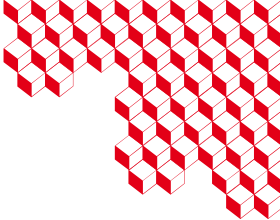


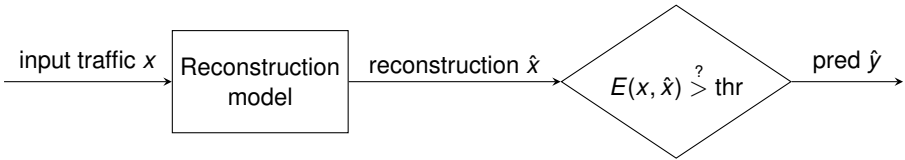
Towards automated network intrusion response

Thomas MARCHIORO

March 11, 2025

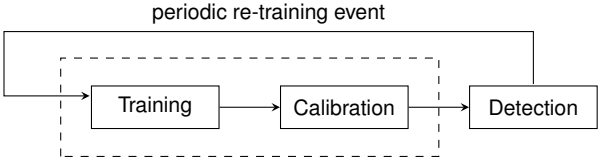
Context: SIGMO-IDS

- **SIGMO-IDS**: Unsupervised (anomaly-based) network IDS, runs on a host and monitors network interfaces
  - Similar to Kitsune
- Detection pipeline:



Context: SIGMO-IDS

- **SIGMO-IDS**: Training/inference workflow



- **Training**: Autoencoder learns to predict normal traffic
- **Calibration**: calculate a threshold based on distances between the predicted network traffic and the actual observed traffic
- Training and calibration are performed using data collected from the monitored network interfaces

From intrusion detection to intrusion response

- You are an L1 SOC analyst, going through your SIEM alerts
- You receive a sequence of alerts from the new top-tier AI NIDS with $< 0.01\%$ FPR

timestamp	src ip	src port	dst ip	dst port	protocol	...	score	label
556172725	192.168.1.30	42966	52.59.177.21	80	tcp	...	0.99	anomaly
1556173067	192.168.1.31	50266	18.184.104.180	80	tcp	...	0.95	anomaly
1556173461	192.168.1.31	34532	18.184.104.180	80	tcp	...	0.63	anomaly
1556173488	192.168.1.31	35706	18.184.104.180	80	tcp	...	0.68	anomaly
1556340863	192.168.1.32	59874	18.194.169.124	80	tcp	...	0.81	anomaly
1556340869	192.168.1.32	50204	18.194.169.124	80	tcp	...	0.84	anomaly
1556340880	192.168.1.32	58430	18.194.169.124	80	tcp	...	0.98	anomaly
1556340887	192.168.1.32	42504	52.28.231.150	80	tcp	...	0.69	anomaly
1556340895	192.168.1.32	45016	176.28.50.165	80	tcp	...	0.73	anomaly
1556548974	192.168.1.34	11	74.125.109.8	-	icmp	...	0.78	anomaly
...	...	...	...	...	...	...	...	...

- **What do you do?** *(a) Inspect everything manually (b) Turn off the NIDS (c) Change job*

State of the art of intrusion response

Industry:

- Signature-based attack trace (e.g., with Sigma rules) – works for specific CVEs / tactics
- Aggregation of security events (e.g., counts of insecure protocols, expired certificates)
- Playbooks – often not followed because too case-specific or company-specific[1]

Countless hours of manual inspection needed $\rightarrow$ **Alert fatigue**

Academia:

- Explanation-oriented – attack graphs[2]
- Action-oriented – game theory, MDP/RL[3]

Open problem: **lack of benchmarks** / **standardized evaluation**

[1] Schlette D et al. Do you play it by the books? A study on incident response playbooks and influencing factors. 2024 IEEE Symposium on Security and Privacy (SP).

[2] Rose JR et al. IDERES: Intrusion detection and response system using machine learning and attack graphs. 2022 Journal of Systems Architecture

[3] Iannucci S, et al. A performance evaluation of deep reinforcement learning for model-based intrusion response. 2019 IEEE FAS*W.

Intrusion response: Desiderata

Desirable properties for an intrusion response system:

- **General**: Apply to different scenarios (intranet, cloud, SCADA, industrial IoT)
- **Actionable**: Information should translate to practical actions
- **Verifiable**: Proposed responses should be easy to understand and verify
- **Measurable**: Should be easily comparable to other solutions

What we can learn from **network intrusion detection systems** (**NIDS**)[4]:

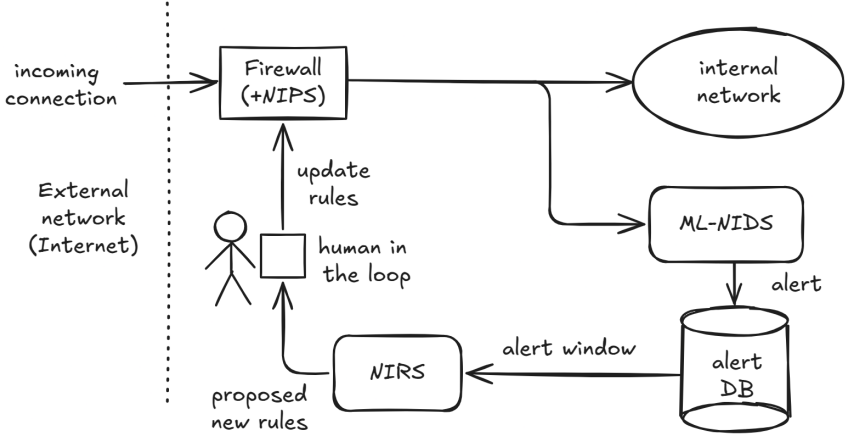
- Most attacks happen on a network → act at the network level
- NIDS can be deployed to virtually any network → deploy at the network entrypoints
- NIDS can be "easily" compared → use standard benchmark datasets / testbeds

→ Can we design "**network intrusion response systems**"?

[4] Apruzzese G et al. Sok: Pragmatic assessment of machine learning for network intrusion detection. In 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P).

Introducing NIRS

- **Network intrusion response systems (NIRS)**: IRS based on firewall rules



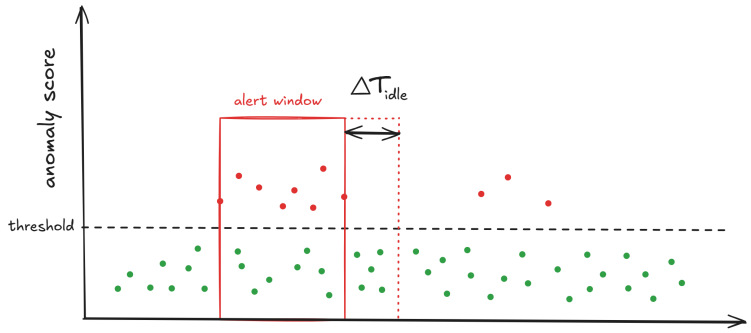
NIRS: Properties

- **General**: Pretty much any internal network is behind a firewall
- **Actionable**: Firewall rule update is simple and effective
- **Verifiable**: Firewall rules (e.g., iptables) have a very simple syntax
- **Measurable**: Can measure false positives (benign traffic accidentally blocked) and false negatives (malicious traffic that went through) $\rightarrow$ Can test on public NIDS datasets (CIC-IDS-2017, NB15, GTCS, etc.)

Limitation: Covers only traffic `external` $\leftrightarrow$ `internal`. If an attacker has already compromised a host on the internal network and is performing lateral movement, that cannot be directly handled by the NIRS.

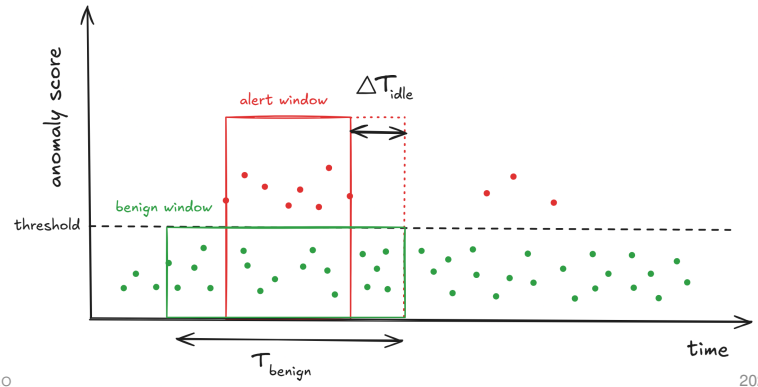
Aggregating NIDS alerts to generate firewall rules

- Idea 1: use an **incremental window** for alerts
  - The window starts when a first alert is received
  - It expands while new alerts are being raised
  - It stops expanding when no alert is received within a time period ΔT_{idle}
- Create a set of rules to block the alerts included in the window



Aggregating NIDS alerts to generate firewall rules

- Idea 2: use an **incremental window** for alerts and a **sliding window** for benign traffic
- Create a set of rules to block the alerts included in the window and does not block benign traffic
  - **Challenge**: rules matching both benign traffic and alerts



NIRS baseline

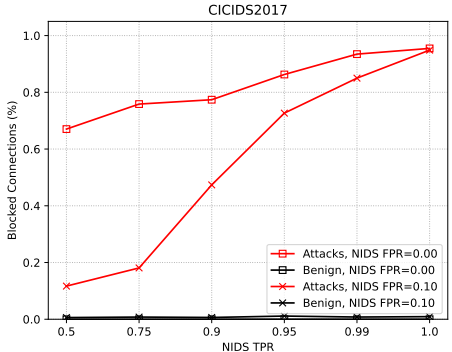
Proposed baseline: atomic rules for blocking traffic (using `iptables`) based on either of these patterns

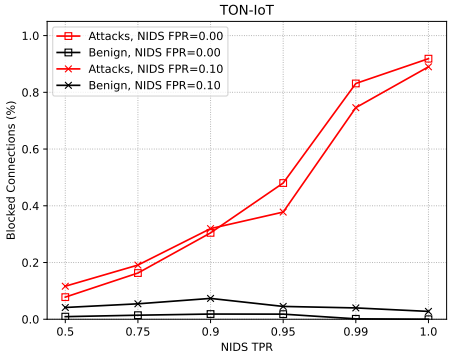
- Source IP:
  ```
  -A INPUT -s $SRC_IP -j DROP
  ```
- Destination IP and port:
  ```
  -A INPUT -p $PROTOCOL -d $SRC_IP --dport $DST_PORT -j DROP
  ```
- Destination IP and protocol:
  ```
  -A INPUT -d $SRC_IP -p $PROTOCOL -j DROP
  ```
- Specific connection:
  ```
  -A INPUT -p $PROTOCOL -s $SRC_IP --sport $SRC_PORT -d $DST_IP -j DROP
  ```

A **ruleset** iteratively is chosen to match the max amount of alerts in the alert window and $\leq \epsilon$ fraction of the benign window, with $0 \leq \epsilon \leq 1$

NIRS baseline: Results with ideal IDS

Results for $\epsilon = 0$, $\Delta T_{\text{idle}} = 60\text{s}$, $\Delta T_{\text{benign}} = 1\text{h}$ on CIC-IDS-2017 and TON-IoT

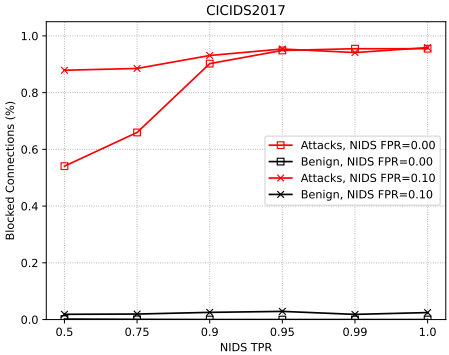


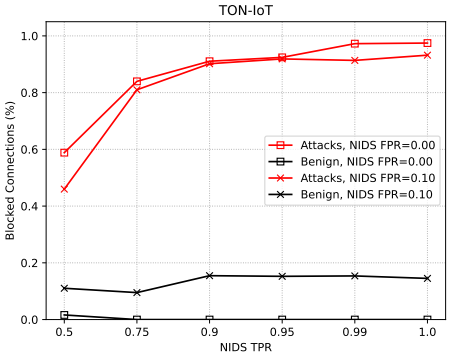


- *Note*: TON-IoT has balanced benign and malicious flows but more diverse IPs for the attackers; in CIC-IDS-2017, all attacks come from the same IP address space

NIRS baseline: Results with ideal IDS

Increasing tolerance to $\epsilon = 0.1$





How to do better than the baseline?

Intuition: creating non-trivial `iptables` rules requires

- **Pattern recognition**: finding patterns in the alert and benign data
- **Knowledge**: prioritizing one rule over another based on common attack patterns (e.g., for DDoS you do not want to block individual source IP addresses)
- **Context**: knowing the role of some of the hosts in the network (e.g., you typically don't want to block port 443 of your web server)

Good candidate for advancing NIRS: Retrieval-Augmented Generation (RAG)

- Pattern recognition $\rightarrow$ AI/LLMs are great at that
- Knowledge $\rightarrow$ Retrieval from knowledge database
- Context $\rightarrow$ Partial or complete context can be inserted in the prompt

Preliminary experiments

First step: benchmark LLMs on valid `iptables` rule generation
Setup:

- Local Ollama deployment
- “Small” sized models ($\leq$8b parameters)
- **Prevent random outcomes** $\rightarrow$ seed 42, temperature 0, context window 2048
- Prompt model to generate rules given context (alerts and benign from random windows)

Preliminary results: amount of **valid rulesets** out of 100 example

- `deepseek-r1:8b` – 0%, complete misunderstanding of the prompt
- `llama3:8b` – 97%
- `mistral:7b` – “0%”, wrong response format
- `qwen2.5:7b` – 98%

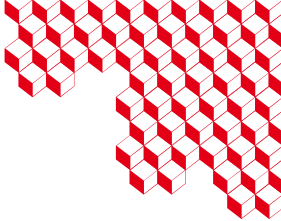
Conclusions

Recap:

- Firewall-based NIRS are simple and actionable
- Even baselines perform relatively well
- Easy to build on top of them (add knowledge, add context)
- Limited to `ext`↔`int` connections

Planned work:

- Better NIRS benchmarking → Looking forward to discussing this
- Simple RAG → Use examples from previous correct generations and/or atomic rules to fill a vector DB
- Better RAG infrastructure → rule validation with knowledge + context



Thank you! Questions?

CEA SACLAY
91 191 Gif-sur-Yvette Cedex
France
thomas.marchioro@cea.fr

Backup: Examples of rules produced by LLMs

Valid rules:

- `llama3: 8b: -A INPUT -s 172.16.0.1/32 -p tcp --dport 80 -j DROP`
- `qwen2.5:7b: -A INPUT -s 172.16.0.0/16 -d 192.168.10.0/24 -p tcp --dport 80 -m state --state NEW -j DROP`

Invalid rules:

- `llama3: 8b: -A INPUT -s 172.16.0.1/32 -p tcp --dport 1056:139 -j DROP`
- `qwen2.5:7b: -A INPUT -p tcp --sport 5000:65535 --dport 728-65535 -j DROP`