Github for project:

<https://github.com/1dentified/mdt_hackathon>

Collection of little known logs for windows firewalls and http activity.

**Team:** 601 AOC MDT from Tyndall (Not the 125th MDT)

Aaron Rosenmund

Devin Scott

Michael Esquivias

Colton Card

All scripts, generated data, cloud environments, config files, and templates were completely made from scratch for this hackathon.

**Mission Statement(Skip if TL):**

This project was born out of the convergence of two separate concerns. Mission Defesne Teams often are not fully connected to a network. When they are they are still missing data from the endpoints. Sometimes these endpoints are standalone for maintenance systems or fuel monitoring, or even in special access rooms. A capability to collect data from an endpoint (Windows) pull that off the machine, enrich that data and ingest into an elastic stack, does not require an installation of an agent or elevated admin permissions on the domain.

The second problem is that the adversaries that even non advanced adversaries have advanced to the point where they can hid from common detection methods. In my own experience with ransomware, in a live knife fight scenario, common logs have been wiped, and the majority of the device is encrypted. Most evidence is destroyed. But there are a few areas that are left untouched.

These are the data sources that provide information about who or what has been talk to these devices, without the need for having capture on the network, further, they provide information that is commonly only at layer 2, a point where most mdt’s are not capturing data. We can find machine to machine communication that indicates the enumeration, exploitation, or lateral movement of automated malware or live actors within a small network we would not otherwise see by analyzing the following resources:

**Currently Integrated Sources:**

**Firewall –** static log file in txt

%systemroot%\system32\LogFiles\Firewall\Firewall.log

**Additional Sources for Mapping Internal Malicious Internal Activity – Future Intergations**

**HTTPerr -**

%systemroot%\system32\LogFiles\Firewall\http\httperr.log

WinEvents Not Support by Winlog Beats and PS Pulls

**SMB -**

Get-smbconnection | outfile current\_smbconnections

[system.io.directory]::getfiles("\\.\\pipe\\")

Microsoft-Windows-SMBClient/Connectivity

Microsoft-Windows-SMBServer/Connectivity

**RDP/Terminal Services -**

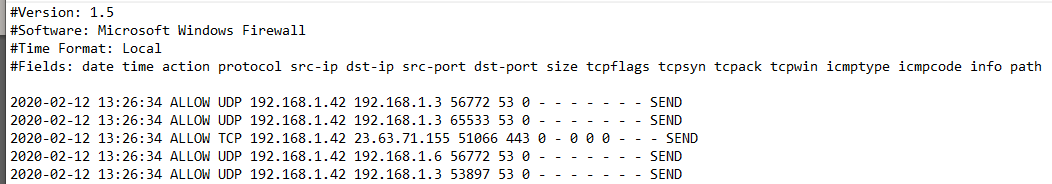
Microsoft-Windows-

Microsoft-Windows-

These sources are not natively supported, are nearly never encrypted in ransomware type attacks and are not wiped stealthy adversaries because they are not part of the standard windows event logging system.

And they have all kinds of usable data!

**Example Firewall.log**



These clearly need to be parsed into fields and enriched, and that is exactly what this PowerShell script does adding the following data:  
Enrichment

* Reverse DNS Lookup
* Add Field for external source IP
* Internal Computer Name

And then saving the results in a .csv file.

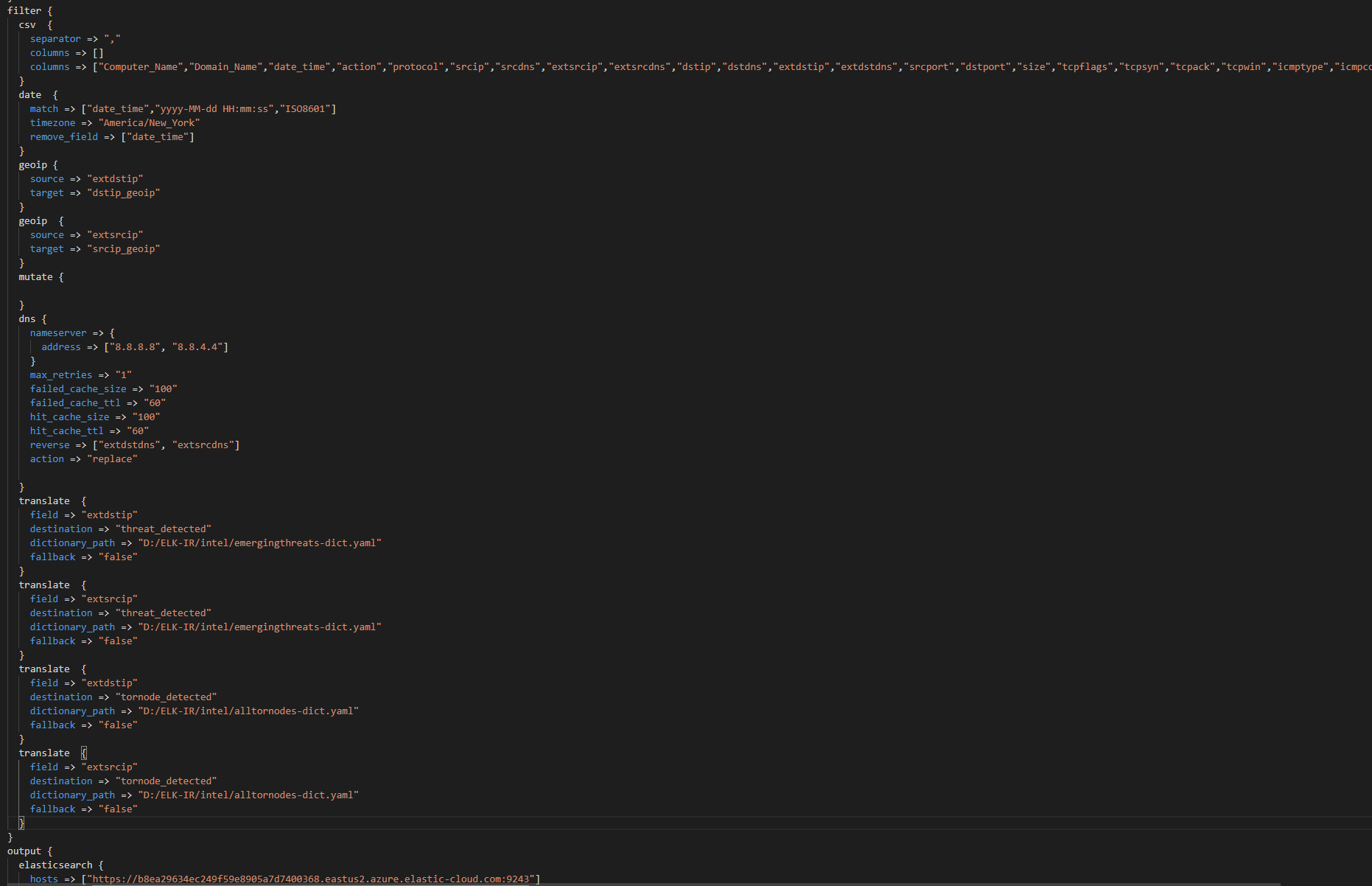
**Enter PowerShell**

Leveraging PowerShell, the script ingests the logs, parses them into a PowerShell object, enriches the data with local DNS cache information, computer name, domain name, and external IP address and then process each record with the additional data into a csv.



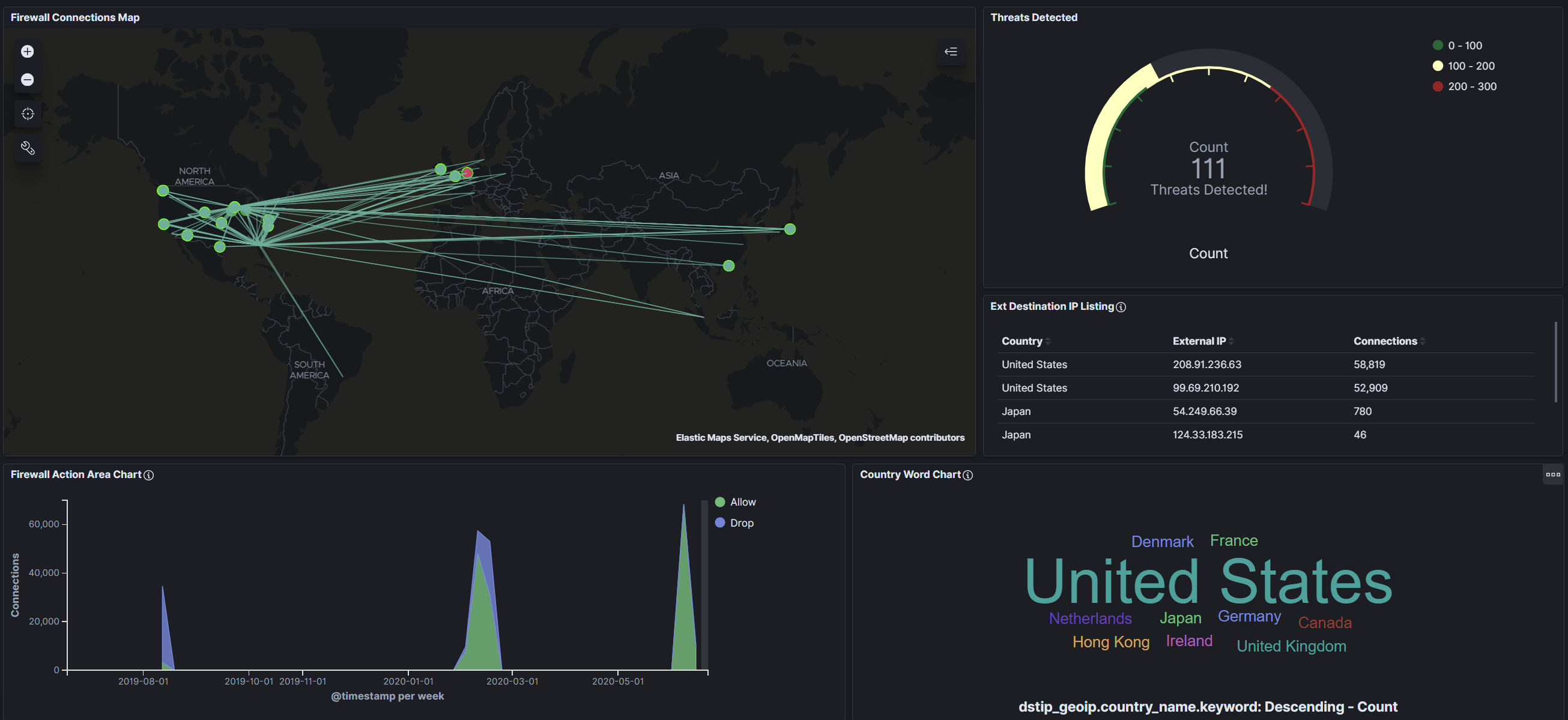
This csv is then read with this custom **logstash.conf** file that performs the following enrichments:

* Revers DNS Resolution
* External Source IP GEOIP Lookup
* External Destination IP GEOIP Lookup
* Date converted to event @timestamp
* Both Source and Destination IP’s checked against emerging threat node list
* Both Source and Destination IP’s checked against emerging threat node list



This is fed into a cloud instance we stood up for this challenge with a custom index template to accurately classify each field for the best analysis. **Index Template 220 lines (full doc on github)**

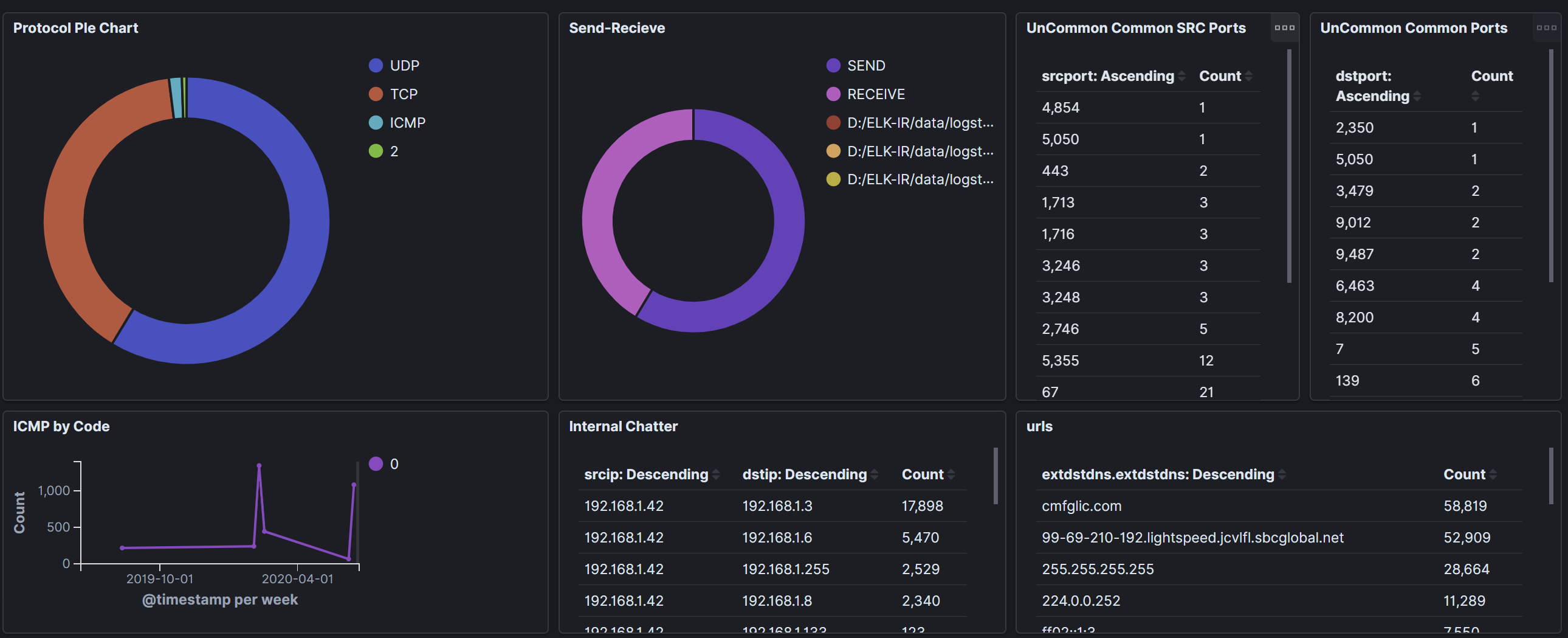
Now all of the hard work pays off. With the content just from this firewall log, transformed to maintain internal and external ip data. We can generate the following analysis:



* Connections weighted by count and colored by direction, with the icons colored by weather it was a detected threat.
* Number of threats identified and on what connections
* Number of tor nodes identified in external connections
* Country connections by top and least talkers
* Dropped vs allowed connections

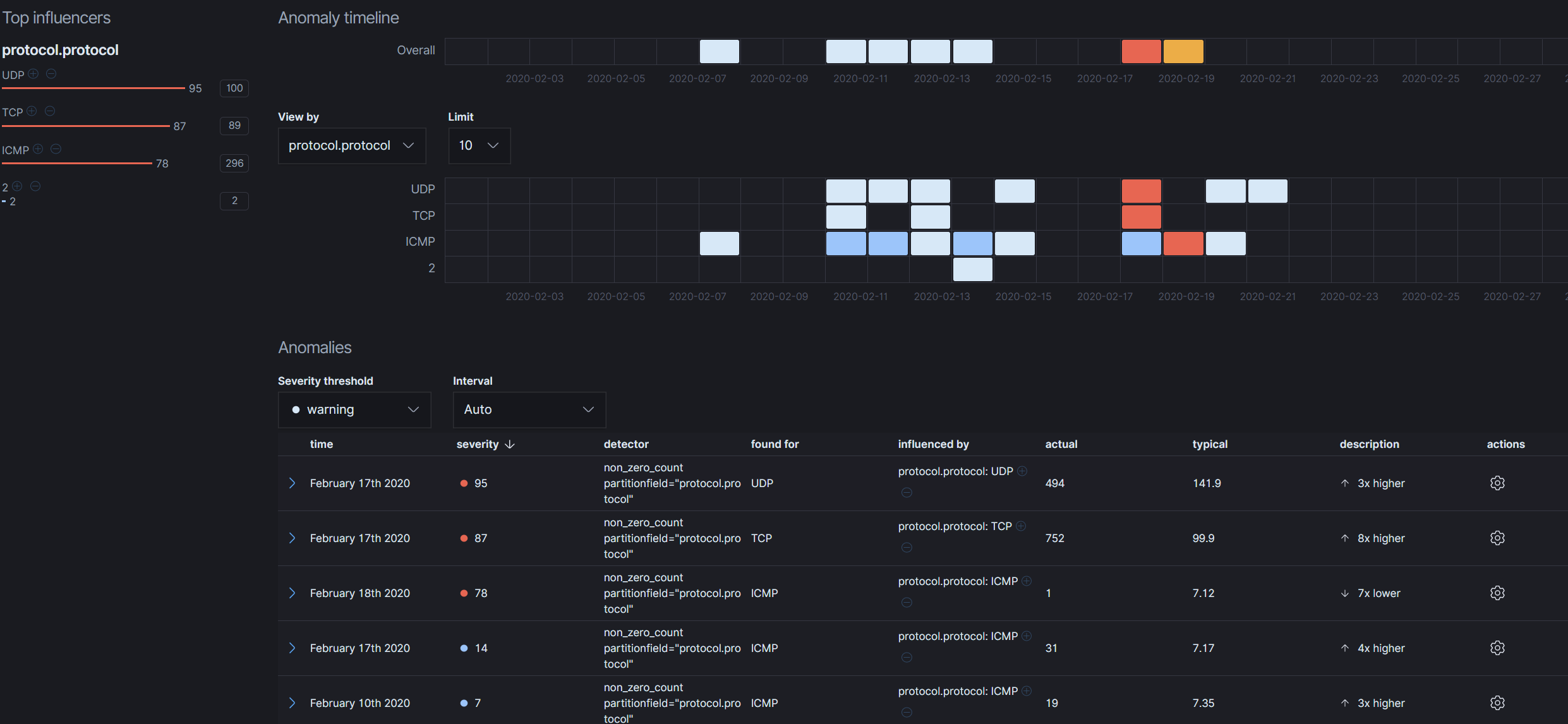
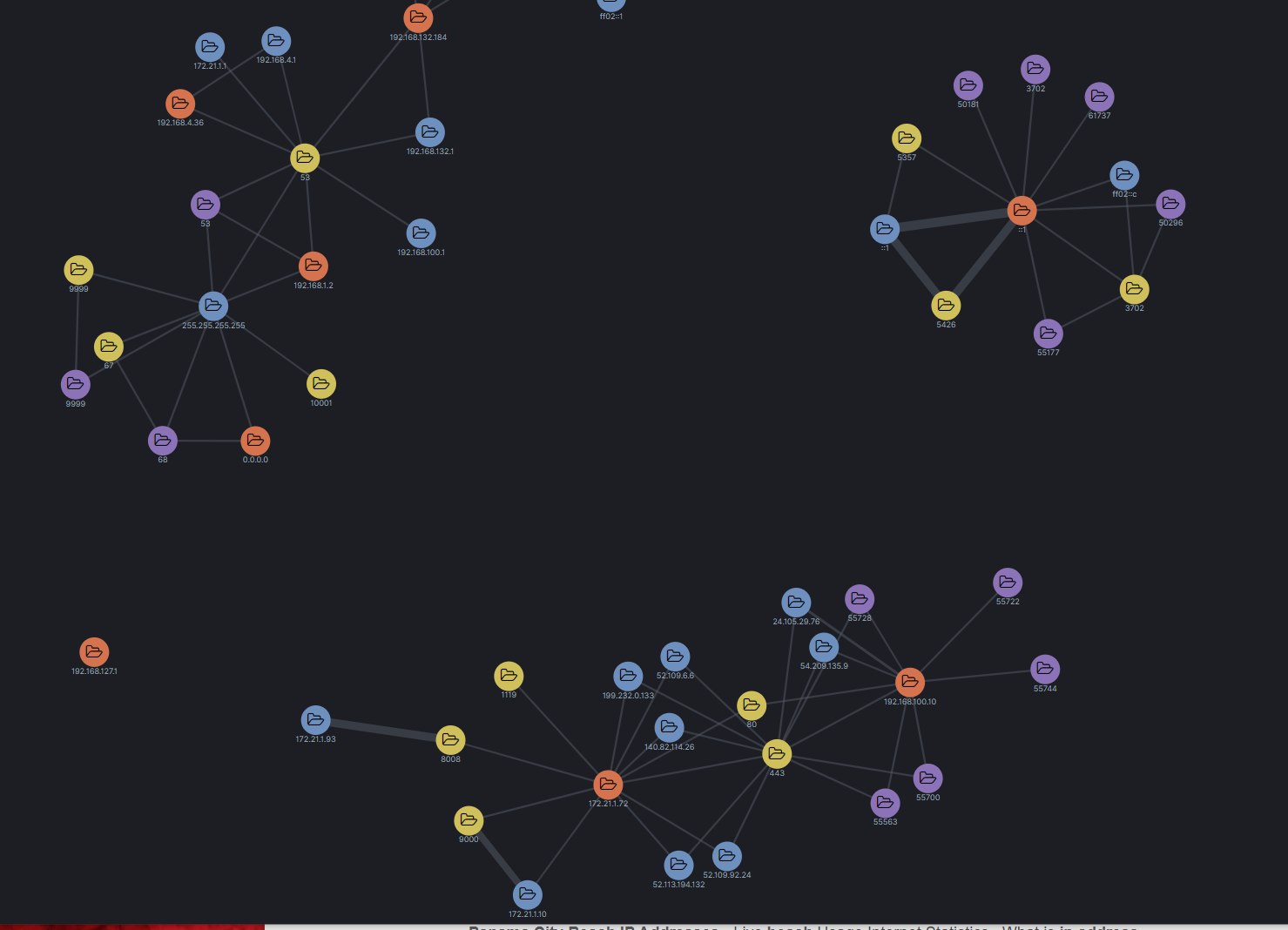
*And much more than can fit on this one dashboard.*

**Or this one:**



* Graph analysis of internal ip connections by port to map out internal network, notice the firewall pulled a number of broadcast traffic as well.
* Attempted connections by protocol type 2 is interesting because it shows up with communications to routers probably stand for layer 2.
* Filter by internal network and get all ip 5 tuple date for device to device communication you would not see form a layer 3 interface tap
* DNS translation gives up top URLs visited or bottom
* ML single metric analysis can alert you on potential for scans or odd behavior based on protocol type.

Or these ML and Graph queries that are not displayed in dashboards!



It goes on and on, the ability provided from this one data source once enriched needs to continue to be explored but is significant for the lack of required access to the inspected device or even to the internet which is a known must for many MDT’s.

We should no longer leave an environment uncontested because of our restrictions on connected hardware, logical access permissions, or the varied nature of available information. We are intelligent Airmen who can adapt to the information available, transform it as necessary, and engage the enemy whenever or wherever they may be in cyberspace. ~ Aaron Rosenmund

**Thank you for your consideration.**

**References:**

All files used are on the github repo mentioned above.

Intel Feeds from MISP:

<https://www.misp-project.org/feeds/>

Conversion of

