REDOCS 2024 - AMOSSYS

Automating dataset labelling

Team

Rémi BOUCHAYER remi.bouchayer@ip-paris.fr
KNDS France / SAMOVAR, Télécom SudParis / LyRIDS, ECE Paris

Armand Florent TSAFACK PIUGIE armand-florent.tsafack-piugie@unicaen.fr
UNICaen, Greyc, Safe, téïcée

Quentin JAYET quentin.jayet@cea.fr
CEA GRENOBLE

Chris VERSCHELDEN chris.verschelden@proton.me
IRIT - ARGOS

Usman ISAH usman.isah@insa-cvl.fr
INSA - CENTER VAL DE LOIRE

Outline

- Introduction
  - Amossys, M&NTIS platform
  - Objective
- Contributions
  1. Rule-based on system logs
  2. Anomaly detection for network traces
- Future works

A bit of context
the company, the tool

Amossys

Part of Almond

- CESTI
- Audit & Consulting
- Cybersecurity R&D

Deliver products and services

M&NTIS platform

A quick history

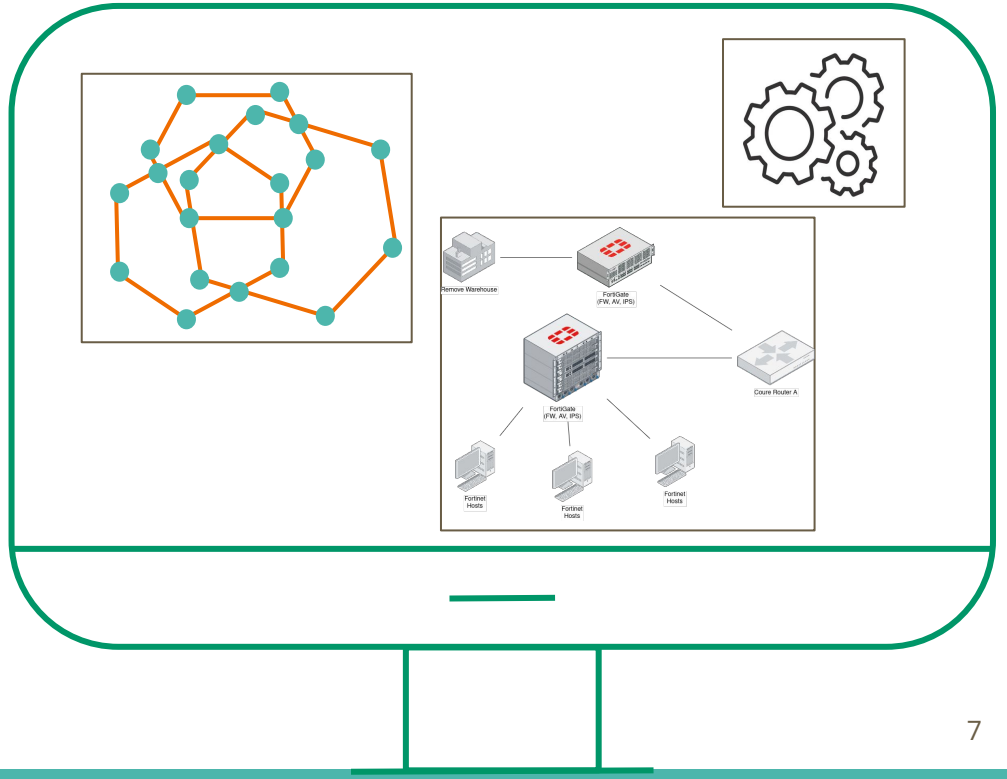
- Simulation environment
- Developed internally at first
  - attack sim
  - produce reports
  - ...
- Sold as a SaaS
  - Test your network virtually

M&NTIS platform

Virtual environment

- Run simulations
- Test tools
- Instantiate network topologies

Example: CyberRange from Airbus



M&NTIS platform

Blue team

- Extract simulated Logs

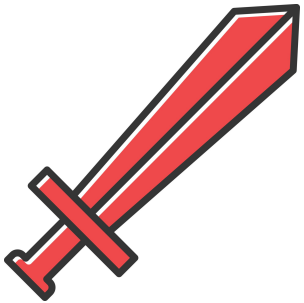
System logs (ECS)

Network logs (PACP)

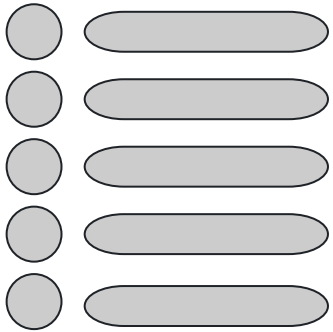
M&NTIS platform

Red team

- Perform an attack scenario



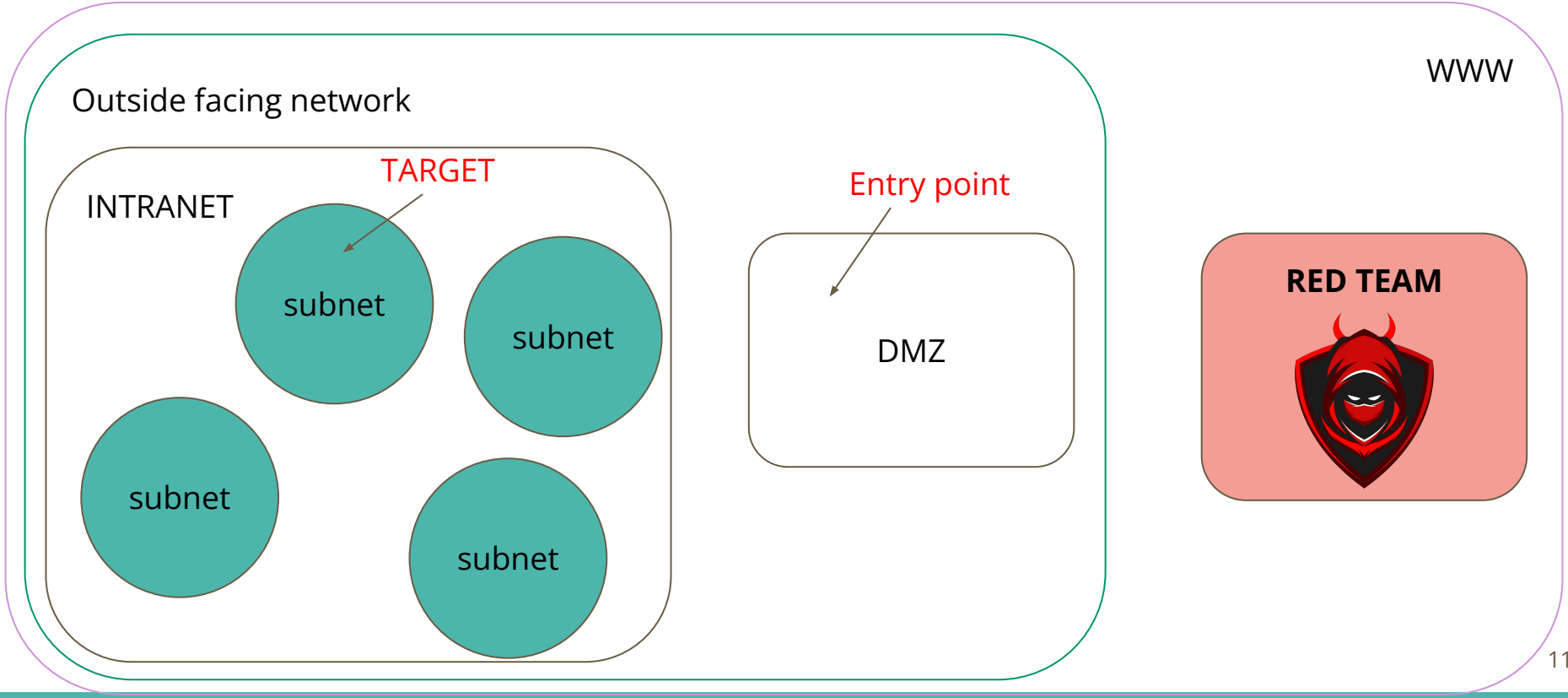
- Produce a step-by-step attack report



We know our environment
Time to run a simulation

Scenario Florama

Network structure

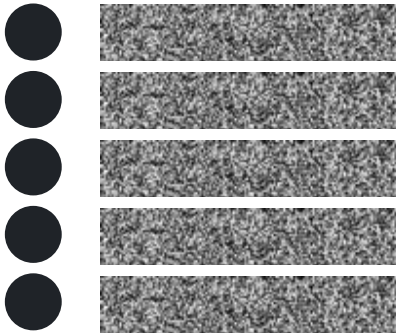


Scenario “Florama”

Benign

Benign scenarios

- No attack is performed
- Background activities
  - send/receive mail
  - browsing
  - etc...



Scenario “Florama”

Attack

Attack steps

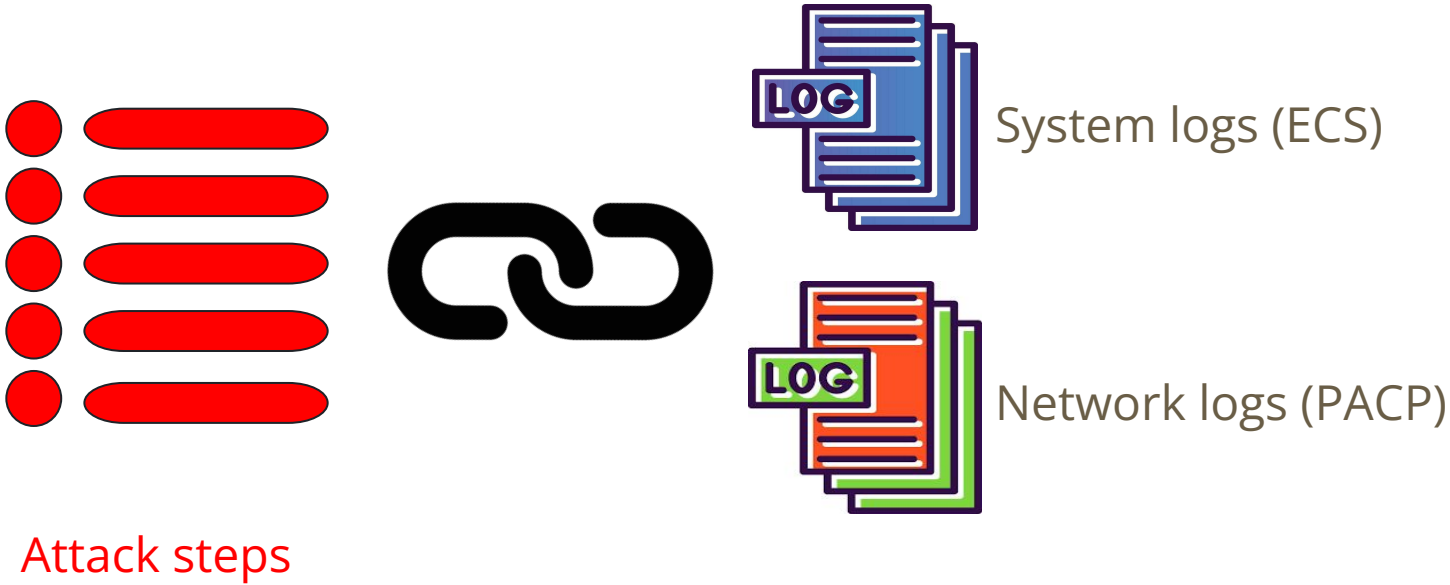
- Deploy attack infrastructure
- Compromise WebServer
- Reach proxy
- Find target and execute code remotely

Expected platform improvements

- Detection of attacks from logs
- Machine learning methods for detection
  - SIEM
  - XDR
  - Network probes

Our objective

- Correlate attack steps with logs



Contributions: develop a hybrid approach

1 - Rule-Based Detection

Exploit system logs (ECS)

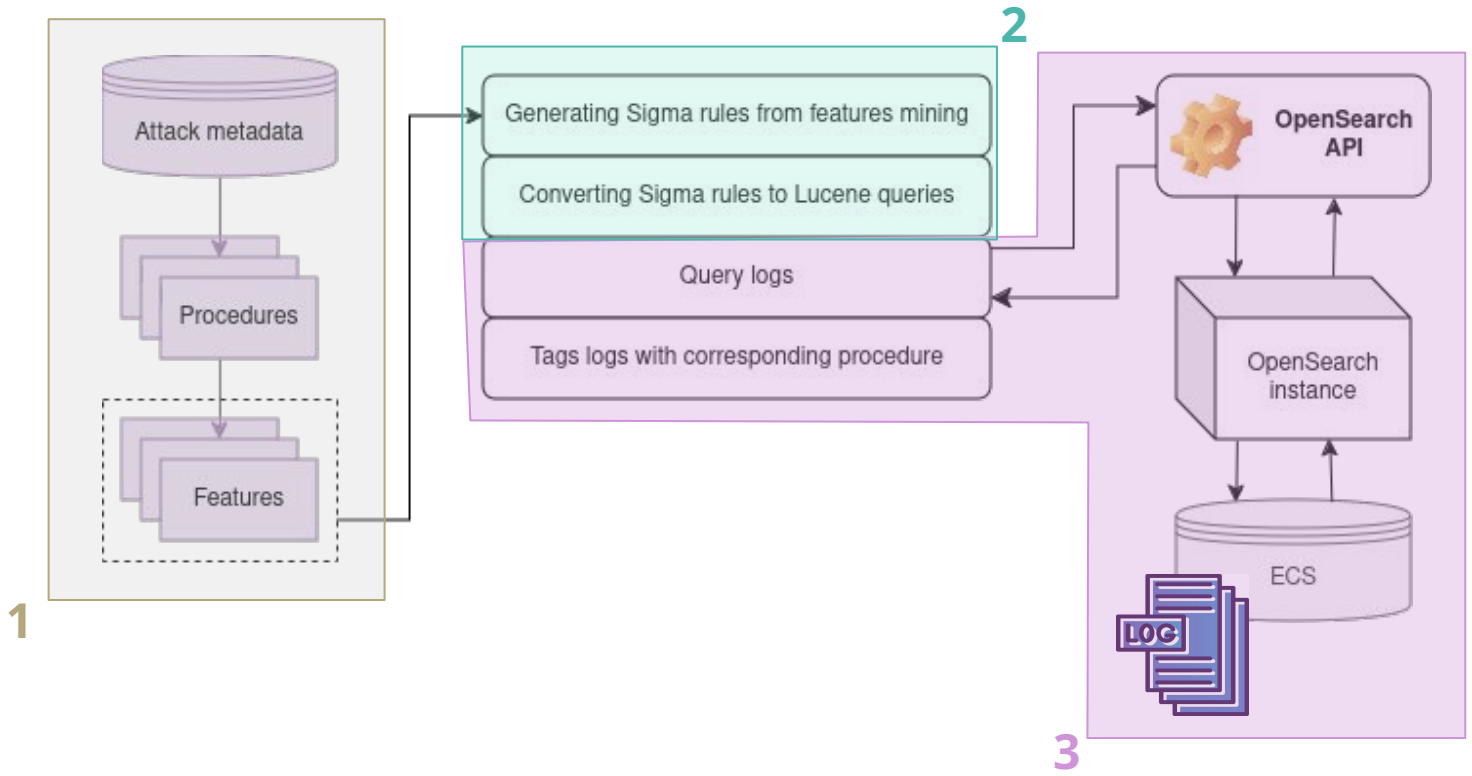
2 - Anomaly Detection

Exploit network logs (PACP)

System logs with rule-based approach

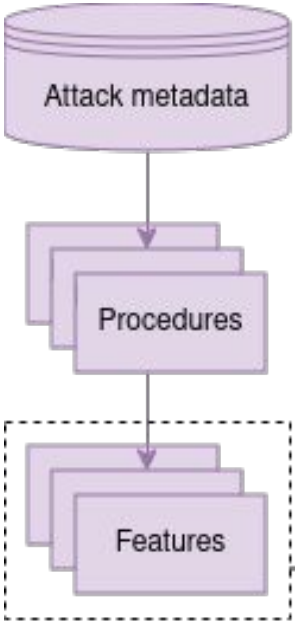
System logs

Detection workflow



System logs

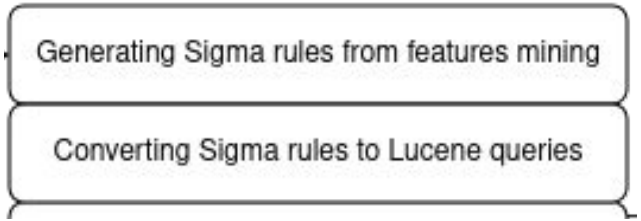
1. Parsing and extracting features



- Contains every attack procedures and steps
- Filtering out setup procedure
- payload : commands and sub-commands
- targets IP addresses
- Start & end timestamps

System logs

2. Detection workflow overview



- Sigma rules
  parsed commands

- Lucene query
  Sigma rule + remaining features

```
title: CVE-2024-1212 Exploitation
id: eafb8bd5
status: experimental
description: |
```

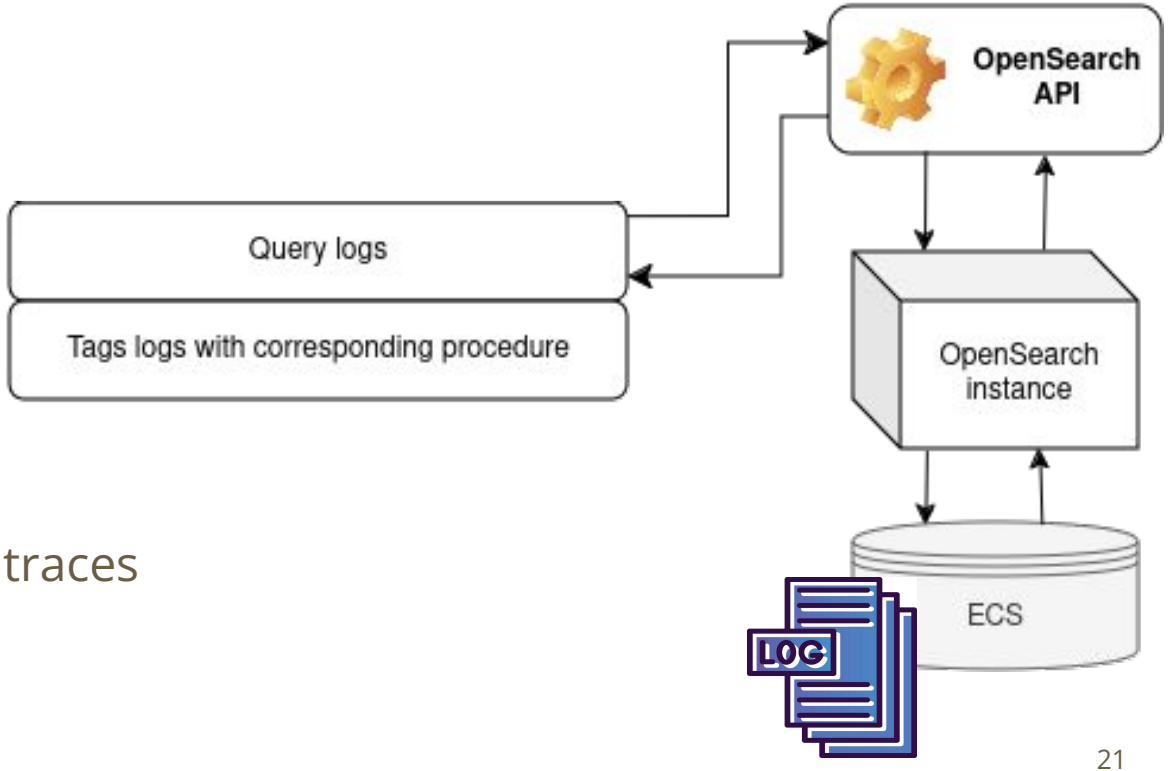
```
{
  "query": {
    "bool": {
      "must": [
        {
          "query_string": {
            "query": "process.args:chmod AND process.args:\\\\+x AND
              process.args:\\\\/tmp\\\\/nmap",
            "analyze_wildcard": "TRUE"
          }
        }
      ],
      "filter": [
        {
          "range": {
            "@timestamp": {
              "gte": "2024-09-05T06:54:11Z",
              "lte": "2024-09-05T06:54:22Z",
              "format": "strict_date_optional_time"
            }
          }
        }
      ]
    }
  }
}
```

```
    condition: all of selection_*
falsepositives:
    - Unlikely
level: high
```

System logs

3. Detection workflow overview

- Run a query
- Receive results
- Correlate results then label traces



System logs
Rule-based detection

Data format: JSON Elastic Common Schema (ECS) events

Parsing

- “Simple” *grep*
- JSON specific *jq*
- Sigma[1] rules log format agnostic (… with the good converter)

Based on the attack report

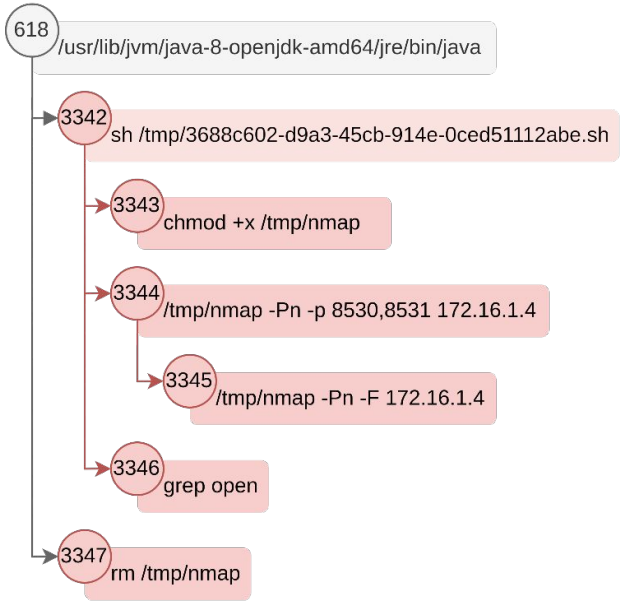
- Identify steps
- Extract commands

1 - Github SigmaHQ/sigma

System logs

From observed data to attack step

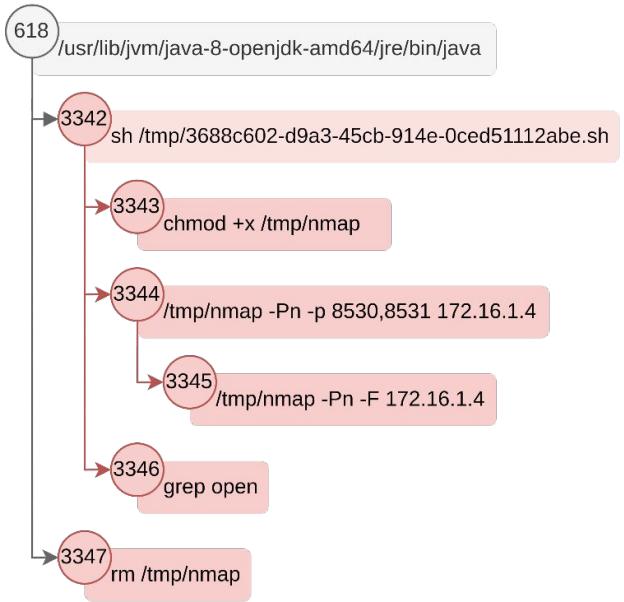
Process tree (reconstructed from logs)



System logs

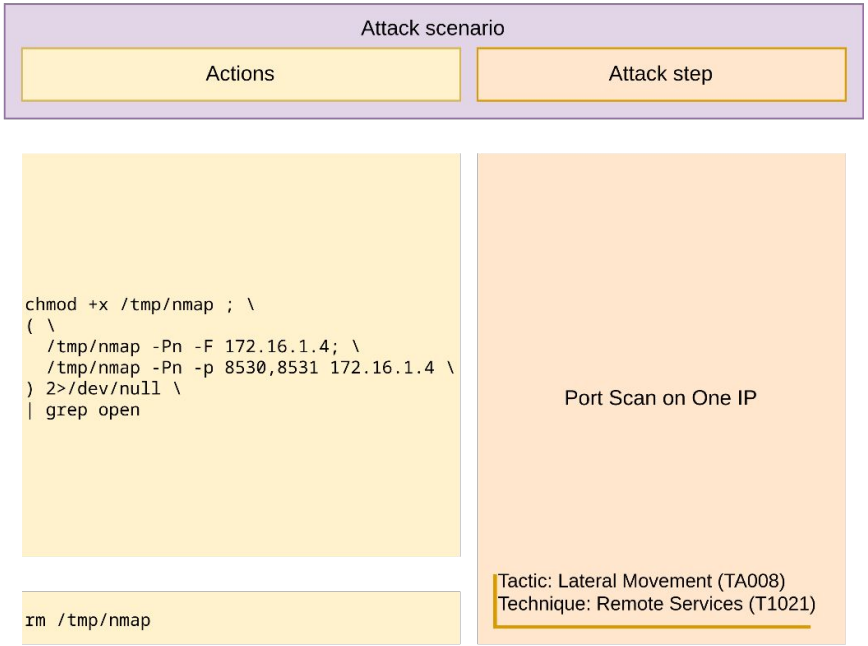
From observed data to attack step

Process tree (reconstructed from logs)



rules

Attack metadata



System logs

Improvements

Enhance robustness

- Actions coverage
- Platform specific rules

Improve detection performance

- Add pivoting rules
  - On scenario specific rules hits
  - Indicators: Process IDentifier (PID), hash, ...
- Confidence score: Sigma tags

Workflow output:
rules with matching logs

```
├── 10_linux_python_reverse_api
├── 13_log4j_exploit
├── 2_malicious_log4j_server
├── 33_subnet_scan
│   ├── 33_subnet_scan_1.yaml
│   ├── 33_subnet_scan_2.yaml
│   ├── attack_1.log
│   └── attack_2.log
├── 3_nginx_server
├── 46_port_scan
│   ├── 46_port_scan_1.yaml
│   ├── 46_port_scan_2.yaml
│   ├── attack_1.log
│   └── attack_2.log
├── 50_ssh_bruteforce
│   ├── 50_ssh_bruteforce_2.yaml
│   ├── 50_ssh_bruteforce_3.yaml
│   ├── attack_2.log
│   └── attack_3.log
├── 52_lateral_remote_control
│   ├── 52_lateral_remote_control_1.yaml
│   └── attack_1.log
├── 62_list_local_process
│   ├── 62_list_local_process_0.yaml
│   └── attack_0.log
├── 7_api_control
└── 94_env_var_api_keys_stealing
    ├── 94_env_var_api_keys_stealing_0.yaml
    └── attack_0.log
```

Contributions: develop a hybrid approach

1 - Rule-Based Detection

Exploit system logs (ECS)

2 - Anomaly Detection

Exploit network logs (PACP)

Network logs
with anomaly detection approach

Network logs

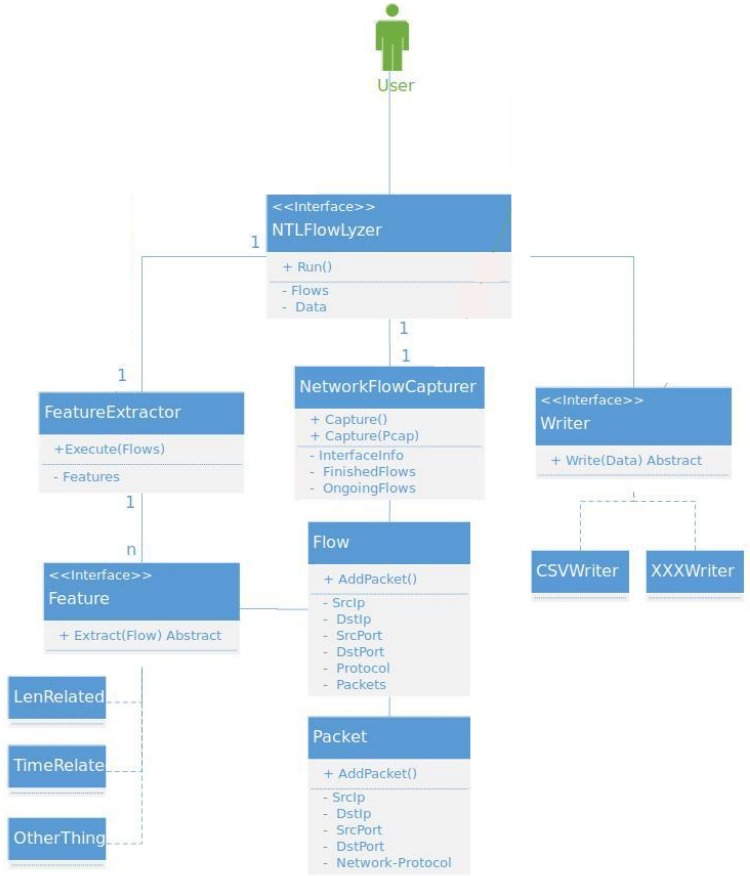
Features extraction

- PCAP file with packets
- Flow meter tool [2]
  - bidirectional flows from layer 3 and 4 data
  - time related features
- Some output features
  - flow_id
  - src_ip
  - src_port
  - dst_ip
  - dst_port
  - protocol
  - IAT
  - timestamp
  - Duration

2 - Github hieulw/cicflowmeter

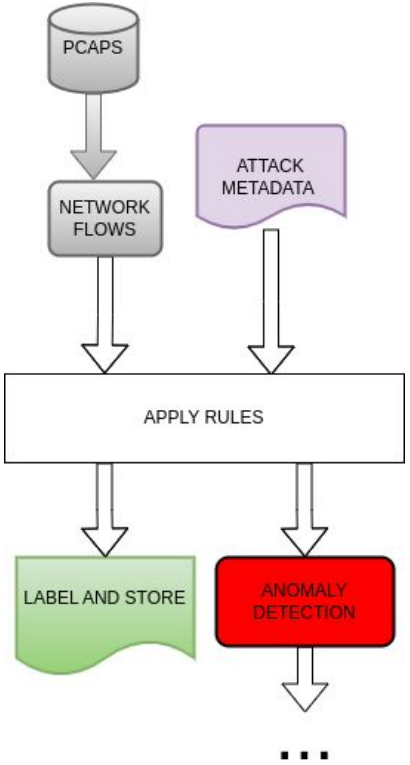
Network logs

Flow Generation



*A. H. Lashkari,et al (2024)

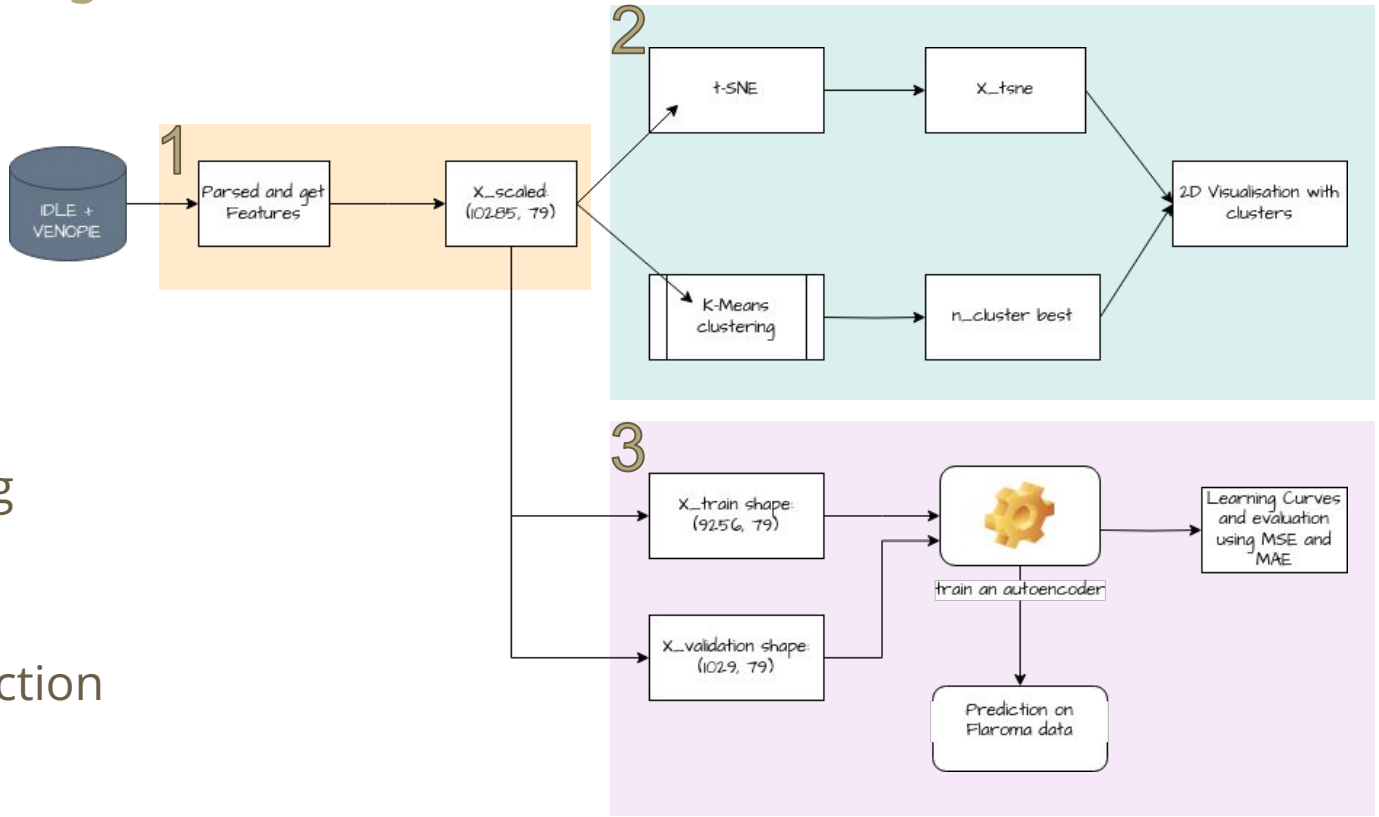
Network flow labelling




```
BEGIN

  LOAD PCAPS
  LOAD attack_metadata
  PROCESS network_flows  FROM PCAPs

  FOR EACH flow IN network_flows  DO
      extract (src ip, dst ip, timestamp)

      IF (src ip, dst ip, timestamp)  MATCH
attack_metadata  THEN

          LABEL flow AS 'attack_metadata.name'
          STORE flow
      ELSE

          CALL AnomalyDetection (flow)
      END IF
  END FOR
END
```

Network logs

Machine learning workflow



1 Pre-processing

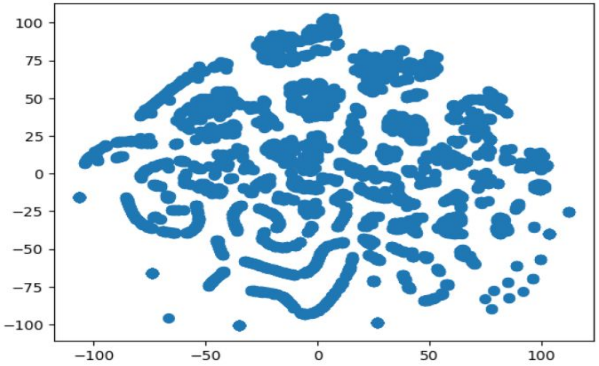
2 Clustering

3 Anomaly-detection

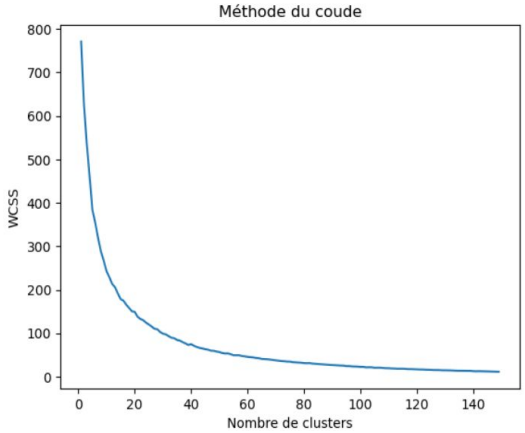
Network logs

Clustering

X_train Shape: (10285,79)

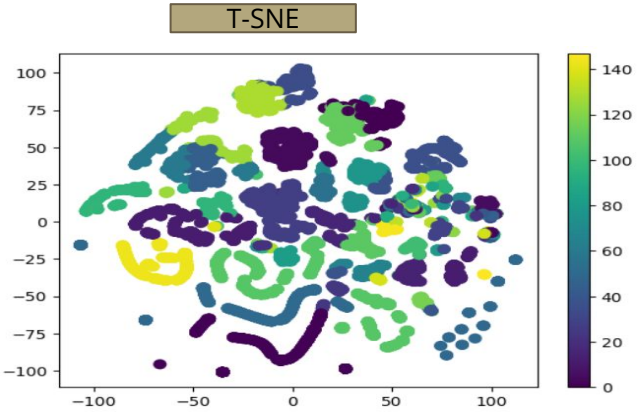


K-Means



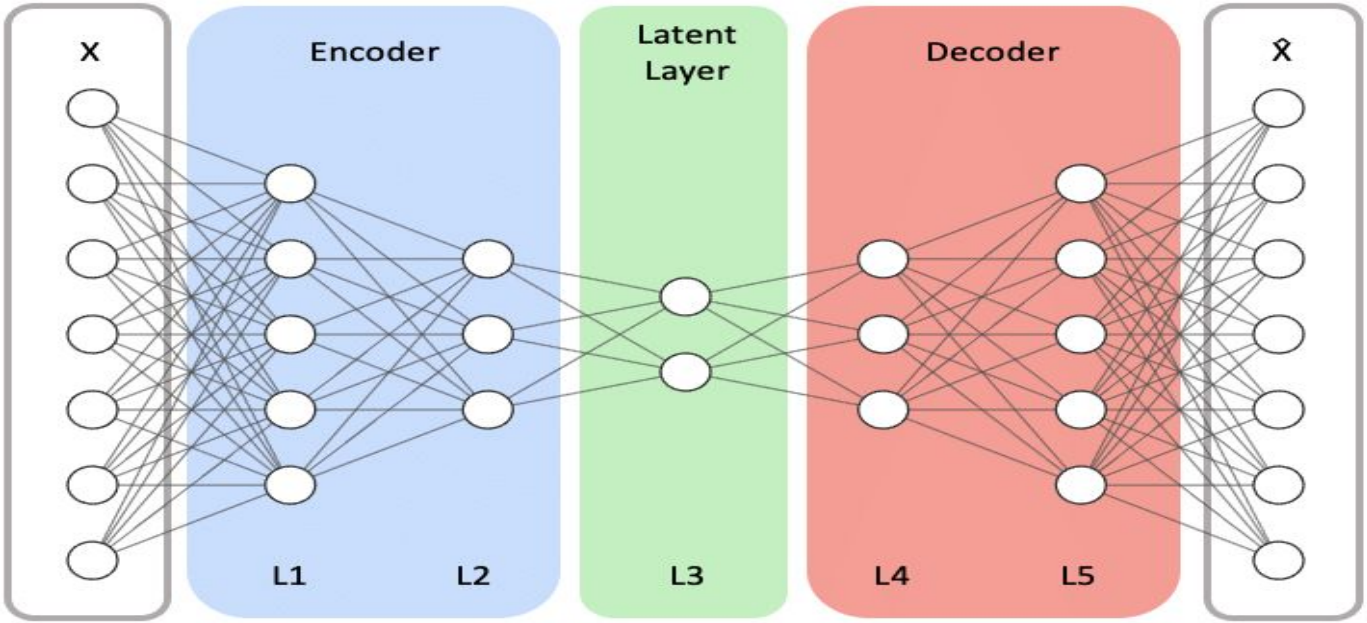
n_clusters

T-SNE



Network logs

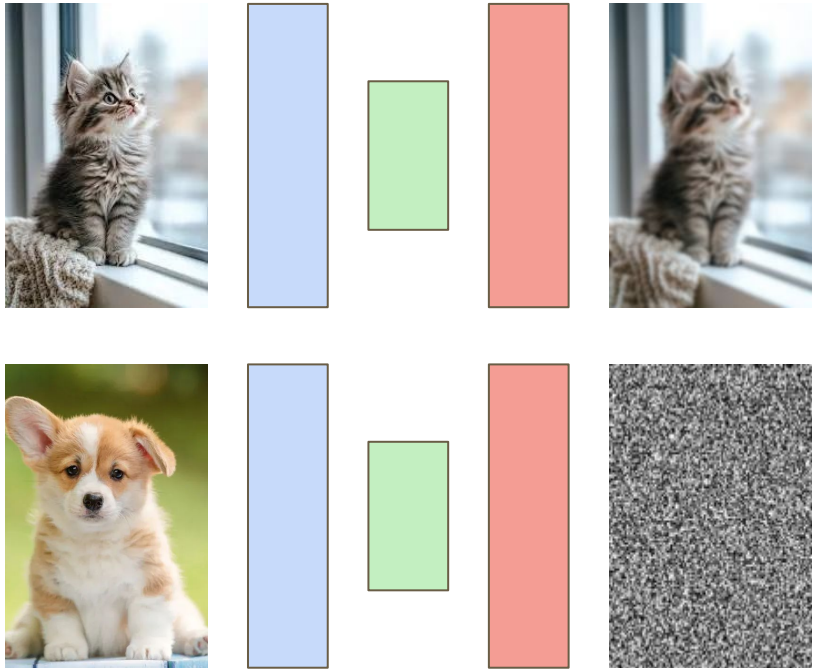
Autoencoder



Song, Y (2021).

Network logs

Autoencoder for Anomaly Detection (AD)



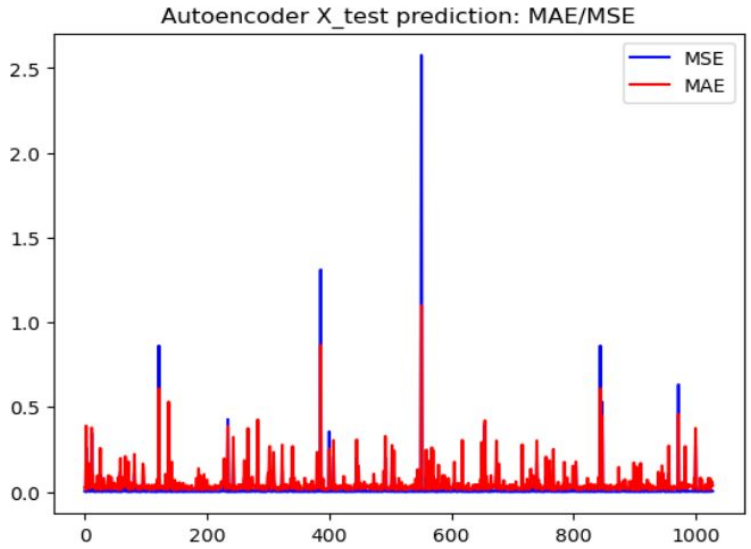
E( , ) = 0.01 Benign

Threshold defined on benign data

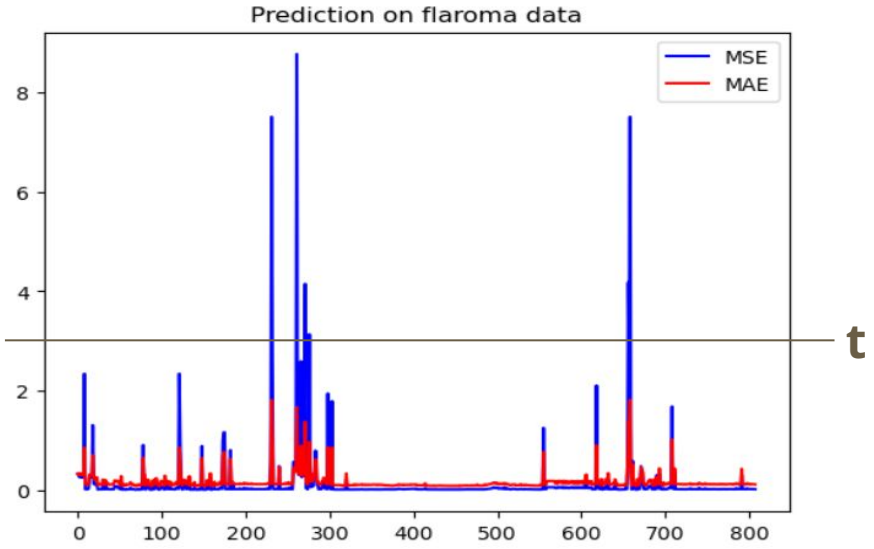
E( , ) = 10.7 **Anomaly**

Network logs

Autoencoder results



Only Benign



With Attack

Network logs

Improvements

- More benign data for training
- Find best features
- Labeled dataset to see if the approach works
- Impact of the topology of the network ?
  - General model or need to train on the targeted network ?
- Use transfer learning on existing models
  - Reduces need of having a large dataset
- Sequence of network log
  - More information than in single log
  - Long Short-Term Memory Neural Network

Future work 🔭

- Ensuring reproductible normal trafic
  - Ease detection of attack logs
- Feedback loop:
  - Use anomaly detection to create new sigma rules
- More ML methods
  - E.g. ensemble learning

Take away

Rules are a great start …. and can be completed with anomaly detection

Need for a feedback loop

Acknowledgments

Pascal, Alexandre & GDR Sécurité
Amossys (and Cosmian, ScreenAct)
CIRM

Our team :)