into an adversary's network and are running a piece of malware that will do your bidding to probe and attack their systems.

The only question is:

how do you send commands to that malware?

The ideal channel would:

1) be open on almost every single network and

2) have lots of other traffic to hide the covert communication.

Unfortunately, this exists – DNS.

Domain **N**ame **S**erver

The same protocol we use to look up **hostnames** can also carry command and control traffic.

Here's how it works:

♥ - The attacker creates a domain ***(like "polkawrench.info")*** used for command and control, and sets up a server ready to answer queries to it.

♠ - The infected system announces itself by making a DNS query in that domain, such as ***"6.7.8.9.newsystem.polkawrench.info"***. The attacker's DNS server pulls this apart and realizes that the computer at "6.7.8.9" is a newly infected system and sends back some generic acknowledgement like "1.1.1.1".

♦ - Now the infected system checks in every 5 minutes by placing a query like ***"timestamp.6.7.8.9.checkin.polkawrench.info"***. The attacker's DNS server replies with ***"0.0.0.0"*** to say "I have no work for you to do." This goes on for hours or days.

♣ - When the attacker does have a job to do, it waits until the next check-in and responds with ***"2.6.8.7" or some other code describing what to do.*** The infected system does that task and sends back the results on the next check-in.

The advantage is very clear; since nearly all networks allow DNS requests to go out and DNS replies to come back, ***the attacker has an almost guaranteed way to communicate with their infected systems.*** Like most network threats, the attacker has a short wait for the next check-in to send a command. The slightly more annoying downside is that DNS requests and replies don't have much space. DNS requests can be up to 256 characters. Replies from the attacker's DNS server can be longer if we send back TXT answers (generic text) instead of A answers (like our "2.6.8.7" address above).

As network defenders, it's our job to run a threat hunting tool that can look for domains ***(like "polkawrench.info")*** that have a lot of unique requests and replies crossing the perimeter. **These are worth investigating.**

SMALL TALK

WITH THE THREAT HUNTING TOOLKIT

By Ethan Robish

Investigations during a threat hunt or incident response are much like getting to know someone. In this article, you'll find a series of questions you can ask to learn more about a new ~~person~~ non-biological entity. With help from the Threat Hunting Toolkit, I'll explain where you can find information in the logs, and provide a command to print it.

You can think of the Threat Hunting Toolkit (THT) as a command line environment containing a curated selection of data-processing utilities you can take with you anywhere, much like a craftsman's toolkit. While THT is designed with processing Zeek network logs in mind (both TSV and JSON), we've also used it successfully with other delimited logs, such as comma or whitespace separated.

Without further ado, let's make some small talk.

WHAT'S YOUR NAME?

A name can tell a lot about a person. The same goes for a system's host name. The host name shows up in different network service communication and can be pulled from a number of Zeek logs.

In the **dns log,** typically the **query field** *will contain a domain* and the **answers field** *will contain one or more IP addresses,* though it can also contain additional domains. *(Note: For **PTR requests,** you can parse an IP address from the query field and a domain from the answers field instead.)*

In the **ssl log,** the **server_name** field will contain a domain that corresponds with the IP address in the **id.resp_h field.** In the **http log,** the **host field** will contain a domain instead.

There are additional sources for systems on the local network as well. These services aren't usually seen in internet-bound traffic and thus, only apply to the local network.

LOG	DOMAIN OR HOST NAME	IP ADDRESS
dns	query	answers
http	host	id.resp_h
ssl	server_name	id.resp_h
dhcp	host_name,domain, client_fqdn	assigned_addr, client_addr
kerberos	client	id.orig_h
ntlm	server_dns_computer_name, server_nb_computer_name	id.resp_h

Here is the form of the command that will print the host names and IP addresses from a given log, sorted by the most frequently seen.

```
filter --http $IP | chop host id.resp_h | mfo
```

EXAMPLE

```
filter --http 165.227.88.15 | chop host id.resp_h | mfo
  18438 drock.saintjameschurch.org      165.227.88.15
```

WHAT DO YOU DO?

Just like a person's chosen profession or hobbies, the services used by or running on the system in question **can reveal a lot about its purpose.**

The **conn log** *contains the port (id.resp_p), protocol (proto), and service (service) for each connection.* This command can tell us what kinds of services an IP address is running by showing those with the most connections. **id.resp_h** *contains the destination IP address,* meaning that it is receiving the connection on a listening port.

```
filter --conn $IP | chop id.resp_h id.resp_p proto service |
filter $IP | mfo
```

EXAMPLE

```
$ filter --conn 123.11.22.33 | chop id.resp_h id.resp_p
proto service | mfo
   3631 123.11.22.33  8080    tcp     ssl
    296 123.11.22.33  443     tcp     ssl
     43 123.11.22.33  22      tcp     ssh
      1 123.11.22.33  80      tcp     http
```

The services running can provide insight into the purpose of the server, as well as providing a sniff test. While investigating a **network time protocol (NTP) server,** I discovered the same IP address was hosting WordPress and a Minecraft server as well. This didn't exactly inspire confidence.

More often than not, I will use the **conn-summary** command to provide an overview for a system (or pair of systems). It's a little bit like looking at the system's résumé.

```
filter --conn $IP | conn-summary
```

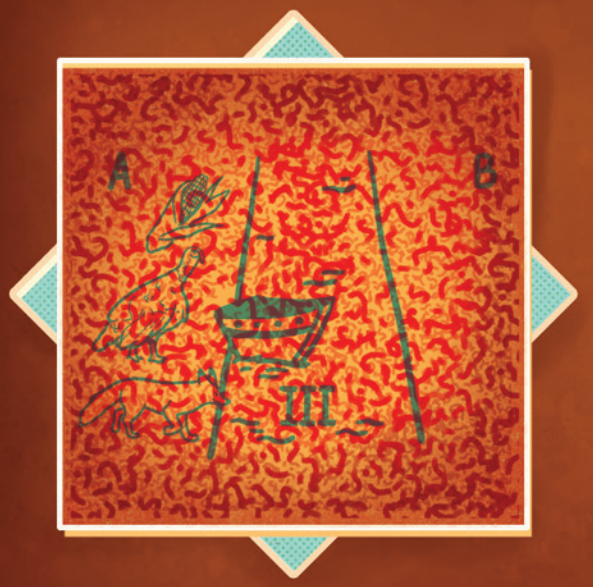
conn-summary will *display a breakdown of traffic between private and public IP ranges (so-called north-south traffic).* The **--all** flag will *summarize all connections,* including traffic only between private IP addresses *(east-west traffic).*

EXAMPLE

```
$ filter --conn 10.10.174.74 | conn-summary
Connections:
============

>== Total === 2023-01-08-00-04-52 - 2023-01-08-23-57-26
  - Connections  819.0 - Payload 13.1m -
    Ports        | Sources                 | Destinations          | Services          | Protocols | States
    443    83.6% | 10.10.174.74      100.0% | 20.190.9.86     10.0% | ssl        83.6% | 6  99.9% | SF      81.1%
    80     16.2% |                         | 35.186.224.25    5.9% | http       16.2% | 17  0.1% | RSTO    10.0%
    123     0.1% |                         | 96.17.76.118     5.1% | ntp         0.1% |           | RSTR     8.9%
                 |                         | 20.69.130.185    4.2% |                  |           |
                 |                         | 13.107.42.12     3.7% |                  |           |
                 |                         | 104.64.63.116    2.6% |                  |           |
                 |                         | 104.214.104.116  2.2% |                  |           |
                 |                         | 52.113.194.132   1.6% |                  |           |
                 |                         | 52.168.117.170   1.5% |                  |           |
                 |                         | 13.89.179.8      1.5% |                  |           |

Bytes:
======

>== Total === 2023-01-08-00-04-52 - 2023-01-08-23-57-26
  - Connections  819.0 - Payload 13.1m -
    Ports        | Sources                 | Destinations          | Services          | Protocols | States
    443    61.2% | 10.10.174.74      100.0% | 72.21.81.240    15.9% | ssl        61.2% | 6 100.0% | SF      88.8%
    80     38.8% |                         | 96.17.13.99      9.7% | http       38.8% | 17   0.0% | RSTO     6.9%
    123     0.0% |                         | 8.252.82.254     7.9% | ntp         0.0% |           | RSTR     4.2%
                 |                         | 69.164.2.128     4.7% |                  |           |
                 |                         | 20.62.190.191    3.1% |                  |           |
                 |                         | 20.69.130.185    3.0% |                  |           |
                 |                         | 13.107.42.12     2.9% |                  |           |
                 |                         | 20.190.9.86      2.8% |                  |           |
                 |                         | 35.186.224.25    2.2% |                  |           |
                 |                         | 184.87.165.39    2.0% |                  |           |
```

You can visit **ethack.github.io/tht/** and search **"conn-summary"** to read more about the conn-summary output format.

WHERE ARE YOU FROM?



Where a person was born, to some extent, sheds light on what life was like for them growing up. In this case, we're going to find out which organization owns the IP address. This can tell us if the IP is from a private ISP, big name CDN, or is owned by an organization.

```
whois $IP
asn $IP
echo $IP | whois-bulk
```

whois and **asn** both take a single IP address or domain for an argument. You can pipe a long list of IP addresses to **whois-bulk** to get a quick summary table.

EXAMPLE

```
ASN lookup for 8.8.8.8

8.8.8.8 -PTR  dns.google
        -ASN  15169 (GOOGLE, US)
        -ORG  Google LLC
        -NET  8.8.8.0/24 (LVLT-GOGL-8-8-8)
        -ABU  abuse@level3.com / network-abuse@google.com
        -ROA  ✓ VALID (1 ROA found)
        -TYP  Anycast IP  DC  Google Cloud
        -GEO  London, Westminster (GB)
        -POR  Open ports: 53, 443
        -REP  ✓  KNOWN GOOD as "Google Public DNS"
```

DO YOU COME HERE OFTEN?

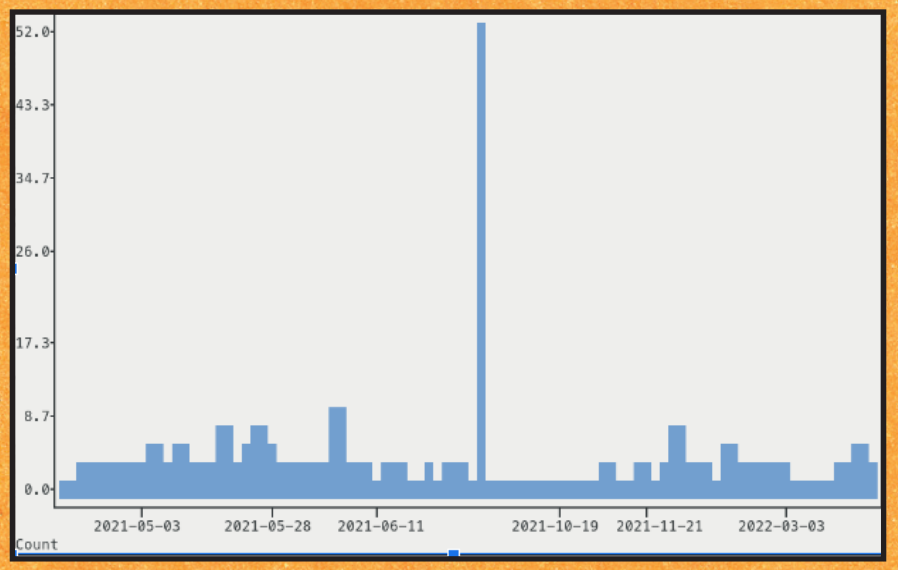
This might be more of a pick-up line than small talk, but it sure is useful to know whether a system has been seen numerous times in the past or if the current behavior is an anomaly. The easiest way to see a trend over time is with a graph. This command will show the number of SSL connections made per day. Empty days are hidden in this case.

```
filter --ssl $DOMAIN | chop ts | ts2 | freq | plot-bar
```

You can use the same formula in any log that contains a timestamp field.

EXAMPLE

```
filter --ssl example.com | chop ts | ts2 | freq | plot-bar
```



CONCLUSION

Just like with people, as you learn more about a system, you start to develop a first impression. This impression can be good or bad, trustworthy or untrustworthy. With threat hunting, this can lead you further down the investigation path or help you decide to move on to other investigations.

You can find more information at the THT project's homepage:
github.com/ethack/tht/

THT COMMANDS
CHEAT SHEET

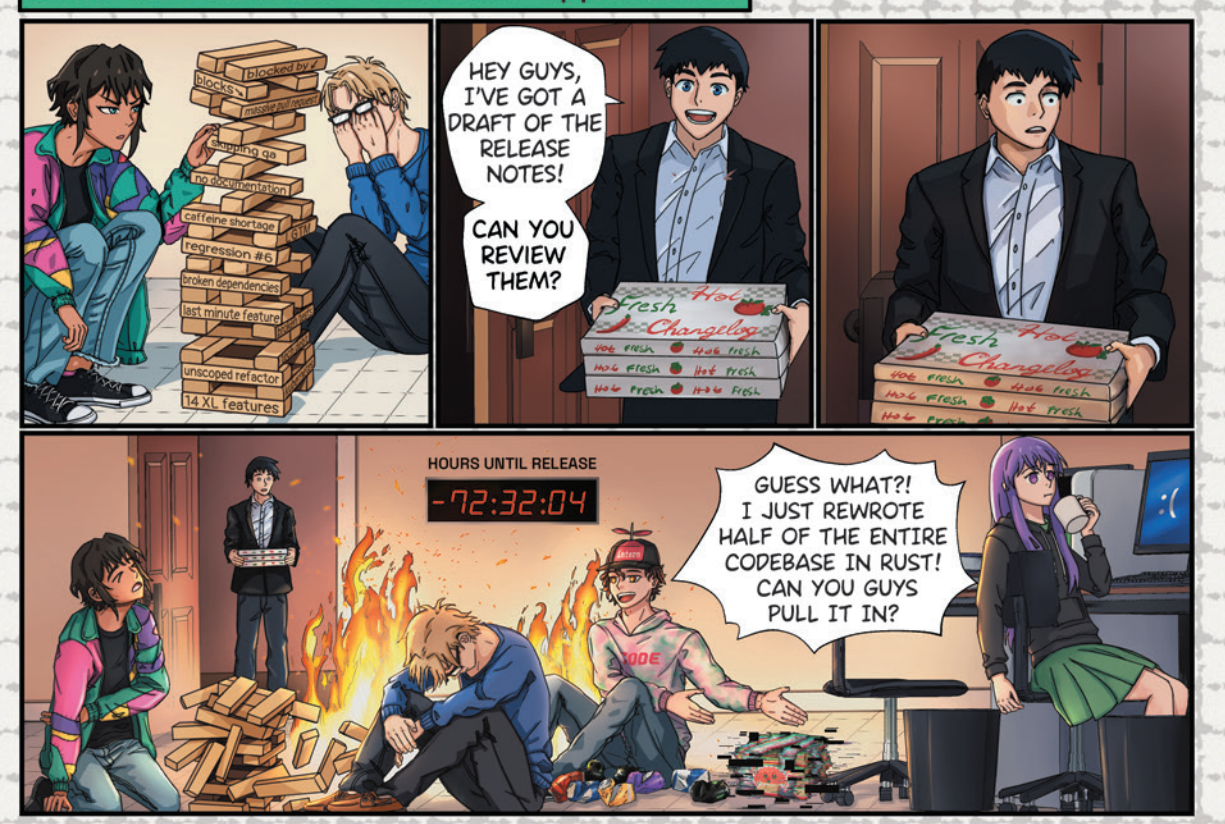
COMMAND	PURPOSE	ALTERNATIVE
`filter`	**search** within files	`find | grep`
`chop`	**select** columns	`cut` OR `zeek-cut`
`count`	non-comment line **count**	`grep -v '^#' | wc -l`
`freq`	**frequency** counts	`sort | uniq -c`
`mfo`	**most frequent occurence**	`sort | uniq -c | sort -nr`
`distinct`	**unique** elements	`sort | uniq`
`countdistinct` OR `card`	**count of unique** elements **(cardinality)**	`sort | uniq | wc -l`
`first`	sorts and prints the **first** element	`sort | head -1`
`last`	sorts and prints the **last** element	`sort | tail -1`
`skip`	**skips** elements and prints the rest	`tail -n +2`
`ts2`	convert/truncate **timestamps**	
`plot-bar`	produce a bar **graph**	

a day in the life of a programmer

sometimes it's better to step away for a while

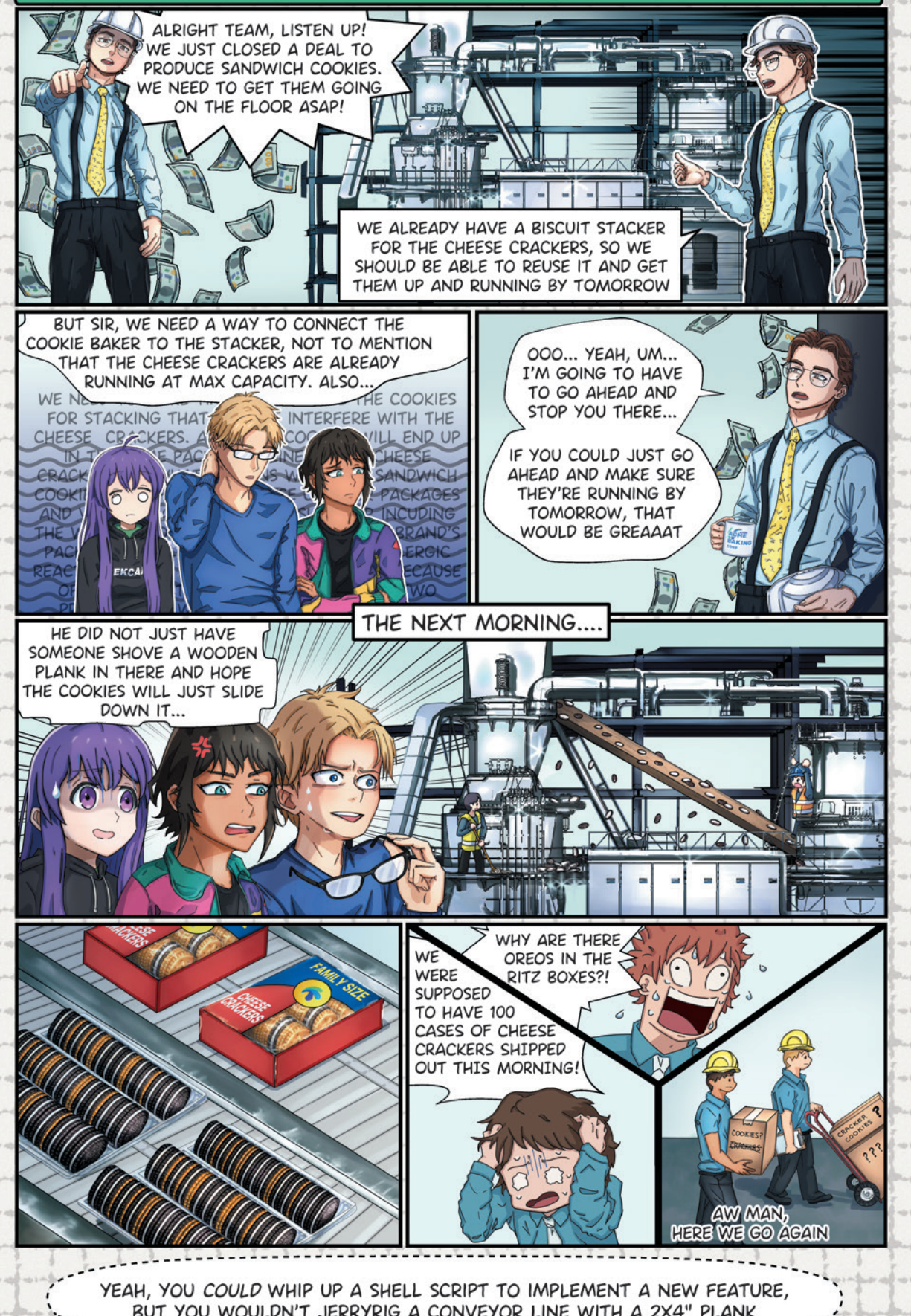
interruptions making you lose your train of thought

when the release deadline approaches



BY NAOMI KRAMER, A PROGRAMMER

why will this feature take so long? just write a script...



YEAH, YOU *COULD* WHIP UP A SHELL SCRIPT TO IMPLEMENT A NEW FEATURE, BUT YOU WOULDN'T JERRYRIG A CONVEYOR LINE WITH A 2X4" PLANK

HELPFUL

MAKE IT EASIER TO FIND THE BAD GUYS

THREAT HUNTING SCRIPTS

TRICKS AND TIPS YOU CAN USE WITH THREAT HUNTING TOOLS

BY LIZA TSIBUR

CLEAN UP THE OUTPUT

MORE THAN ONE WAY TO SOLVE A PROBLEM

RITA

By default, RITA produces comma-separated output. While this format is efficient, the lack of whitespace can make it challenging to read.

```
rita show-beacons lab1 | head -5
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,Total Bytes
1,10.55.100.111,165.227.216.194,20054,92,29,52,1,52,7774,20053,0,0,0,0,1845020
0.838,10.55.200.10,205.251.194.64,210,308,29398,4,300,70,109,205,0,0,0,0,64850
0.835,10.55.200.11,205.251.197.77,69,308,1197,4,300,70,38,68,0,0,0,0,21313
0.834,10.55.100.111,34.239.169.214,34,1259,5,14388,1,156,15,30,0,0,0,0,42831
```

Adding ASCII Formatting

You can use the "-H" switch to add some ASCII formatting to the RITA output. Piping the output through "less -S" will avoid line wrapping.

```
rita show-beacons lab1 -H | less -S
+-------+---------------+-----------------+-------------+-----------+-------------+------------+
| SCORE |   SOURCE IP   | DESTINATION IP  | CONNECTIONS | AVG BYTES | INTVL RANGE | SIZE RANGE |
+-------+---------------+-----------------+-------------+-----------+-------------+------------+
|     1 | 10.55.100.111 | 165.227.216.194 |       20054 |        92 |          29 |         52 |
| 0.838 | 10.55.200.10  | 205.251.194.64  |         210 |       308 |       29398 |          4 |
| 0.835 | 10.55.200.11  | 205.251.197.77  |          69 |       308 |        1197 |          4 |
| 0.834 | 10.55.100.111 | 34.239.169.214  |          34 |      1259 |           5 |      14388 |
```

Redirect RITA HTML Output

RITA's "html-report" option outputs data into a series of webpages. This makes it easier to browse the output using a web browser. The problem is RITA defaults to creating the HTML files under the current directory. What if you want to place the files somewhere else? Create a one line script with the following entry:

```
(cd /var/www/html; rita html-report $1)
```

Now when you run the script, you simply specify the dataset you want to see in HTML format. Rather than creating the files in the current location, they will be created under "/var/www/html".

Excluding Previously Hunted IPs from RITA

RITA does not support safelisting to remove previously hunted IPs. However, you can work around this by piping the output through "grep." You will want to use grep with the following switches:

- `-v` = Print all lines that do not match the specified pattern
- `-F` = Match as a fixed string
- `-w` = Only match when pattern has white space before and after
- `-f` = Load match entries from a file

Next, add the IP addresses you have previously hunted to a file, one per line. For this example, we'll name the file "exception.txt". Now it's a simple matter of redirecting the RITA output through the grep command:

```
rita show-beacons lab1 | grep -v
-w -F -f exception.txt
```

The final output will contain beacon data for every IP address except for the ones you have listed in the exception.txt file.

Zeek

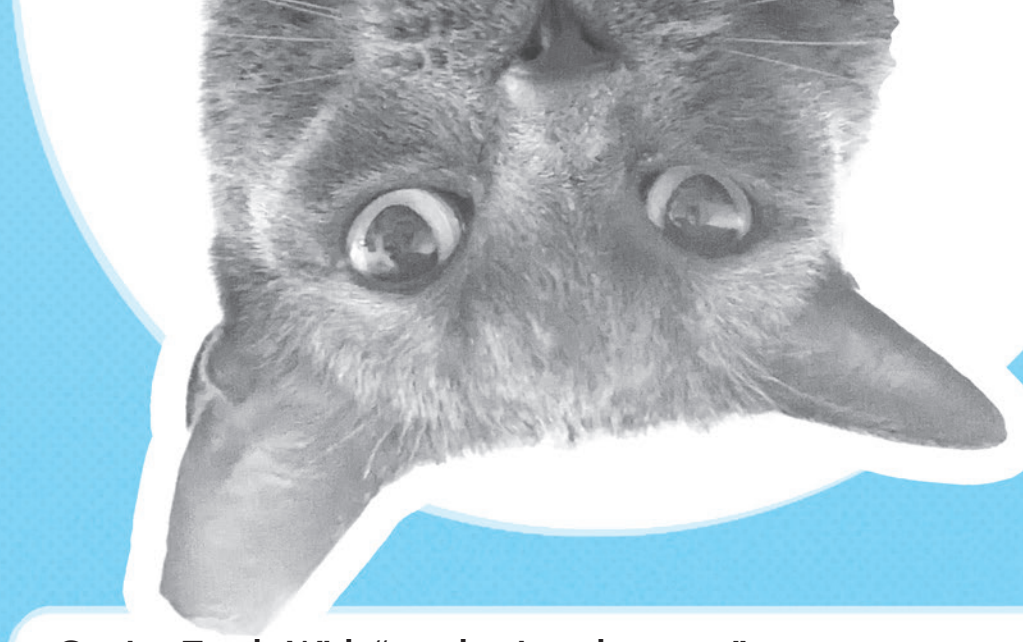
Zeek provides a summary of all network connections. This information is detailed but can be challenging to sort through.

Quick Info Dump

When you want to investigate an IP address or fully qualified domain name (FQDN), Zeek can provide helpful information, which is spread across multiple files. Here's a handy script:

```
echo 'DNS info'
cat dns.* | zeek-cut answers query | sort | uniq | grep -Fw $1
echo 'HTTP info'
cat http.* | zeek-cut id.resp_h host user_agent | sort | uniq | grep -Fw $1
echo 'TLS info'
cat ssl.* | zeek-cut id.resp_h server_name validation_status | sort | uniq | grep -Fw $1
```

When you run the script followed by an IP address or FQDN, it will search multiple files and print out identification data. If no data is present, the script simply prints a blank line. If your logs are in compressed format, use "zcat" instead of "cat".



Getting Totals With "sort | uniq -c | sort -rn"

In many cases, you may want to identify the quantity of an attribute that has been seen on the network. For example: which IP pairs created the greatest number of connections? Which internal system sent the most data to the internet? Which IP pairs generated the longest duration connections? Using sort and uniq can help you quickly total these values.

sort | uniq | sort Examples

To see the top 10 IP pairs that are creating the most number of connections:

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

Top 10 user agent strings seen on the network:

```
cat http.log | zeek-cut user_agent | sort | uniq -c | sort -rn | head
```

The "-k" switch can be used with sort to change the column used for sorting. For example, to identify the top 10 longest duration connections when the duration field is in the third column, you can run the command:

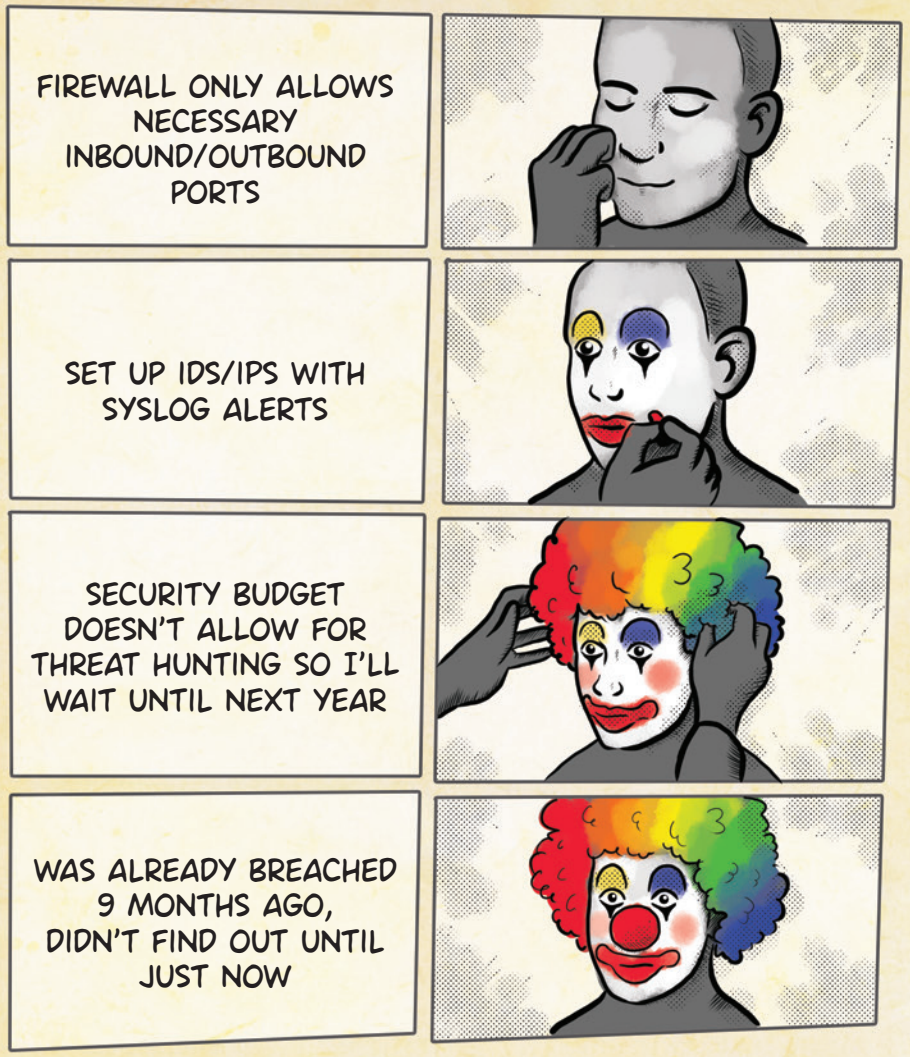
```
zeek-cut id.orig_h id.resp_h duration | sort -k3 -rn | head
```

It can also be useful to pipe the output through additional tools prior to presenting it. For example, what if we wanted to see the longest cumulative communication time between each pair of IP addresses? We can't extract that data directly, as there may be multiple connections between IP pairs that need to be added together. Datamash can be used to add values in a column. Here's an example:

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

In this example, datamash will add up (sum) all of the values in the third column (duration) when the values in the first two columns (-g 1,2) are the same. Note that we could easily script this command to reduce the amount of typing that is required each time we wish to run it.

FUNNY PAGES




```
sudo ngrep -q
interface: eth0
filter: ((ip || ip6) || (vlan && (ip || ip6)))

U 67.205.148.134:7 -> 161.35.113.192:7 #1
  I know you are but what am I?.
U 161.35.113.192:7 -> 67.205.148.134:7 #2
  I know you are but what am I?.
U 67.205.148.134:7 -> 161.35.113.192:7 #3
  I know you are but what am I?.
U 161.35.113.192:7 -> 67.205.148.134:7 #4
  I know you are but what am I?.
U 67.205.148.134:7 -> 161.35.113.192:7 #5
  I know you are but what am I?.
U 161.35.113.192:7 -> 67.205.148.134:7 #6
  I know you are but what am I?.
```

AS IP HAS MATURED, IT HAS OUTGROWN SOME OF ITS MORE CHILDISH PROTOCOLS

```
18:01:26.050454 IP 67.205.148.134.2632 > 161.35.113.192.80:
Flags [FPU], seq 821843089, win 512, urg 0, length 0

18:01:27.049688 IP 67.205.148.134.2633 > 161.35.113.192.80:
Flags [FPU], seq 373643960, win 512, urg 0, length 0

18:01:28.049829 IP 67.205.148.134.2634 > 161.35.113.192.80:
Flags [FPU], seq 270386048, win 512, urg 0, length 0
```

O CHRISTMAS TREE, O CHRISTMAS TREE, HOW LOVELY ARE THY PACKETS

```
16:22:06.240444 IP 73.54.58.10.3860 > 161.35.113.192.22:
Flags [P.], seq 481:561, ack 144, win 510, length 80

16:22:06.240485 IP 161.35.113.192.22 > 73.54.58.10.3860:
Flags [.], ack 561, win 501, length 0

16:23:06.074933 IP 73.54.58.10.3860 > 161.35.113.192.22:
Flags [P.], seq 561:641, ack 144, win 510, length 80

16:23:06.074971 IP 161.35.113.192.22 > 73.54.58.10.3860:
Flags [.], ack 641, win 501, length 0

16:24:06.107069 IP 73.54.58.10.3860 > 161.35.113.192.22:
Flags [P.], seq 641:721, ack 144, win 510, length 80

16:24:06.107107 IP 161.35.113.192.22 > 73.54.58.10.3860:
Flags [.], ack 721, win 501, length 0

16:25:06.258165 IP 73.54.58.10.3860 > 161.35.113.192.22:
Flags [P.], seq 721:801, ack 144, win 510, length 80

16:25:06.258208 IP 161.35.113.192.22 > 73.54.58.10.3860:
Flags [.], ack 801, win 501, length 0
```

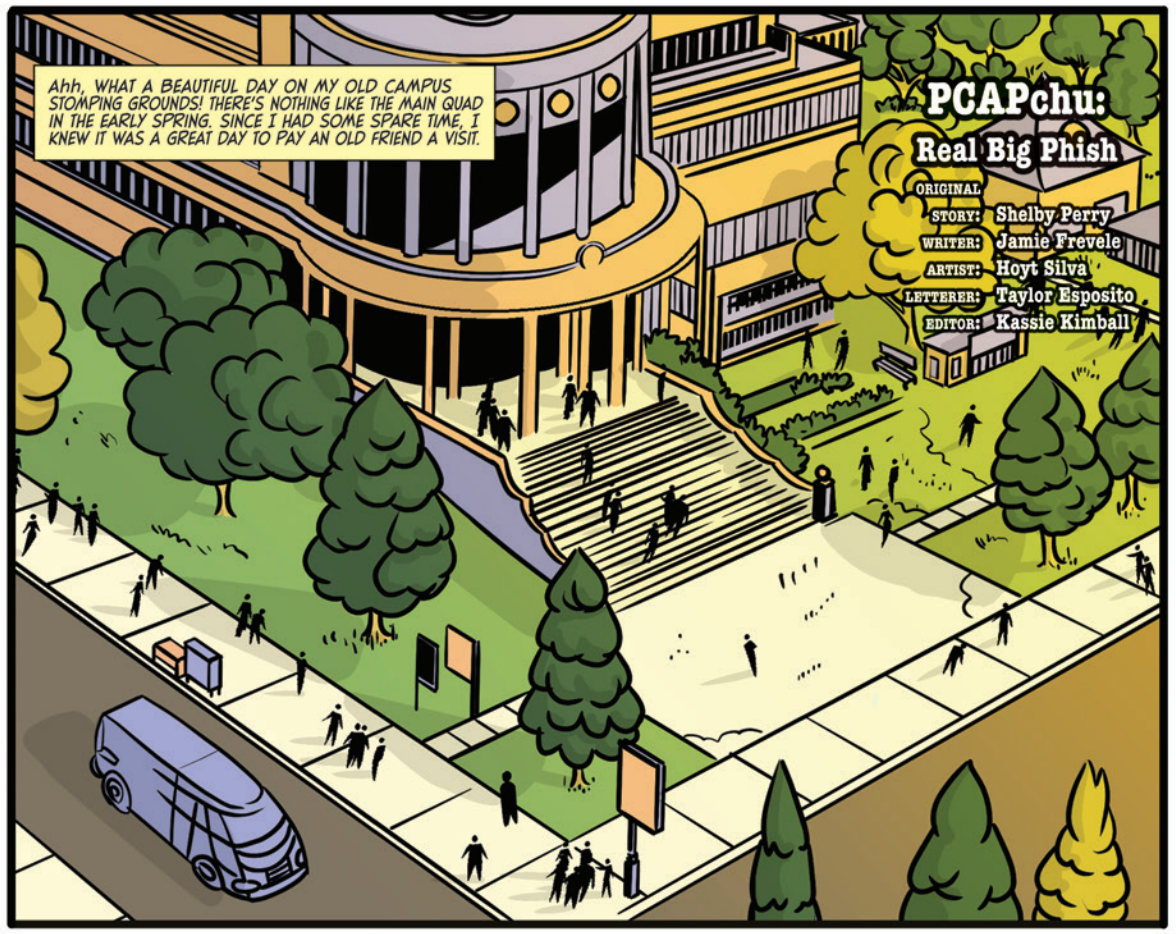
ACK, ACK, ACK, ACK, KEEPIN' ALIVE, KEEPIN' ALIVE

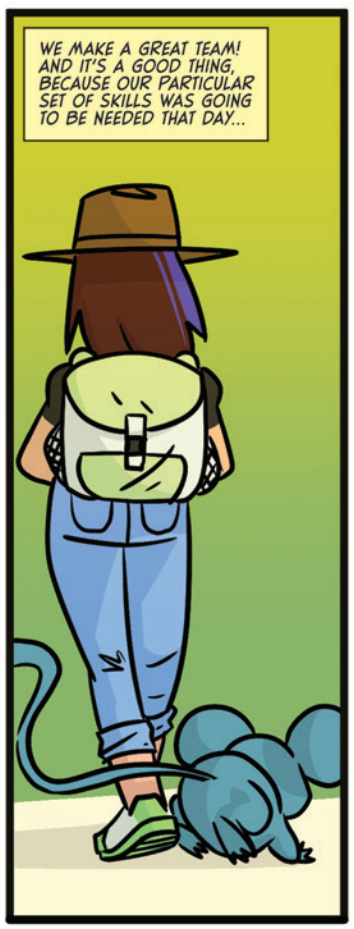
THE WWH FILES

THE BREACH IS OUT THERE

wildwesthackinfest.com

I WANT TO HACK





THAT OLD FRIEND I MENTIONED IS MY FORMER VISUAL ARTS PROFESSOR. PROFESSOR CREDUL WAS MY **FAVORITE** PROFESSOR WHILE I WAS HERE, AND I WANTED TO TELL HIM ABOUT THE AMAZING CAREER PATH I'VE CHOSEN IN CYBERSECURITY. I WANTED TO MAKE HIM PROUD.

PROFESSOR CREDUL?

LIZA! HOW GREAT TO SEE YOU! WHAT BRINGS YOU HERE TODAY?

I HAD A DAY OFF AND WANTED TO SURPRISE YOU!

WELL, I'M DEFINITELY SURPRISED!

IN A GOOD WAY, I HOPE!

OF COURSE! THOUGH I WAS JUST TRYING TO FIGURE OUT A DIFFERENT SURPRISE WHEN YOU AND YOUR LITTLE BUDDY WALKED IN.

WHAT'S THE MATTER?

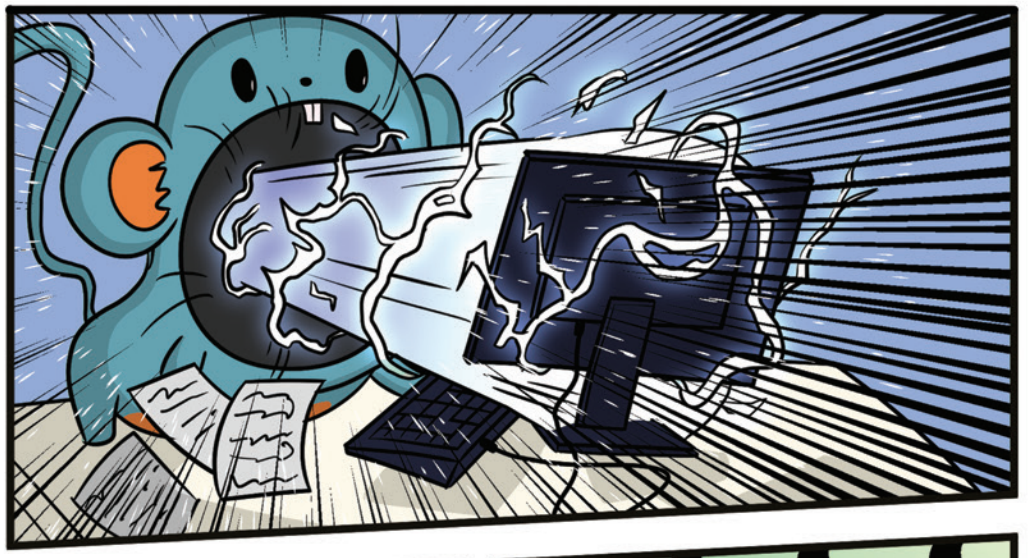
I DON'T KNOW HOW IT HAPPENED, BUT A BUNCH OF MY FILES ARE NOT WHERE THEY SHOULD BE, LIKE SOMEONE MOVED THEM AROUND.

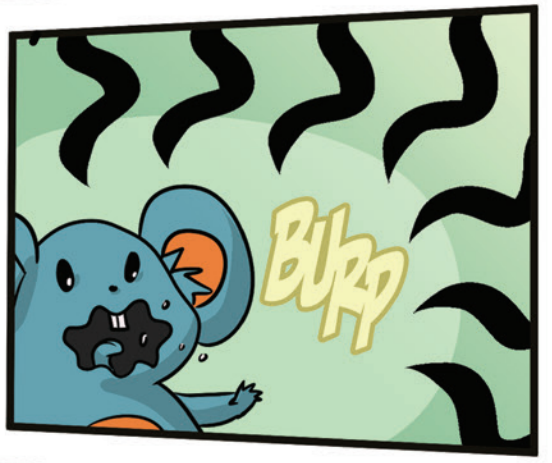
Hmmmm...

AND THE WHOLE THING'S JUST RUNNING SO SLOW.

DO YOU MIND IF PCAPchu AND I HAVE A LOOK?

GO RIGHT AHEAD!





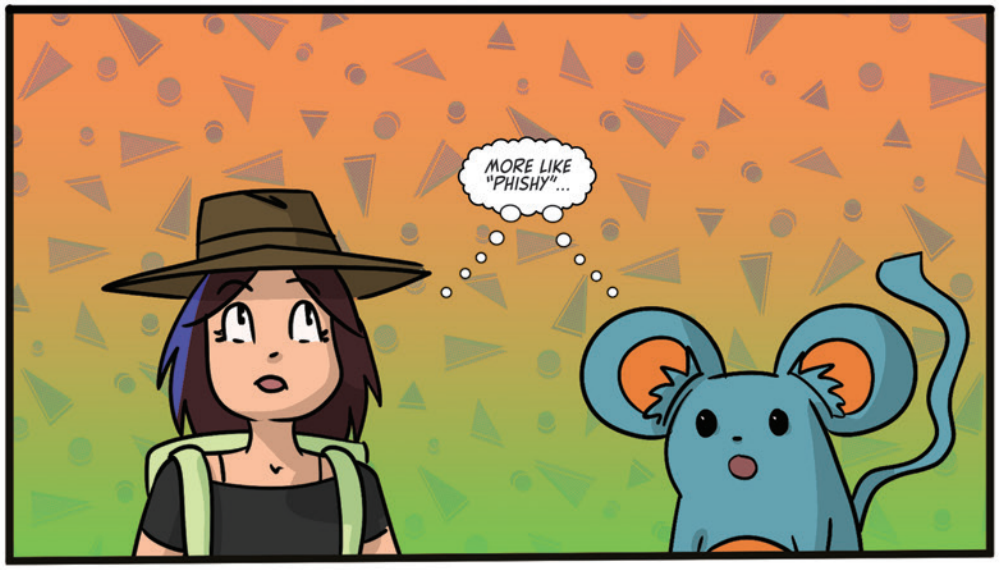
Uh oh.

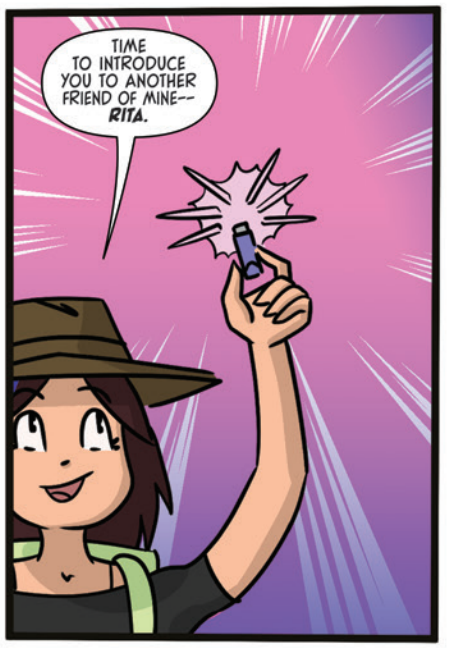
Uh oh?

PCAPchu'S EARS INDICATE THAT THERE'S SOMETHING SUSPICIOUS IN THE DATA HE JUST ABSORBED.

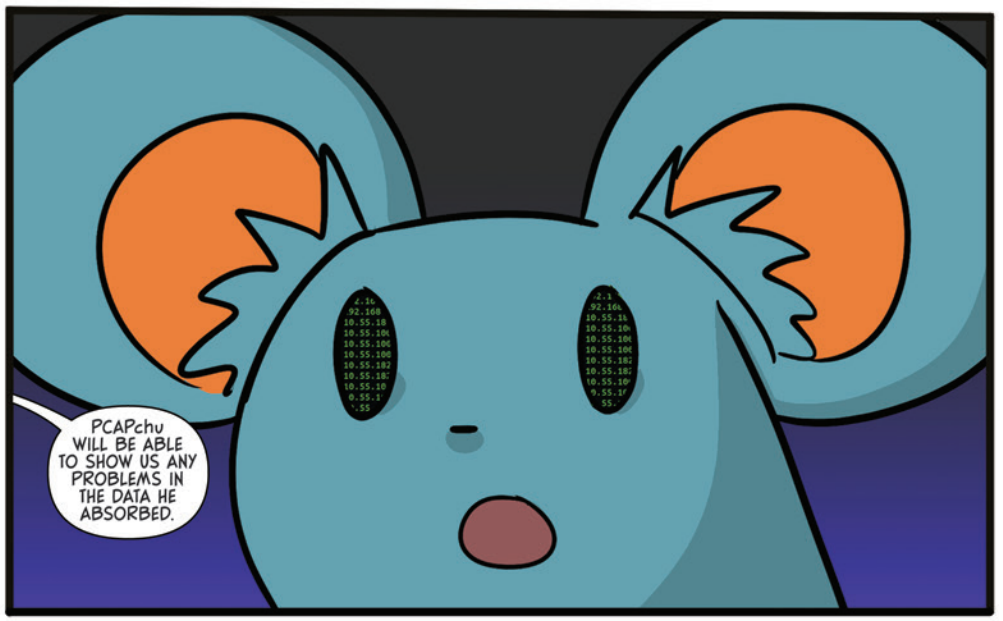
YOU DIDN'T DOWNLOAD ANYTHING OUT OF THE ORDINARY, DID YOU?

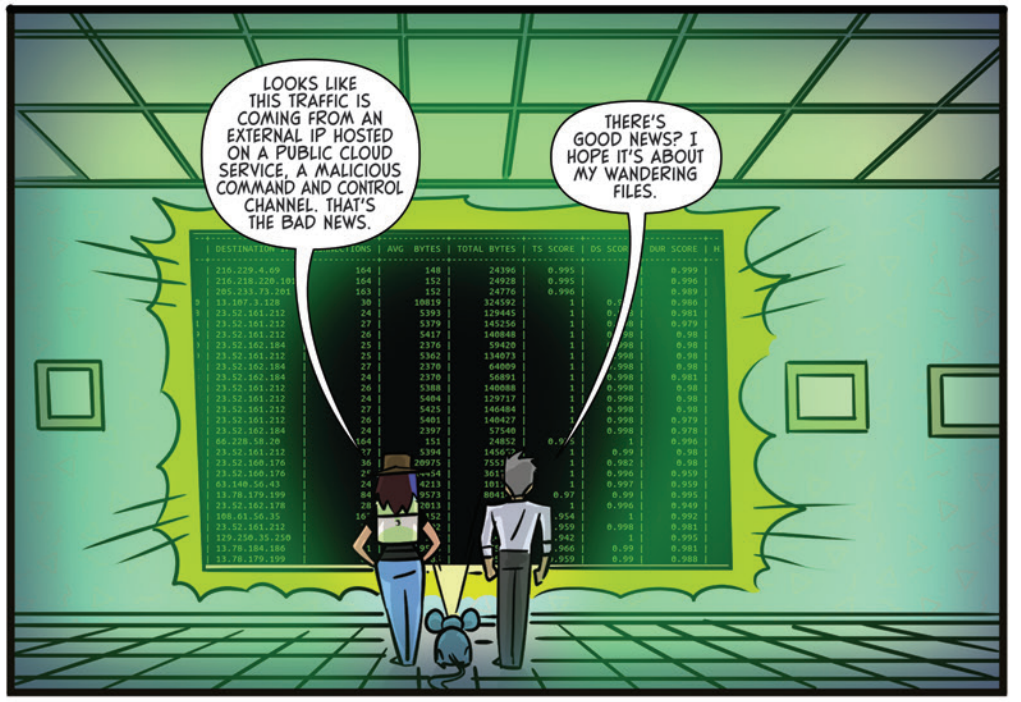
NOT ON PURPOSE...I DID CLICK ON A LINK IN A FISHY-LOOKING EMAIL THOUGH.

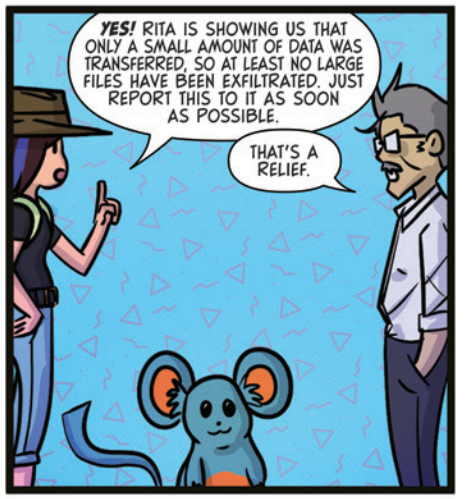

MORE LIKE "PHISHY"...


TIME TO INTRODUCE YOU TO ANOTHER FRIEND OF MINE-- *RITA.*


RITA IS AN OPEN-SOURCE THREAT HUNTING SOFTWARE THAT CAN DETECT WHERE ALL THIS TRAFFIC IS COMING FROM.

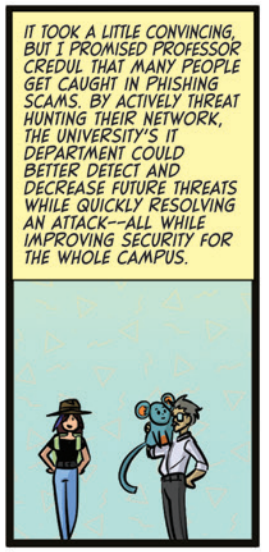

PCAPchu WILL BE ABLE TO SHOW US ANY PROBLEMS IN THE DATA HE ABSORBED.

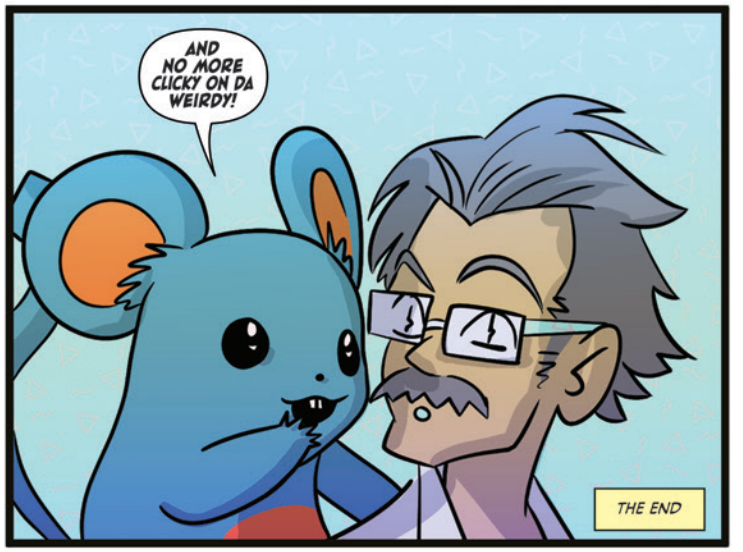

LOOKS LIKE THIS TRAFFIC IS COMING FROM AN EXTERNAL IP HOSTED ON A PUBLIC CLOUD SERVICE, A MALICIOUS COMMAND AND CONTROL CHANNEL. THAT'S THE BAD NEWS.
THERE'S GOOD NEWS? I HOPE IT'S ABOUT MY WANDERING FILES.


YES! RITA IS SHOWING US THAT ONLY A SMALL AMOUNT OF DATA WAS TRANSFERRED, SO AT LEAST NO LARGE FILES HAVE BEEN EXFILTRATED. JUST REPORT THIS TO IT AS SOON AS POSSIBLE.
THAT'S A RELIEF.


NOW TO TELL THE TECHIES THAT ONE OF THE UNIVERSITY'S TENURED PROFESSORS FELL FOR A PHISHING SCAM.


PROFESSOR CREDUL, IT HAPPENS TO THE BEST--AND BRIGHTEST! A LOT OF THESE EMAILS ARE REALLY, REALLY CONVINCING.


IT TOOK A LITTLE CONVINCING, BUT I PROMISED PROFESSOR CREDUL THAT MANY PEOPLE GET CAUGHT IN PHISHING SCAMS. BY ACTIVELY THREAT HUNTING THEIR NETWORK, THE UNIVERSITY'S IT DEPARTMENT COULD BETTER DETECT AND DECREASE FUTURE THREATS WHILE QUICKLY RESOLVING AN ATTACK--ALL WHILE IMPROVING SECURITY FOR THE WHOLE CAMPUS.


AND NO MORE CLICKY ON DA WEIRDY!
THE END

you're a **Threat Hunter** too

At my high school reunion, we talked about what we all do for work these days.

Gail ran a restaurant when she left school but is now a health inspector for the Vermont Department of Health.

Steve is a night watchman for a group of warehouses in New Jersey.

Lynn became a bridge inspector.

Karl spent most of his career as a pilot and now performs plane inspections for a major airline.

April works in a hospital lab, looking for disease in blood samples.

Allen works for the USDA as a food inspector, performing regular checks at food production facilities.

Me? I'm a network threat hunter. Sometimes, that takes a while to explain (*cough* every holiday *cough*), but to these six friends, it made sense.

We each spend our time actively and regularly looking for threats of one kind or another. We have scheduled inspections, we look for signs of trouble, and, when we find one, hand them off to someone who can investigate further to fix the issue. The goal is to find issues early, before they become real problems.

If we just waited for an alert to go off, it might already be too late to fix.

If you go for regular checkups with your doctor or have your car inspected every year, you're a threat hunter too!

-- Bill Stearns

CREDITS

Publisher: John Strand
~~Chaos~~ Excitement Creator: Jason Blanchard
Editor-n-Chief: Kassie Kimball
Creative Director & Puzzle Maker: Caitlin Cash
Production Coordinator & Floof Curator: Shelby Perry
Associate Editor: Andréa Culling
Proofreader & Morale Officer: Deb Wigley

Collaborators:
Chris Brenton
Bill Stearns
Liza Tsibur
Logan Lembke
Ethan Robish
Naomi Goddard

In Order of Appearance:

Cat with Toy: Sammy
Kayaking Husky: Aspen
Sleepy Cat: Kissima aka "Hissy"
Bandana Dog: Kobe
Long Eared Floof: Lexi
Excited to go for a Walk: Denver
Corn Cob: Himself
Turkey: Tom
Hungry Fox: Requested to remain anonymous for legal reasons
Lurking Cat: Sharky
Black Void: Luna
Upside Down Cat: Misty

Special Thanks:
to the team at Active Countermeasures and YOU, our incredible community!

and a warm, happy welcome to new baby Ezra Kimball: Junior Editor (once you learn to read in a few years)

ABC
DEF
GHI
JKLM

NOP
QRS
TUV
WXYZ

Subscribe to PROMPT# for free!

BLACK HILLS Information Security | 115 West Hudson St. Spearfish, SD 57783 | Contact Us: 701-484-BHIS or info@promptzine.com
PRINTED IN CANADA. POLYBAGGED IN CANADA. INSERTS PRINTED IN USA.