Introduction and Quick Reference for IPAL

Intrusion Detection with IPAL

- **Introduction to IPAL**
  - Why an Industrial Protocol Abstraction Layer?
  - Idea and Concept of IPAL
- **Hands-on IPAL**
  - Transcribing Industrial Protocols into IPAL
  - Introduction to the IIDS Framework
  - Tooling, Datasets & Development

Where to get an industrial IDS from?

- **Buy a solution from a vendor …**
  - Vendors sell closed systems, e.g., an industrial-grade hardware box
  - It is hard to figure out from public sources what the underlying detection models are
    - Often boils down to rule-based IDSs
  - May target a *single* industrial domain only

Figure from omicronenergy.com/

- **… or deploy state-of-the-art research IIDSs**
  - IIDSs from research are designed for narrow use-cases
  - Few IIDSs are available as open-source or on request only
  - Mostly built as prototype for scientific evaluation

There is no established IIDSs solution

The problem of protocol pluralism

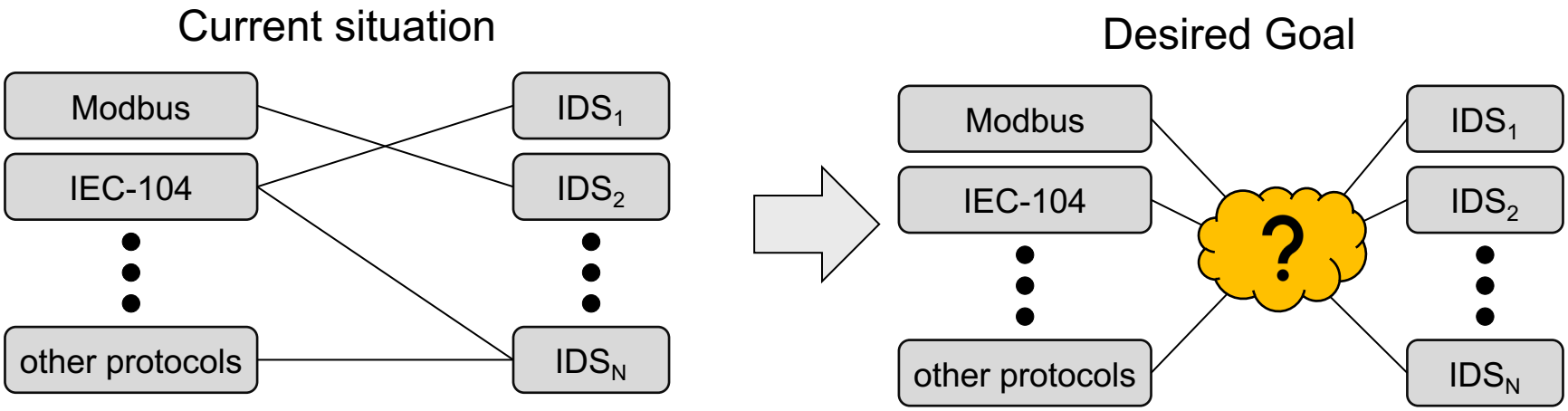
- **Industries developed a huge variety of niche protocols**
  - Modbus/TCP, CIP, IEC-104, NMEA 0183, …
- **The current state-of-the-art requires a tailored implementation of an IIDS for each protocol**
  - An IIDS developed for power distribution networks in Europe with the IEC-104 protocol cannot be (re-)used in America leveraging DNP3
- **Still, industrial protocols share a lot of common features**
  - They are all designed to exchange (physical) process values
  - They exhibit only three fundamental communication paradigms

Motivation designing a generic IIDS tool

- **The heterogeneity of the industrial domain is overrated**
  - Industries encompass all kinds of applications ...
    - Water treatment, power distribution, manufacturing, and so on
  - … but share great similarities w.r.t. the properties leveraged in IIDSs
    - Repetitiveness of machine-to-machine communication
    - Reoccurring processes leveraged by process-state based IIDSs
- **Still, IIDSs mostly focus on a single or few domains**
  - Approaches like DTMC have been proposed twice already for the water treatment domain or gas facilities
  - However, IIDSs are inherently meant to be adaptable!
    - E.g., anomaly detection trains the specific behaviors of an ICS

Do not re-invent similar IIDS for different domains!

Idea: The Industrial Protocol Abstraction Layer



- **Decouple IIDSs from the underlying industrial protocols**
  - IIDSs do not have to care about the protocol specifics anymore
  - Protocol parsing can be implemented by domain experts
- **Protocol-independence promises a seamless translation of IIDSs from one industrial protocol to another domain**

Abstracting Industrial Protocols

- **IPAL translates every industrial packet into a common format**
  - It must contain all information required by state-of-the-art IIDSs
- **Base**
  ident

Which packet features relevant for IIDSs can you think of?

Network-based IIDS

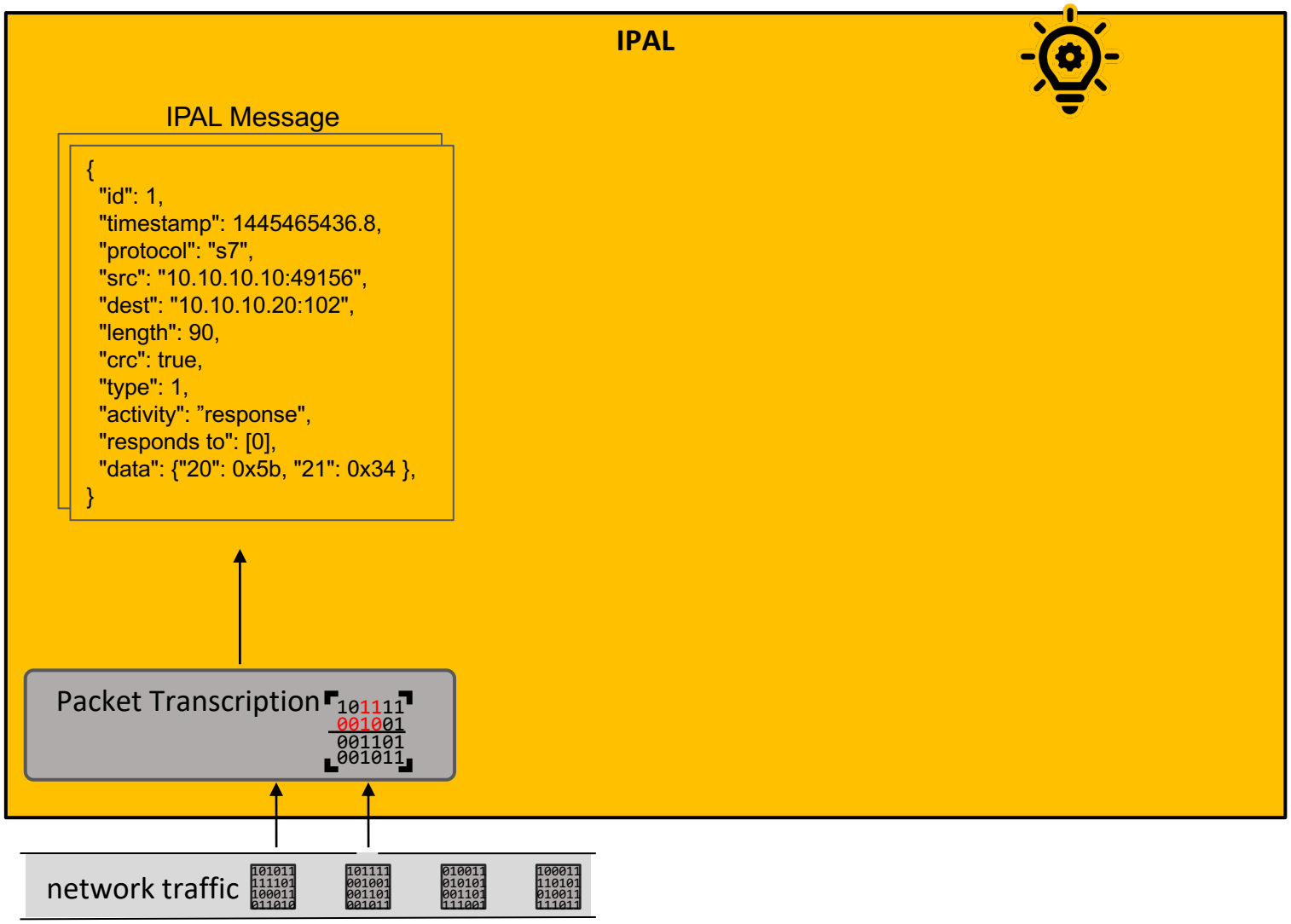
1. Timestamp
2. Source & Destination
3. Packet Length
4. Message Type
5. Activity (request, response, …)
6. Process Values
7. Attack Label
8. Responds to List
9. Unique ID

Process state-aware IIDS

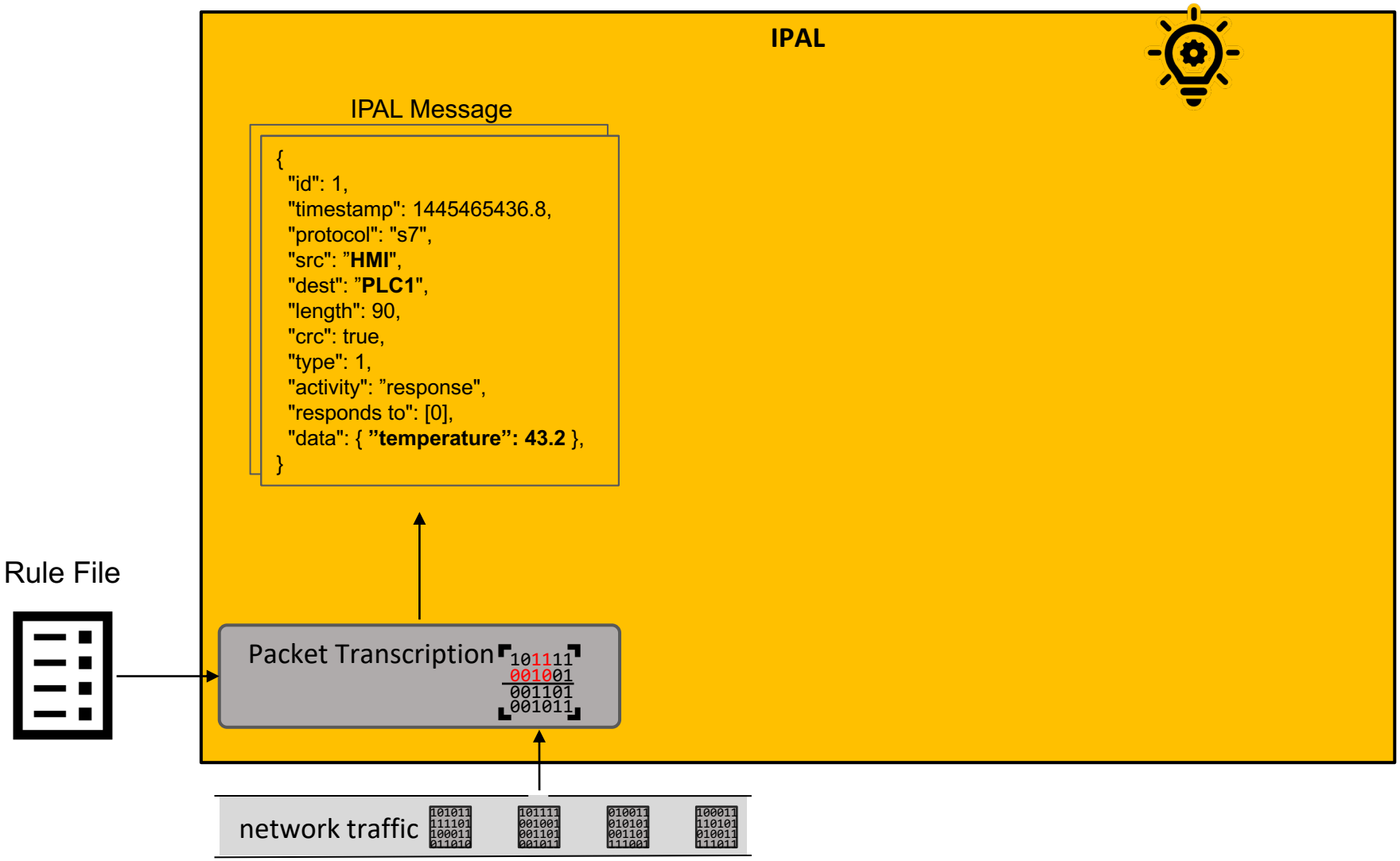
1. Timestamp
2. Process Values
3. Attack Label

	Publication(s)	Year	Detection Methodology	Representation	Timing	Comm. Entities	Message Type	Comm. Values	Process State	Packet Features	External Model	Trains on
Timings	Valdes et al. [113]	2009	Flow Periodicity	P	●	●	○	○	○	●	○	B
	Barbosa et al. [22]	2012	Periodicity	P	●	●	○	○	○	○	○	B
	Ponomarev et al. [92]	2016	Telemetry	P	●	●	○	○	○	●	○	A/B
	Lin et al. [80]	2017	Inter arrival time	P	●	●	●	●	○	○	○	B
	Lin et al. [79]	2019	Inter-arrival time	P	●	●	●	○	○	○	○	B
Sequences	Goldenberg et al. [50]	2013	DFA	P	●	●	●	◐	○	○	○	B
	Yoon et al. [121]	2014	PST	P	◐	●	●	●	○	○	○	B
	Caselli et al. [26, 27]	2015	DTMC	P	●	●	●	●	○	○	○	B
	Ferling et al. [42]	2018	DTMC	P	●	●	●	●	○	○	○	B
	Lin et al. [78]	2018	PST	P	●	●	●	●	○	○	○	B
	Yun et al. [122]	2018	Nearest Neighbor	P	●	●	○	○	○	○	○	B
Classifiers	Shang et al. [101]	2016	SVM	P	●	●	●	●	○	●	○	A/B
	Feng et al. [40]	2017	LSTM	P	●	●	●	○	●	●	○	B
	Perez et al. [82]	2018	SVM, RF, BLSTM	P	●	●	●	○	●	●	○	A/B
				P	●	●	●	○	●	●	○	A/B
				P	○	○	●	○	○	●	○	A/B
				P	●	◐	○	○	○	●	○	B
				P	●	●	○	○	○	●	○	A/B
Critic				S	○	○	○	○	●	○	●	A
				S	○	○	○	○	●	○	○	A/B
				S	○	○	○	○	●	○	○	B
				S	●	○	○	○	●	○	○	A/B
				S	○	○	○	○	●	○	●	A/B
	Kong et al. [62, 72, 73]	2016	Temporal logic	S	●	○	○	○	●	○	○	B
	Adepu et al. [2]	2016	Invariants	S	●	○	○	○	●	○	●	-
	Feng et al. [41]	2019	Invariants	S	○	○	○	○	●	○	○	B
	Monzer et al. [85]	2019	Rules	S	●	○	○	○	●	○	●	B
	Das et al. [35]	2020	Data Analysis	S	◐	○	○	○	●	○	○	A/B
Behavior Prediction	Hadžiosmanović et al. [52]	2014	Autoregression	S	●	○	○	○	●	○	○	B
	Caselli et al. [26]	2015	DTMC	S	●	○	○	○	●	○	○	B
	Ahmed et al. [6]	2017	Kalman Filter	S	●	○	○	○	●	○	○	B
	PASAD [15, 16]	2018	PCA	S	●	○	○	○	●	○	○	B
	Choi et al. [31]	2018	Control Invariants	S	●	○	○	○	●	○	○	B
	Myers et al. [87]	2018	Petri-nets	S	●	○	○	○	●	○	○	B
	Kravchik et al. [74]	2018	Neural Networks	S	●	○	○	○	●	○	○	B
	TABOR [81]	2018	TA, BN	S	●	○	○	○	●	○	○	B
	Anton et al. [12]	2019	Matrix Profiles	S	●	○	○	○	●	○	○	B
	HybTester [28]	2019	Hybrid-Automata	S	●	○	○	○	●	○	○	B
	Kim et al. [67]	2019	Neural Networks	S	●	○	○	○	●	○	○	B
	Denque Anton [13]	2020	Matrix Profiles	S	●	○	○	○	●	○	○	B
	SAVIOR [94]	2020	Physical Invariants	S	●	○	○	○	●	○	◐	B
Correlations	Krotofil et al. [75]	2015	Entropy Analysis	S	●	○	○	○	●	○	○	B
	Alippi et al. [10]	2016	Hidden Markov Model	S	●	○	○	○	●	○	○	B
	Aggarwal et al. [3]	2018	Hidden Markov Model	S	●	○	○	○	●	○	○	B
	Hau et al. [53]	2019	Statistics	S	●	○	○	○	●	○	○	B
	NoiSense [4]	2020	Noise Fingerprinting	S	●	○	○	○	●	○	◐	B
	ProcessSkew [8, 9]	2020	Noise Fingerprinting	S	●	○	○	○	●	○	●	B
Classifiers	Nader et al. [88]	2014	SVDD, KPCA	S	●	○	○	○	●	○	○	B
	Junejo et al. [63]	2016	Machine-learning	S	◐	○	○	○	●	○	○	A/B
	Inoue et al. [60]	2017	SVM, DNN	S	◐	○	○	○	●	○	○	B
	Chen et al. [29]	2018	SVM	S	○	○	○	○	●	○	○	A/B
	AADS [1]	2019	DNN	S	◐	○	○	○	●	○	○	B
	Anton et al. [12]	2019	OCVM, Isolation Forest	S	○	○	○	○	●	○	○	B
	FALCON [100]	2020	LSTM+ML	S	●	○	○	○	●	○	○	A/B

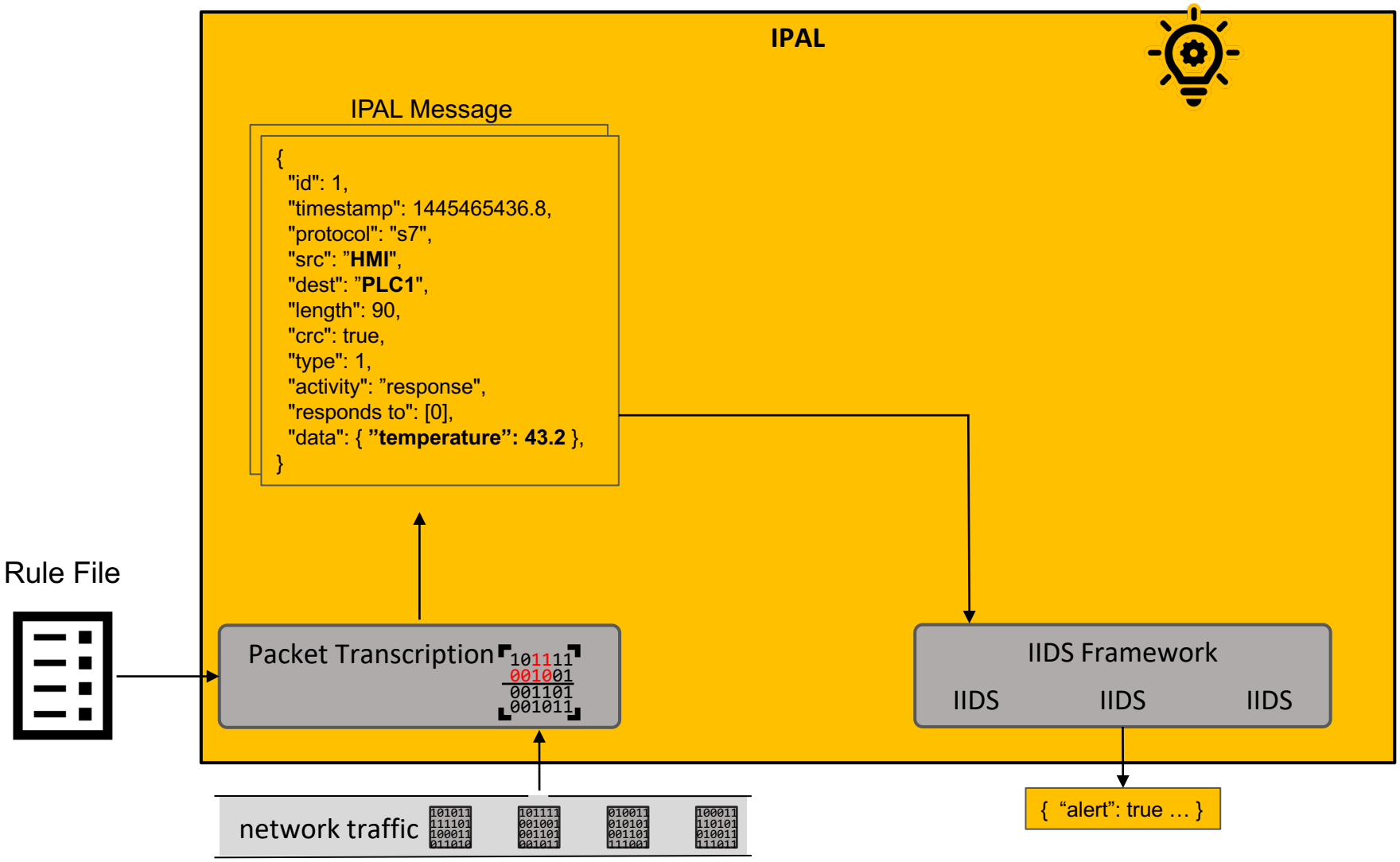
Workflow of the IPAL Framework – Transcriber



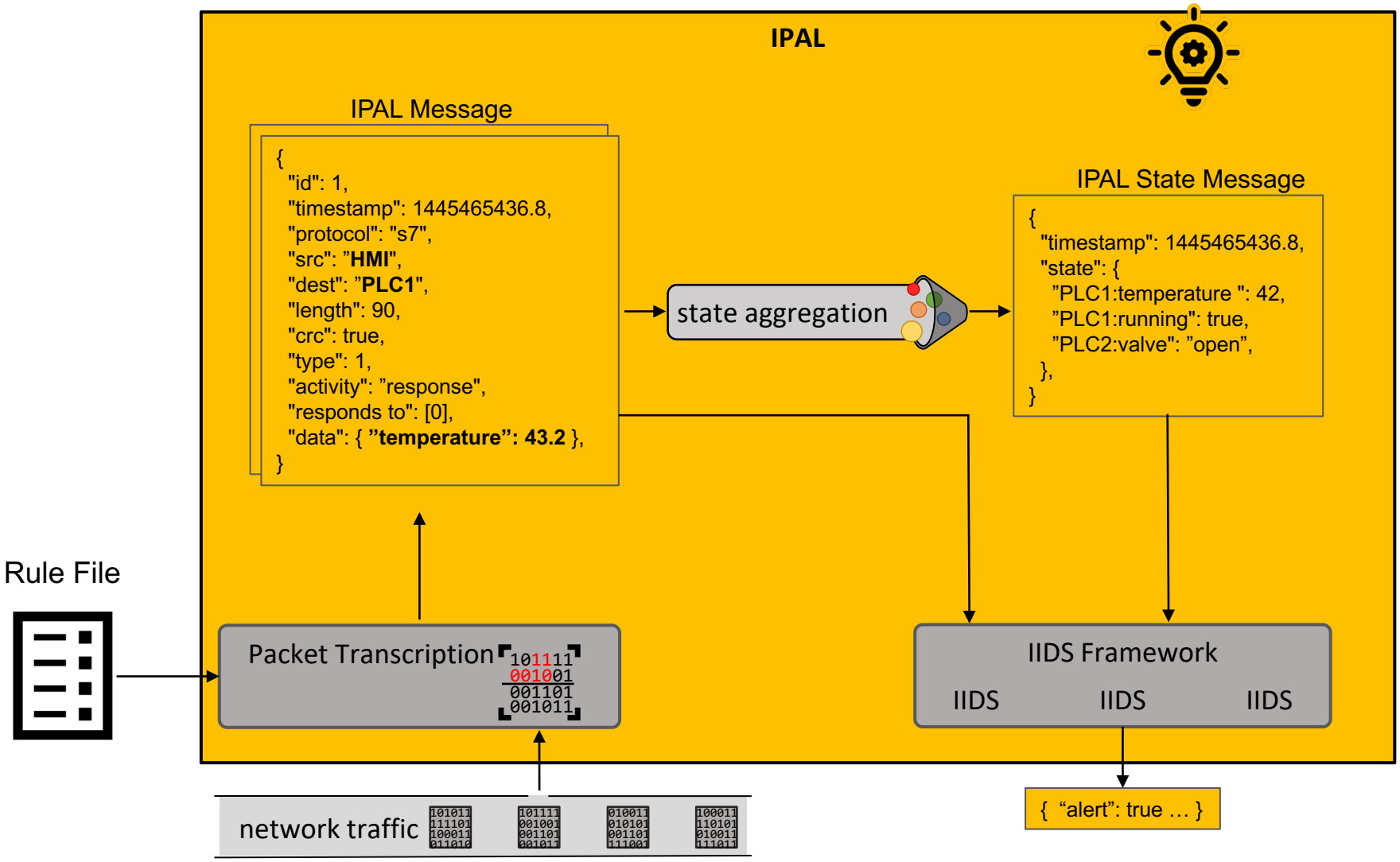
Workflow of the IPAL Framework – Rule Files



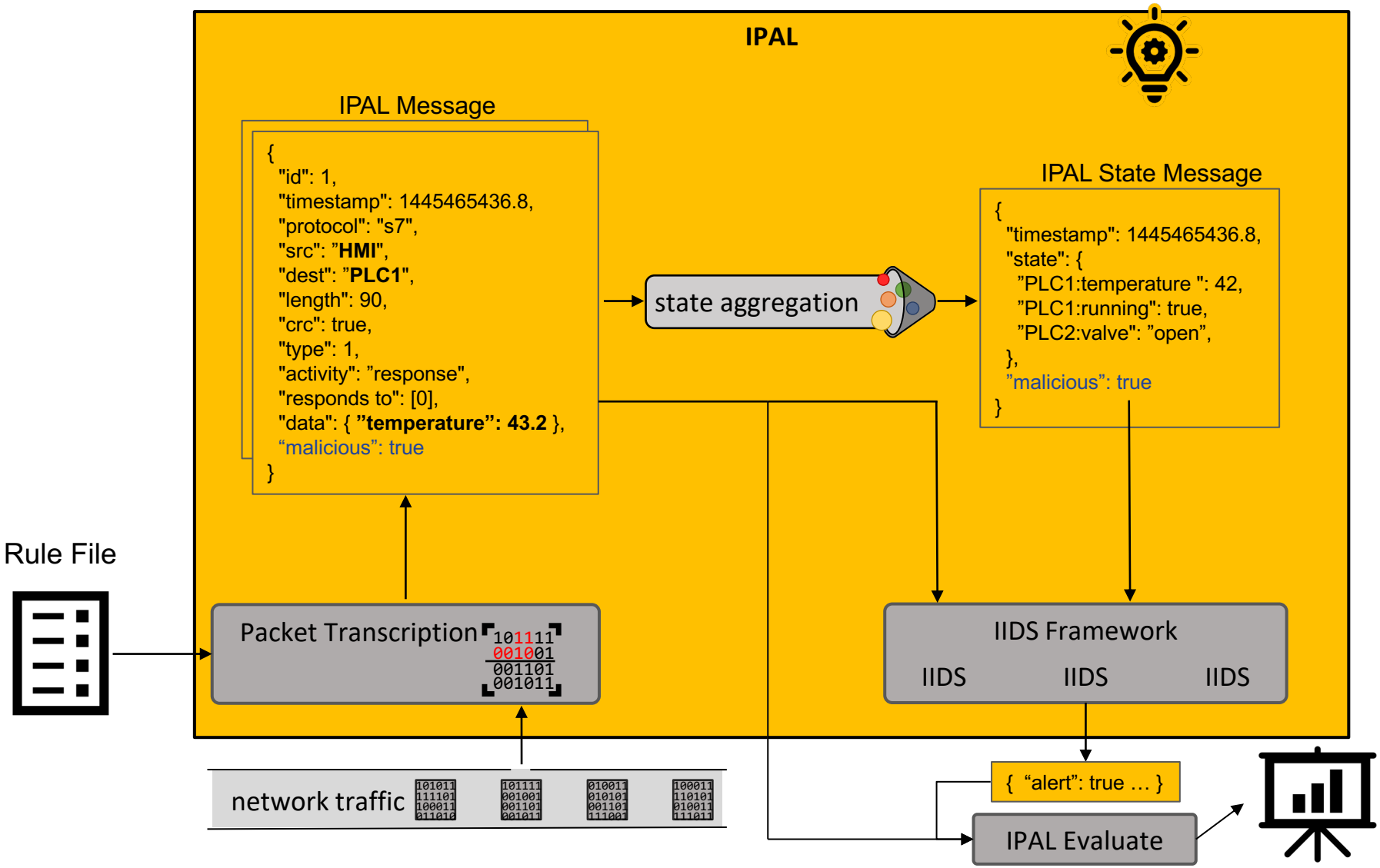
Workflow of the IPAL Framework – IIDS Framework



Workflow of the IPAL Framework – State Aggregation



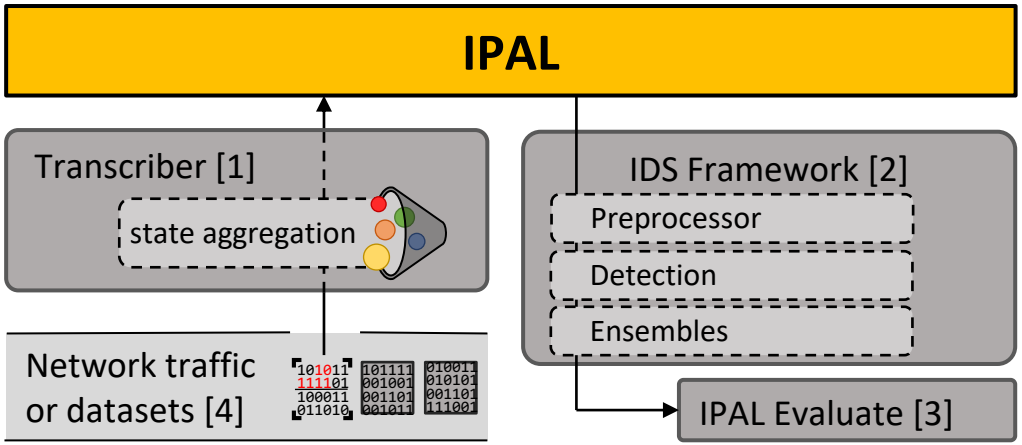
Workflow of the IPAL Framework – Evaluation



Chapter 4: Intrusion Detection with IPAL

- **Introduction to IPAL**
  - Why an Industrial Protocol Abstraction Layer?
  - Idea and Concept of IPAL
- **Hands-on IPAL**
  - Transcribing Industrial Protocols into IPAL
  - Introduction to the IIDS Framework
  - Tooling, Datasets & Development

IPAL Overview



- **Transcriber converts network traffic into IPAL messages or process-states**
  [1] https://github.com/fkie-cad/ipal_transcriber
- **IIDS Framework bundles training and execution of IIDS implementations**
  [2] https://github.com/fkie-cad/ipal_ids_framework
- **IIDSs' performance can be analyzed and visualized with IPAL Evaluate**
  [3] https://github.com/fkie-cad/ipal_evaluate
- **Datasets is a collection of datasets and scripts to convert them to IPAL**
  [4] https://github.com/fkie-cad/ipal_datasets

Quick Installation Guide

1. **Clone the repository**
2. **Installation**
   - Installation with pip install . (recommended) OR pip install -e .
   - Use it locally pip install –r requirements.txt and ./ipal-transcriber
   - Installation as virtual environment ./misc/install.sh
   - Building a docker image docker build –t <name>:latest .
   - Known Problem: too old packages, e.g., numpy or tshark
     - A quick Internet search may solve dependency issues easily
- **Optionally install the development tools**
  - pip3 install -r requirements-dev.txt
  - pre-commit install

Please refer to the README in each repository.

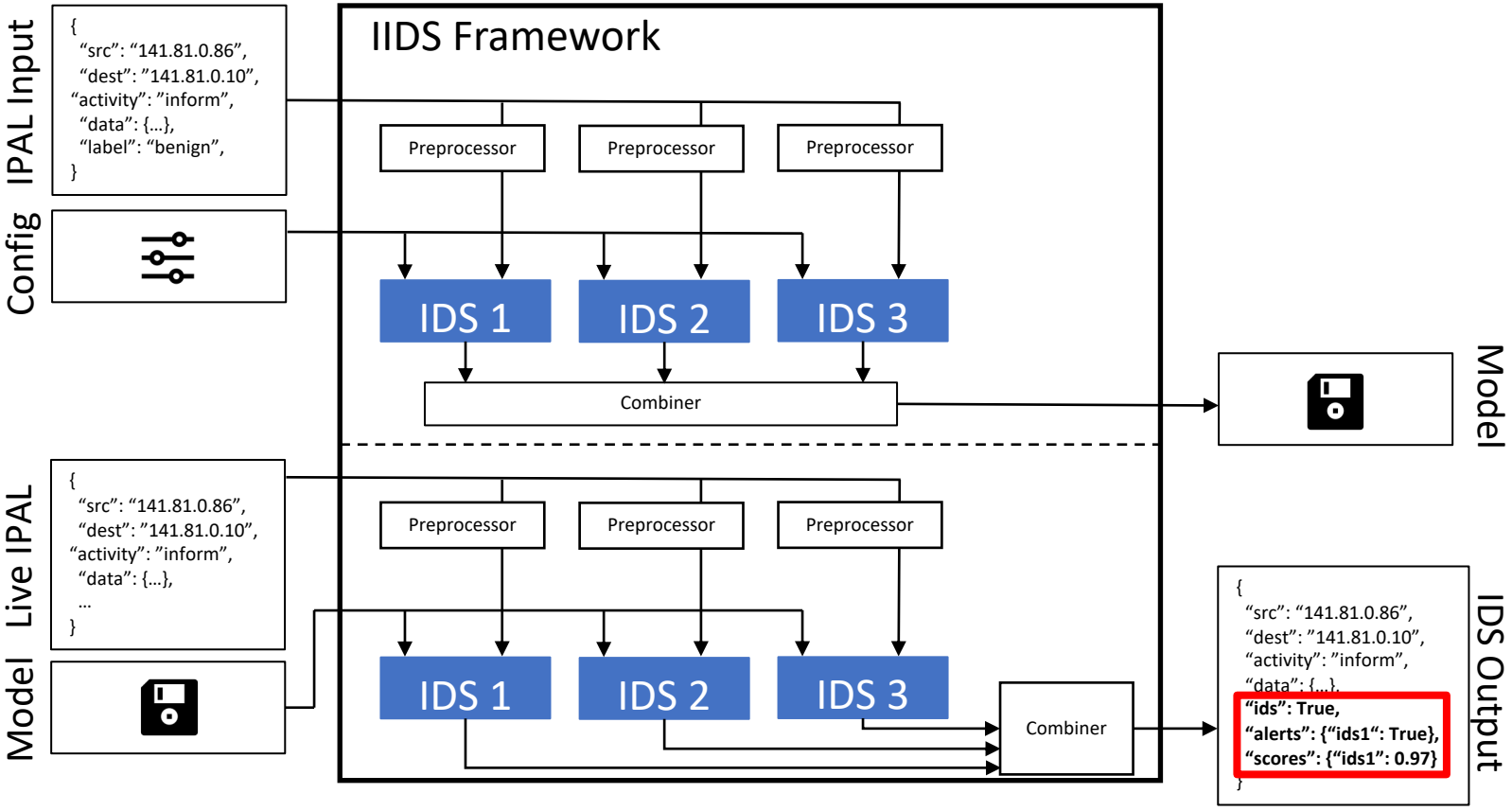
Transcriber Quick References

- **Transcribe a packet trace into IPAL format**
  - ipal-transcriber --pcap <pcap> --rules <rules> --ipal.output <ipal>
- **State Extraction to generate the process state**
  - ipal-state-extractor --ipal.input <ipal> --state.output <output> --filter <process variables> timeslice --timeslice.interval <ms>
  - ipal-transcriber --pcap <pcap> --rules <rules> --ipal.output <ipal> --filter <process variables> timeslice --timeslice.interval <ms>
- **Remove process data, e.g. after detection to reduce disc space**
  - ipal-minimize --all <ipal file>

- **Any file that ends with .gz get automatically compressed**
- **Files can be pipes with –**
  - ipal-transcriber --ipal.output – | grep PLC1 | ipal-state-extractor --ipal.input –
- **For further information ipal-transcriber –h or ipal-state-extractor -h**

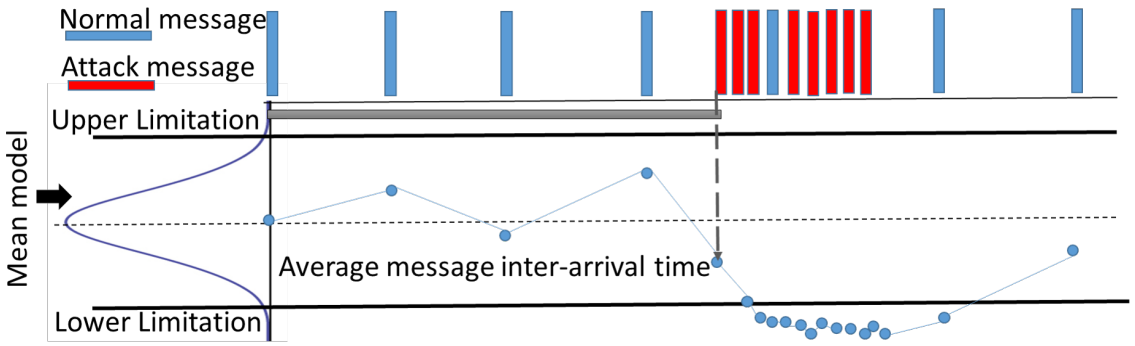
IIDS Framework – Behind the Scenes

- **The Framework is divided into training (learning a model) and live detection of anomalies**



IIDS Inter-arrival Time (Background Tutorial)

- **Idea: Attacks (DoS, PitM, PotS) influence timing between network packets**
  - Communication is periodic for one packet type
    - E.g., Maritime GNSS positions are broadcasted every two seconds
- **Approach: Model packet inter-arrival time**
  - Calculate average time between packets of the same type, derive standard deviation and margin of error
    - NMEA/GLL: inter-arrival time 2.006s (stddev 0.051)



Lin et al. "Timing-based anomaly detection in SCADA networks." CRITIS 2017

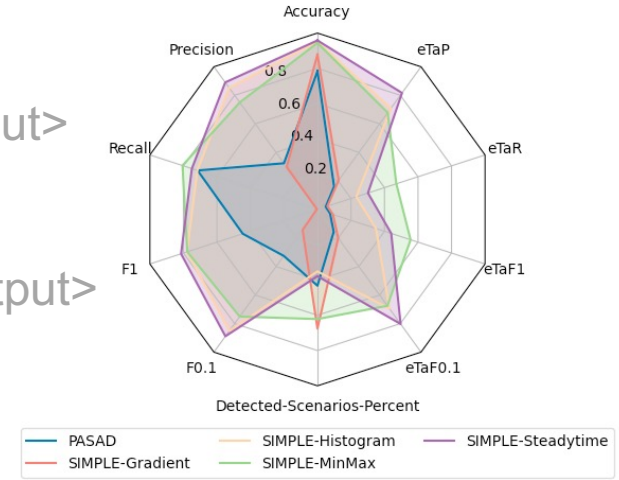
IIDS Framework Quick References

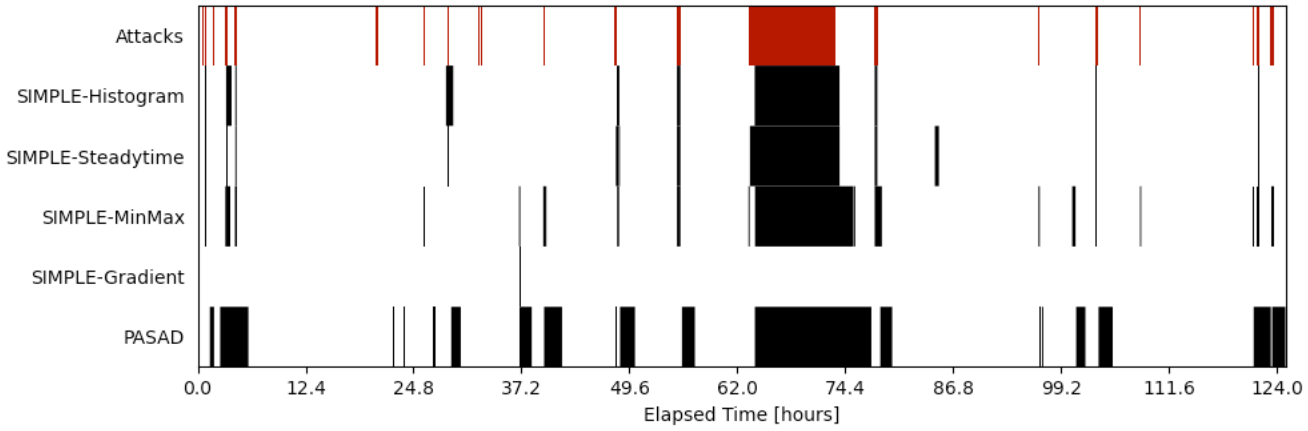
- **Obtaining a default IIDS configuration file**
  - ipal-iids --default.config <iids name>
- **Train an IIDS**
  - ipal-iids --config <config> --train.ipal <ipal> or --train.state <state>
  - ipal-visualize-model <config>
  - Retraining requires the --retrain option!
- **Performing Intrusion Detection**
  - ipal-iids --config <config> --output <output>
    --live.ipal <ipal> or --live.state <state>

- **Add --log info or --log debug for additional debugging information**

IPAL Evaluate Quick References

- **Analyze the performance of your IIDS**
  - ipal-evaluate --attacks <attacks.json> <IIDS output>
- **Visualize the results and alerts**
  - ipal-plot-alerts --attacks <attacks.json> <IIDS output>
  - ipal-plot-metrics <output of ipal-evaluate>





- **Lazy searching for an optimal parameter? Use ipal-tune**

How to Contribute & Git Merge Request Workflow

- **If you have troubles or if you encounter errors**
  - Feel free to contact us!
  - For bug reports, please open an issue on Github

- **Contributing Code / Bugfixes**

1. **Fork the repository**
2. **Implement fix on a new branch**
   - Test your code with pytest
3. **Push to your forked repository**
4. **Create a merge request**

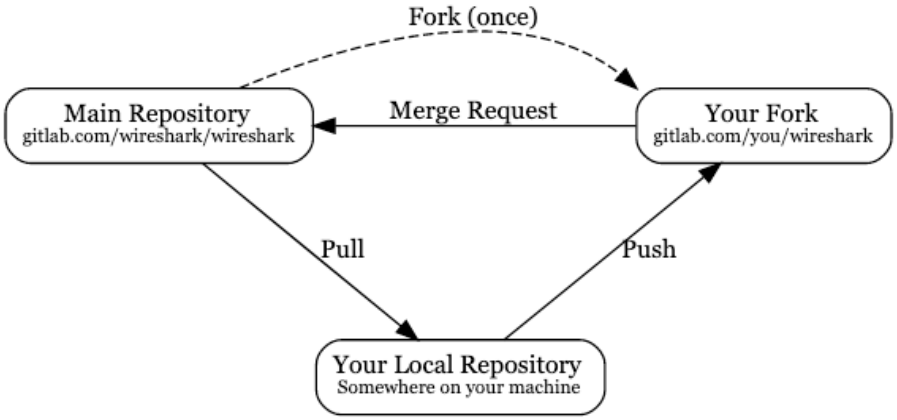


Image: https://www.wireshark.org/docs/wsdg_html_chunked/ChSrcContribute.html
[1] https://reflectoring.io/github-fork-and-pull