DKCERT

DeiC

# pDNSSOC

## Leveraging MISP indicators via a pDNS-based infrastructure as a poor man's SOC


Presented by: Panagiotis Matamis



DKCERT

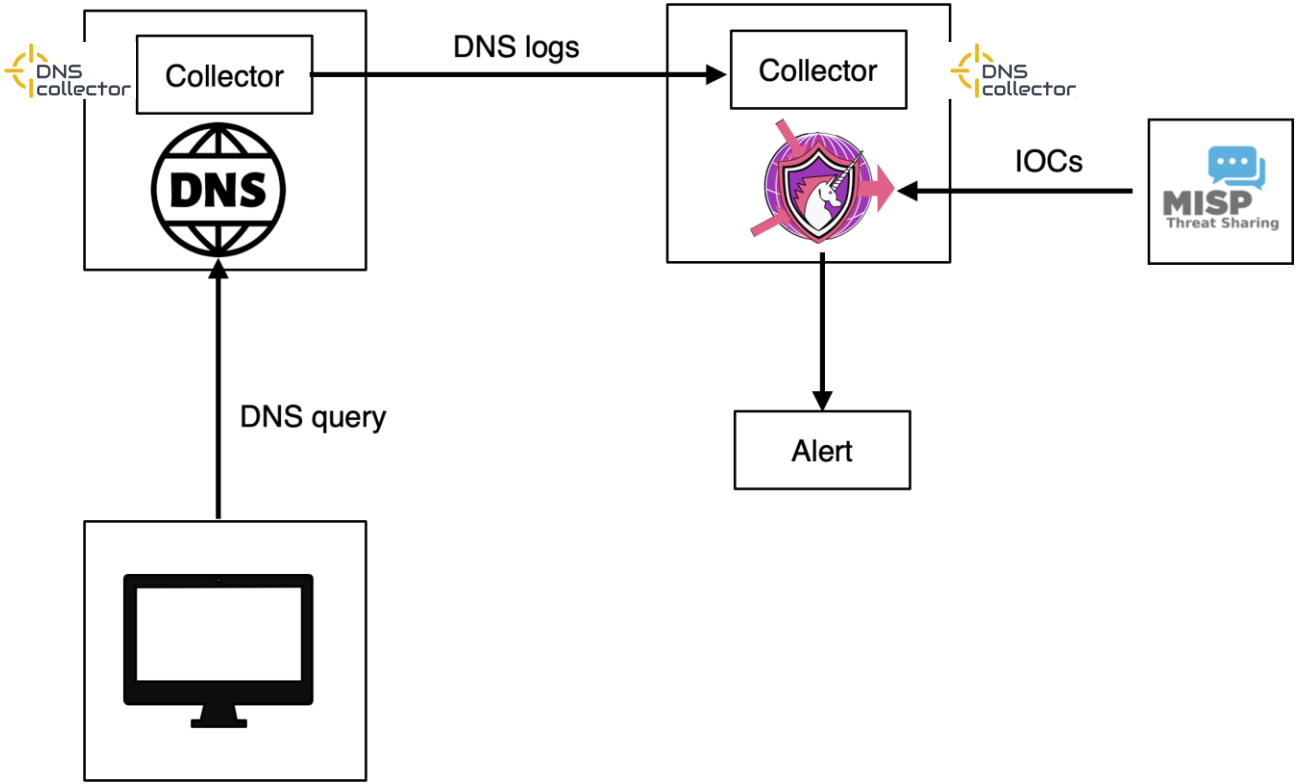
# pDNSSOC: Who is interested so far?

Red CLARA

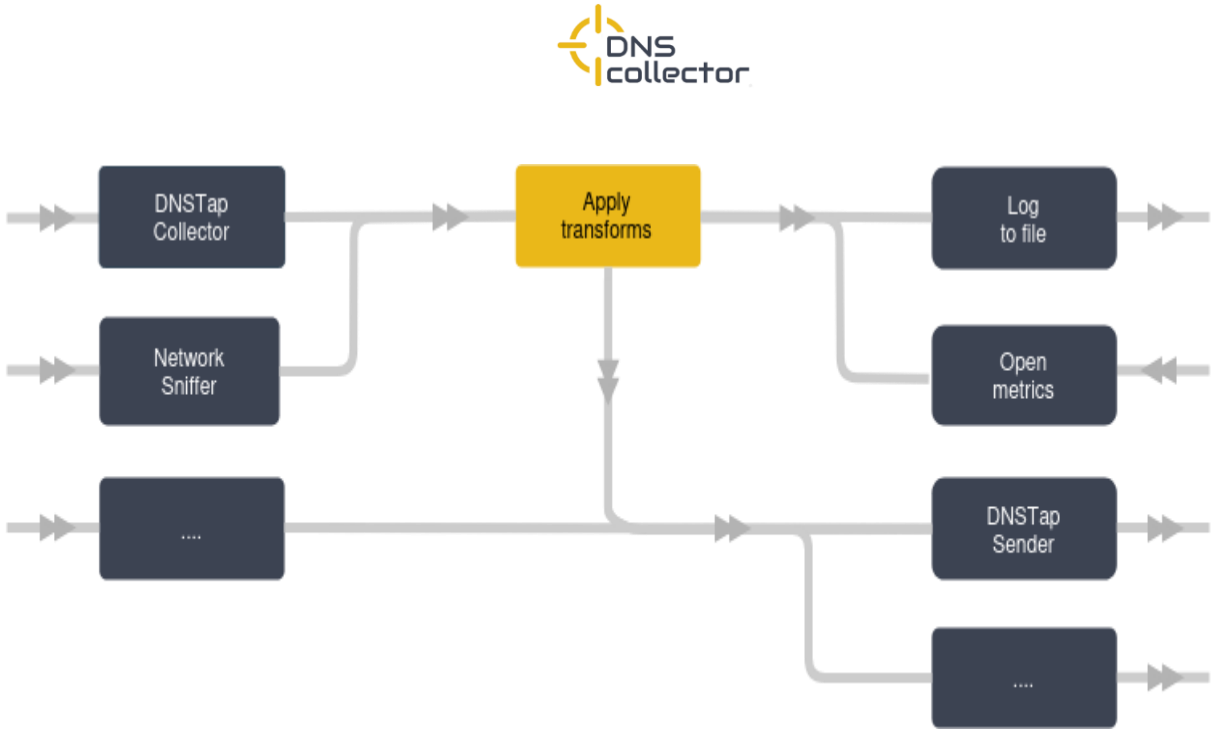
project **SICURA-LAC**

- Regional Network Provider RedClara
- National Research and Education Networks Mexico: CUDI, Ecuador: CEDIA, Guatemala: USAC/RAGIE, Chile: REUNA, Uruguay: RAU, Costa Rica: CONARE, Colombia: RENATA, Panama: Network in formation led by SENACYT, Argentina: Activities led by ARIU
- Global Collaborators
- SAFER
- Various organizations in the EU (hospitals in France and Switzerland for example)
- Various organizations in the US (ESNet, FNAL have shown interest).

# pDNSSOC: Overview



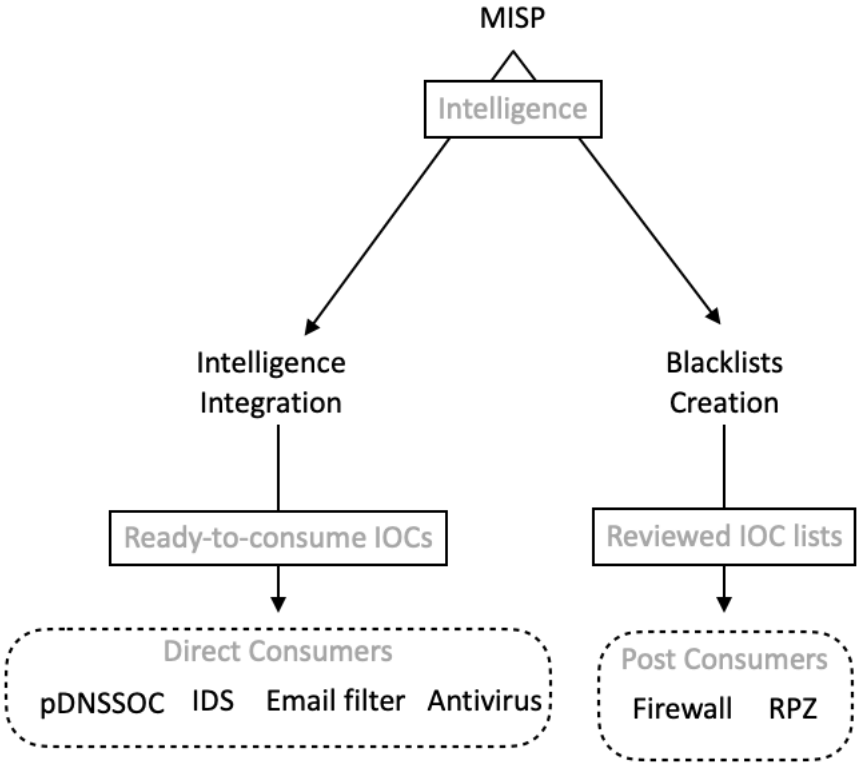
# pDNSSOC: leveraging dnscollector + MISP




## Transformers

- Custom Relabeling for JSON structure
- Add additionnal Tags
- **Traffic Filtering and Reducer**
- Latency Computing
- **Apply User Privacy**
- Normalize DNS messages
- Add Geographical metadata
- Various data Extractor
- Suspicious traffic Detector and Prediction

# pDNSSOC: leveraging dnscollector + MISP




MISP Threat Sharing

Has nice features like warning lists and ioc decaying models in order to reduce false positives.

# pDNSSOC: privacy concerns

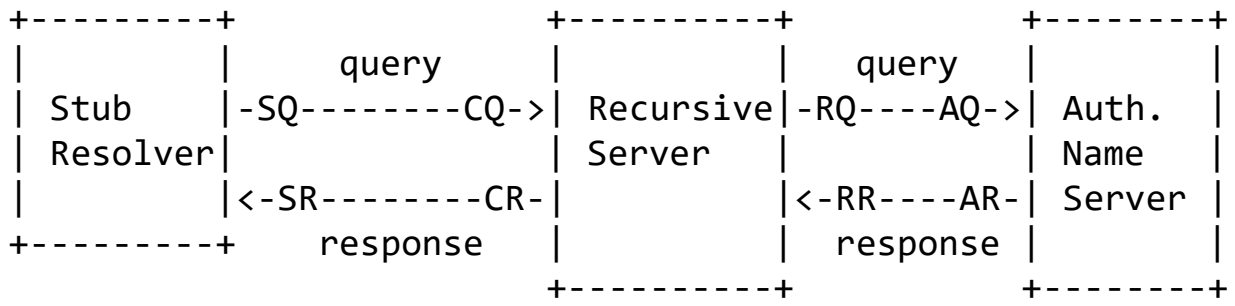
Log only resolver responses

DNSTAP

```
// dnstap config (BIND)
dnstap {
      resolver response;
};
```

dnscollector

```
transforms:
  filtering:
    keep-queryip-file: "recursors.txt"
    log-queries: false
    log-replies: true
```



# pDNSSOC: privacy concerns

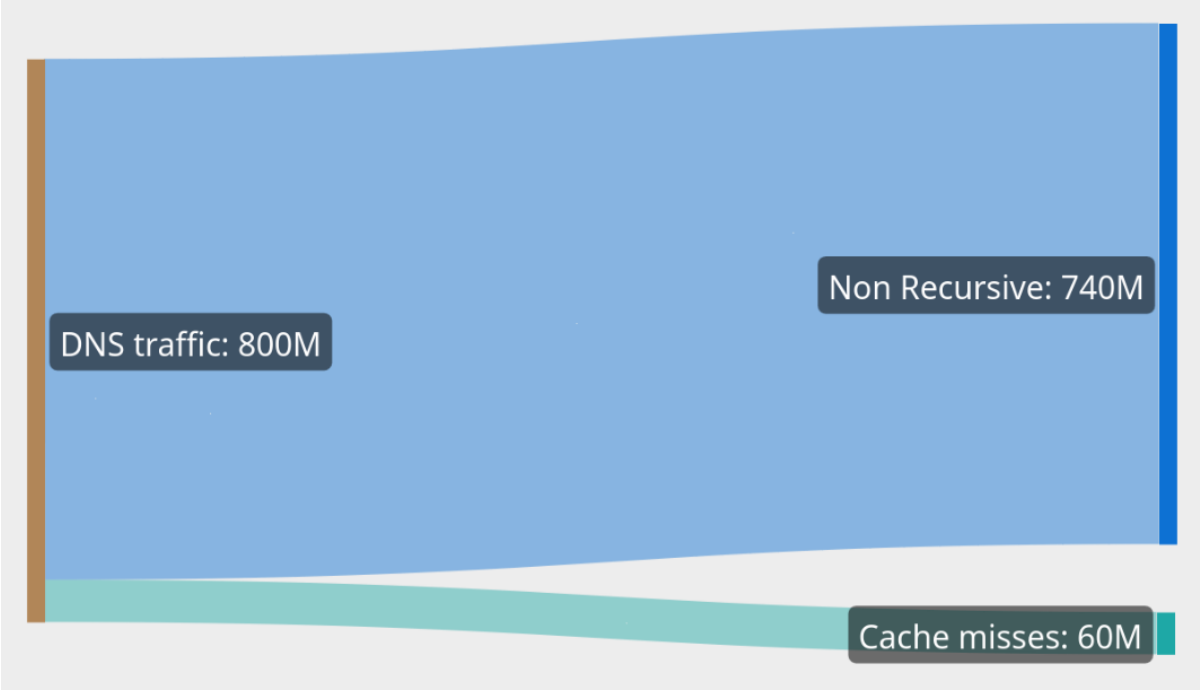
Mask client IP

```
{
    …
    "query-ip": "188.184.*.*",
    …
}
{
    …
    "query-ip": "40307c253772c29ca7fb…",
    …
}
{
    …
    "sensor_id": "aaba123afd74…",
    …
}
```

```
transforms:
  user-privacy:
    anonymize-ip: true
    anonymize-v4bits: "/16"
    anonymize-v6bits: "::/64"
    hash-ip: false
    hash-ip-algo: "sha256"
    minimaze-qname: false
```

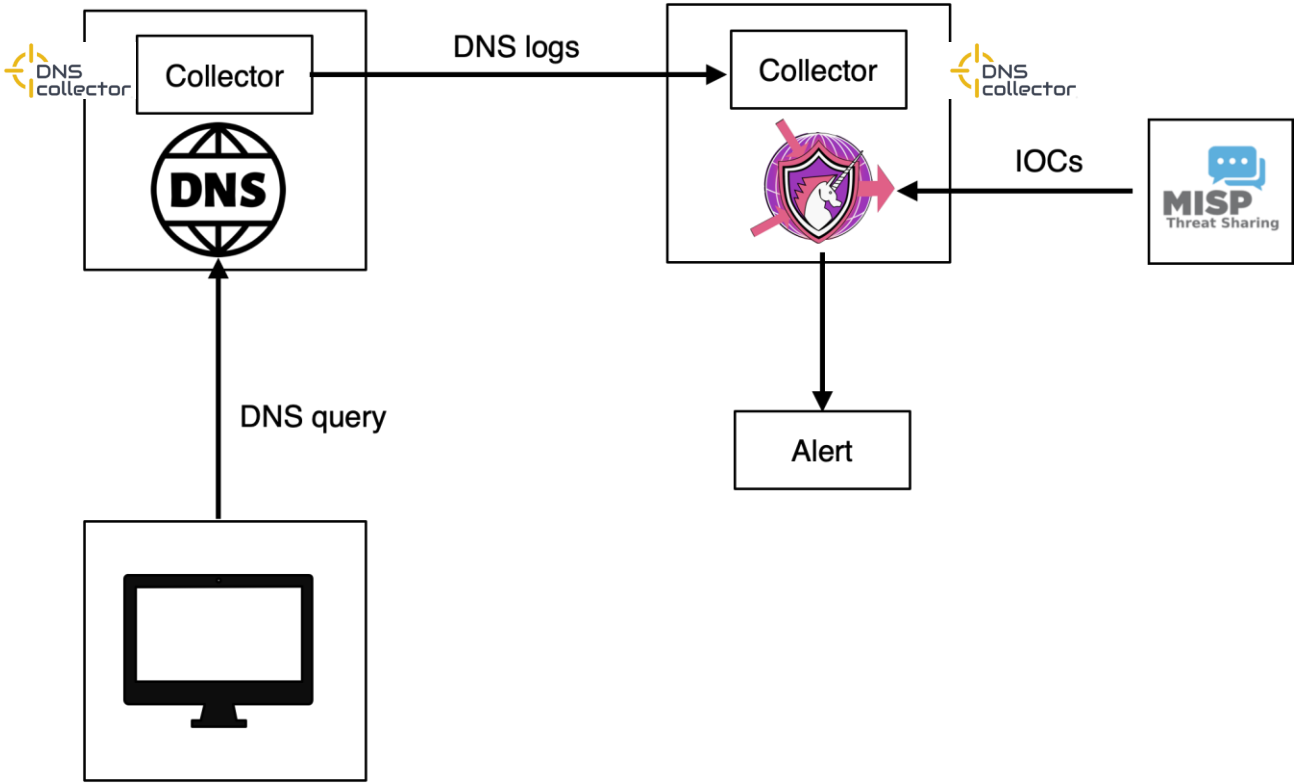
DKCERT

# pDNSSOC: Data size

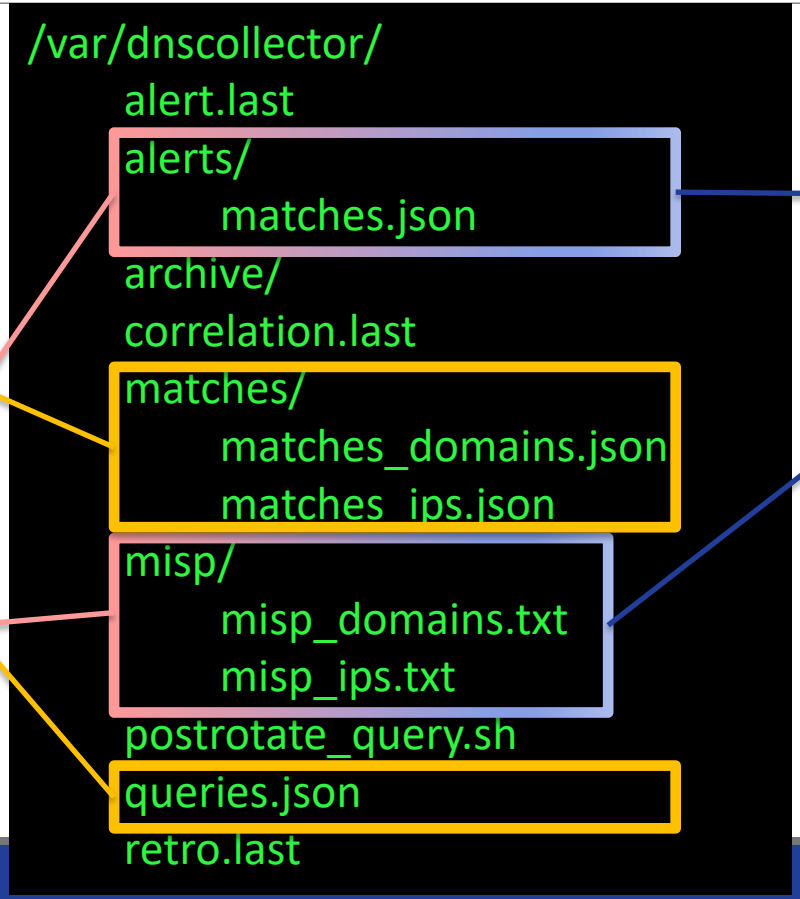



- Processing only cache misses results in significantly less data.
- At CERN, cache misses account for 5% of the total DNS traffic

# pDNSSOC: Overview



# pDNSSOC: Important files




DNS collector

MISP Threat Sharing

pDNSSOC

pdnsscoc-cli:

- fetch-iocs
- correlate
- alert
- daemonize

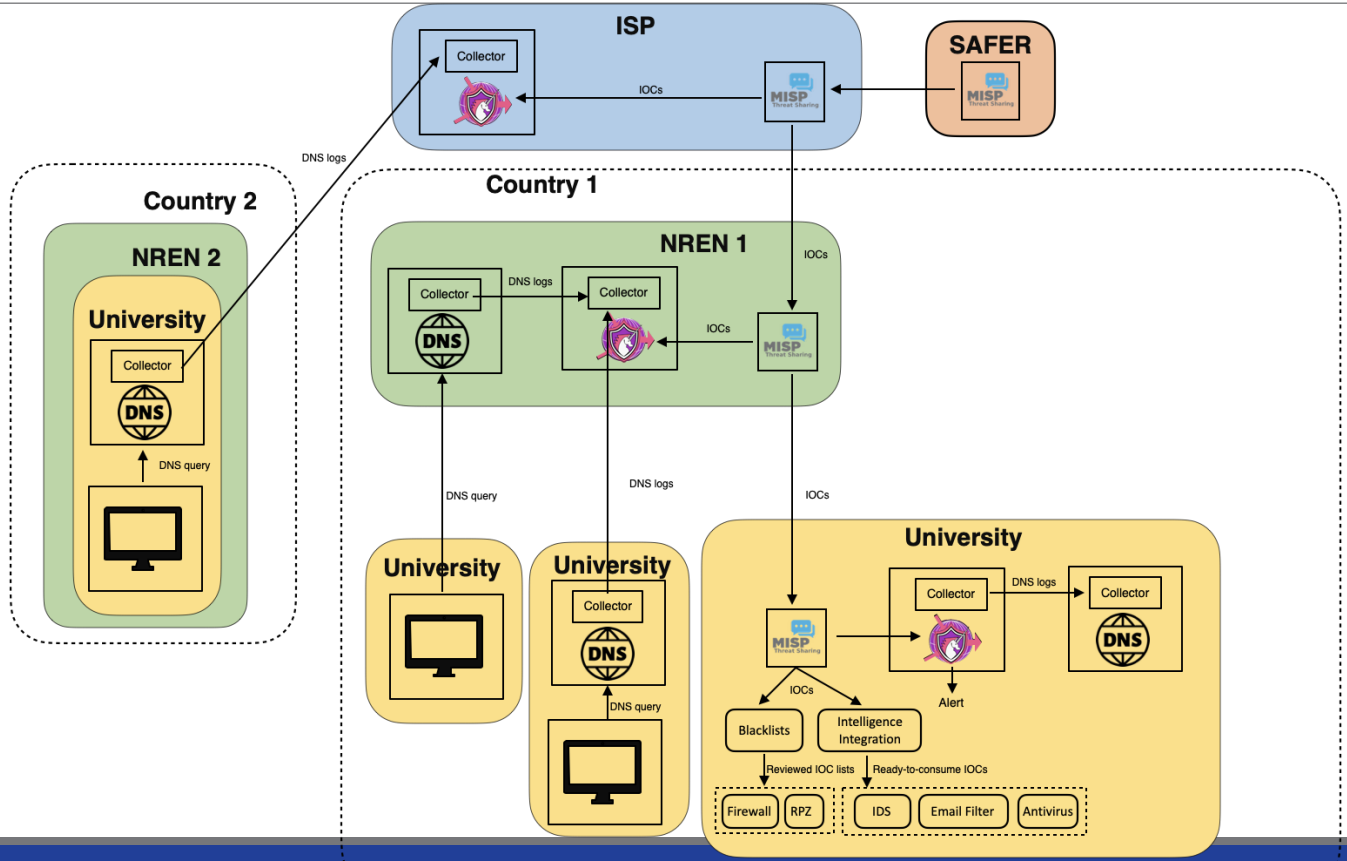
# Time for a DEMO!

3 virtual machines running

- A **Bind DNS server** with DNSTAP logging enabled and go-dnscollector running
- A **MISP** instance
- A **pDNSSOC** server running pdnssoc-cli and go-dnscollector (pyMISP is used in order to communicate with MISP via the API)

DKCERT

# pDNSSOC - Overview



# > Thank you

Any questions??

pma@cert.dk

# Appendix – Steps taken during the demo

- ssh to the bind dns server
- sudo rndc stop or sudo systemctl stop named
- sudo -u bind fstrm_capture -t protobuf:dnstap.Dnstap -u /var/cache/bind/dnstap.sock -w /var/cache/bind/dnstap.fstrm &
- sudo -u bind go-dnscollector -config /etc/dnscollector/client.yml &
- sudo systemctl start named
- sudo systemctl status named
- ss | grep 192.168 (ss is like netstat and we grep for the ip of the server, if we see a connection on port 7001, a random port that I picked during configuration, we are good)

# Appendix – Steps taken during the demo (2)

On the pdnssoc server side:

# start the dnscollector and use ss and grep to double check if the connection was established on port 7001

- go-dnscollector -config /etc/dnscollector/config.yml &
- ss | grep 192.168.46

#open a browser, login to MISP and show the events and attributes with the ids flag on

# we fetch the iocs from MISP into /var/dnscollector/misp/misp_domains.txt

- pdnssoc-cli -c /etc/pdnssoc-cli/config.yml fetch-iocs

# 53 domains and 1 ips

- cat /var/dnscollector/misp/misp_domains.txt
- cat /var/dnscollector/misp/misp_ips.txt

# Appendix – Steps taken during the demo (3)

# show the file structure and explain what is happening
- ls -alR /var/dnscollector/

# /var/dnscollector/matches/match_domains.json is being updated by the dnscollector
# /var/dnscollector/alerts/matches.json is being updated by the pdnssoccli

# we do some dig commands with the @192.168.46.101
- dig amazon.eu @192.168.46.101
- dig amazon.eu -4 @192.168.46.101
- dig amazon.eu any @192.168.46.101

# we can also edit the file /etc/resolv.conf and remove all other nameservers
# now we can dig without specifying @192.168.46.101 and we can even use our browser
- dig amazon.de

# Appendix – Steps taken during the demo (4)

# run pdnssoc-cli and correlate the matches
# /var/dnscollector/alerts/matches.json will get updated with the new alerts
- pdnssoc-cli --config /etc/pdnssoc-cli/config.yml correlate --start-date 2024-04-04T00:00:00

# if we run the command without a start date, it will consult the file /var/dnscollector/correlation.last which contains a timestamp of the last correlation and continue from there
- pdnssoc-cli --config /etc/pdnssoc-cli/config.yml correlate

# if we run the alert command, an email will be sent with the alert
- pdnssoc-cli --config /etc/pdnssoc-cli/config.yml alert

# correlate creates the alerts to be dispatched
# alert dispatches the alert (via email)

# Appendix – Steps taken during the demo (5)

# show the difference between the /var/dnscollector/queries.json, ./matches/matches_domains.json and /alerts/matches.json
tail -f queries.json
tail -f matches_domains.json
tail -f matches.json
# show the masked query ip address
# show the alert log which includes a correlation (MISP) section

# Appendix – Steps taken during the demo (6)

# Actually, we don't need to run the correlation manually
# we can run the deamonize command or the supervisord and the previous commands will run automatically
# show them the supervisor.conf file

- cat supervisor.conf

# supervisord will run both the go-dnscollector and execute pdnssoc-cli commands at certain intervals.
# The pdnssoc-cli configuration file has a schedules section where you can specify how often you will run the fetch-iocs, correlate and alert commands. An example with greatly reduced intervals for testing reasons is shown on the right.
# run supervisord - everything happens automatically now

- supervisord -c supervisor.conf

```
schedules:
  fetch_iocs:
    interval: 5 # minutes
  correlation:
    interval: 1 # minutes
  retro:
    interval: 1440 # 1440 min is 24h
  alerting:
    interval: 7 # minutes
```

# Appendix – Steps taken during the demo (7)

# mention that older logs (anything inside the archive folder) will be scanned as well at a certain interval and matches not previously found would be detected as well
# mention that warning lists are taken into consideration and IOCs which are part of a warning list aren't fetched.
# mention that there is an option in the configuration file of pdnssoc-cli that can ignore IOCs older than a certain time (30 days is the default value).
# mention that pdnssoc wasn't meant for blocking but only detecting and talk a little bit about rpz and show an example
# theoretically the same iocs could be included inside a DNS response policy zone (rpz) and be blocked next time

- cat /etc/bind/db.rpz.local  (to demonstrate a sample rpz list on the dns server)
- dig websiteincludedin.db.rpz.local  (to see the different response and that access is denied)

# References

https://github.com/CERN-CERT/pDNSSOC
https://github.com/CERN-CERT/pdnssoc-cli
https://github.com/dmachard/go-dnscollector/blob/main/README.md
https://www.misp-project.org/
https://www.circl.lu/doc/misp/warninglists/
https://www.misp-project.org/misp-training/a.5-decaying-indicators.pdf
http://www.cern.ch/security
https://codimd.web.cern.ch/s/K80tX5NuP
https://safer-trust.org/about/