# Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems

Tristan Bilot, University of British Columbia
2025/12/05

# What is this talk about?

Mainly on 2 papers published at **USENIX Sec'25.**

- ***ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems***
- ***Sometimes Simpler is Better: A Comprehensive Analysis of State-of-the-Art Provenance-Based Intrusion Detection Systems***
  - + some other unpublished results

# What is this talk about?

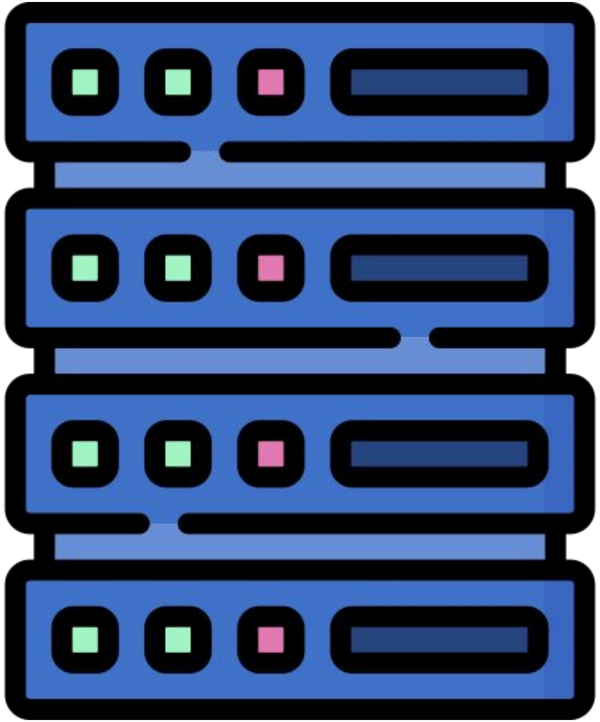
Mainly on 2 papers published at **USENIX Sec'25.**

- ***ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems***
- ***Sometimes Simpler is Better: A Comprehensive Analysis of State-of-the-Art Provenance-Based Intrusion Detection Systems***
    - + some other unpublished results

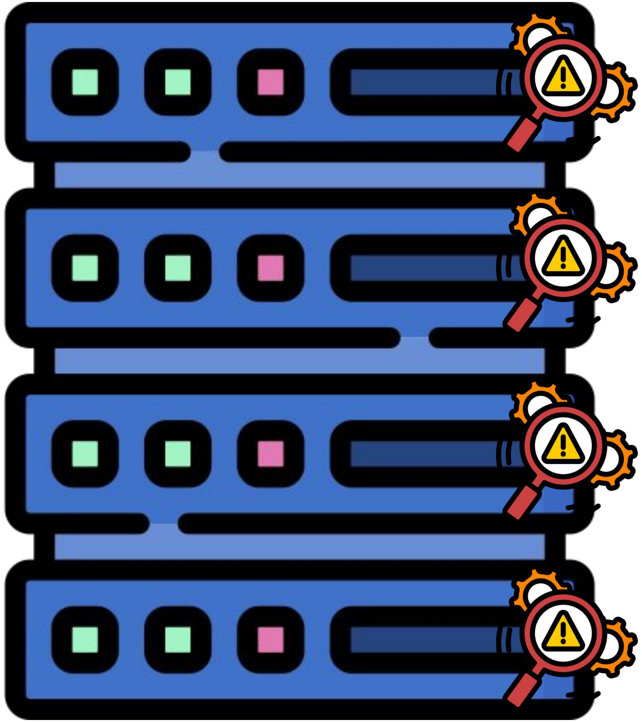
Do not hesitate to interrupt if you have a question

# Motivation

# Host-based Intrusion Detection

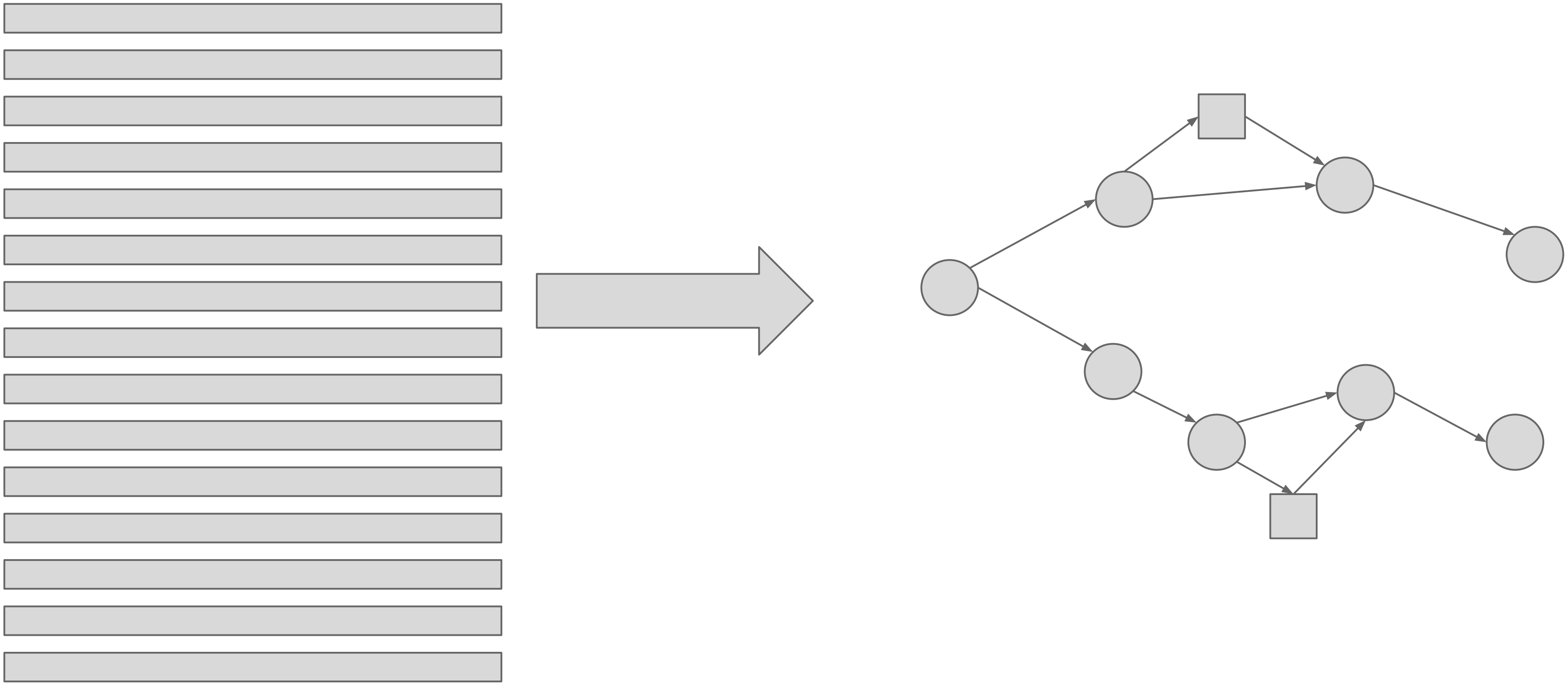


# Host-based Intrusion Detection Systems



- Deployed on individual machines
- Analyze system traces
- Look for signs of malicious behavior

# Provenance-based Intrusion Detection Systems (PIDS)



# How to build a provenance graph?

httpd receive packet from 63.169.38.150

httpd read config.ini

httpd read index.html

httpd send packet to http://63.169.38.150/

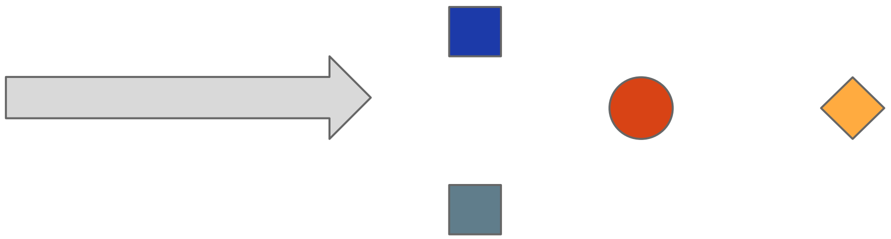
# How to build a provenance graph?

**httpd** receive packet from **63.169.38.150**

**httpd** read **config.ini**

**httpd** read **index.html**

**httpd** send packet to **63.169.38.150**



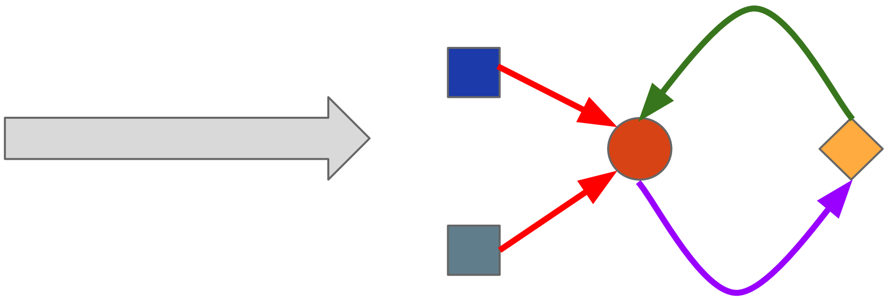
# How to build a provenance graph?

httpd receive packet from 63.169.38.150

httpd read config.ini

httpd read index.html

httpd send packet to 63.169.38.150



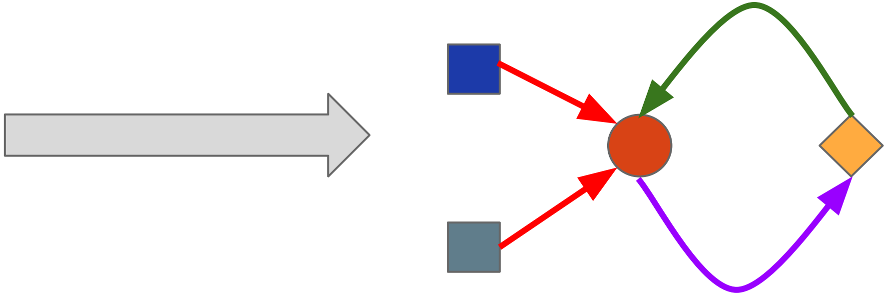
# How to build a provenance graph?

httpd receive packet from 63.169.38.150

httpd read config.ini

httpd read index.html

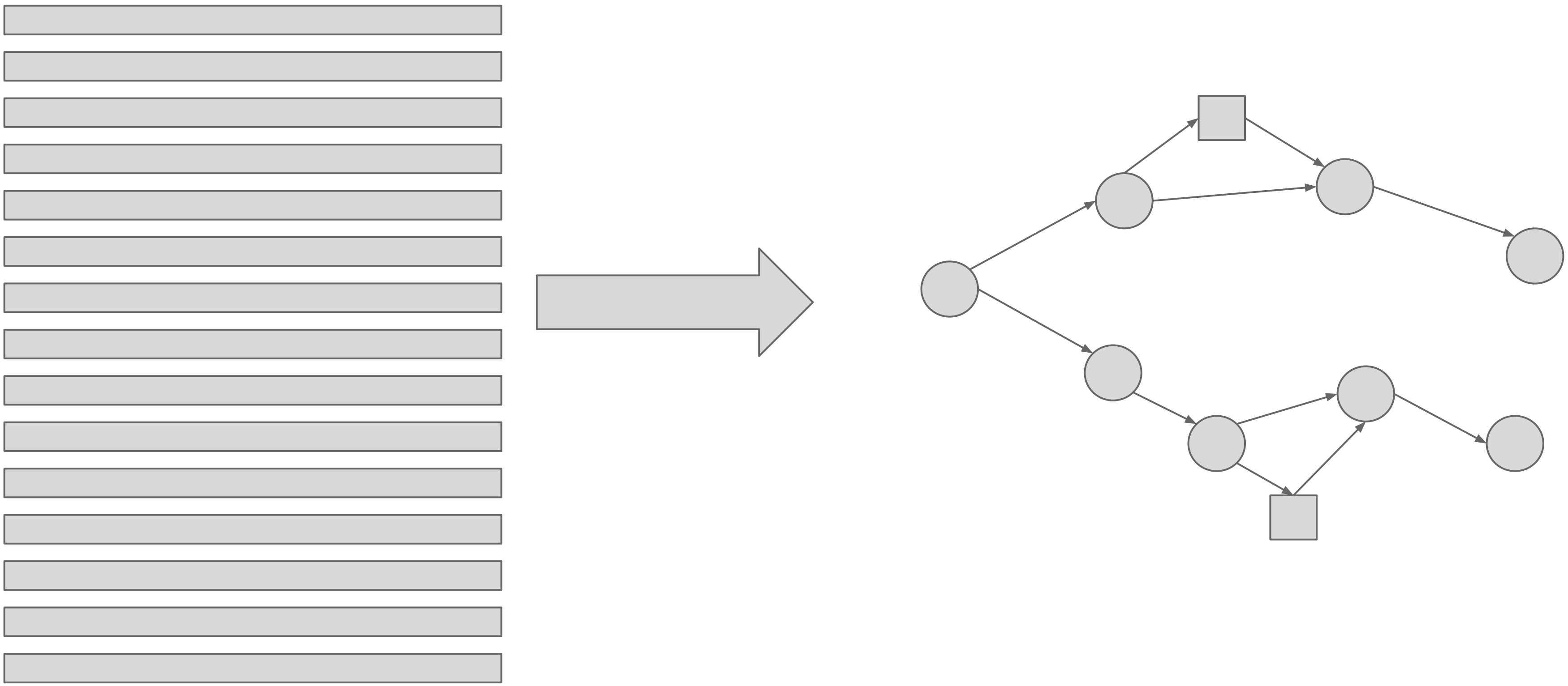
httpd send packet to 63.169.38.150



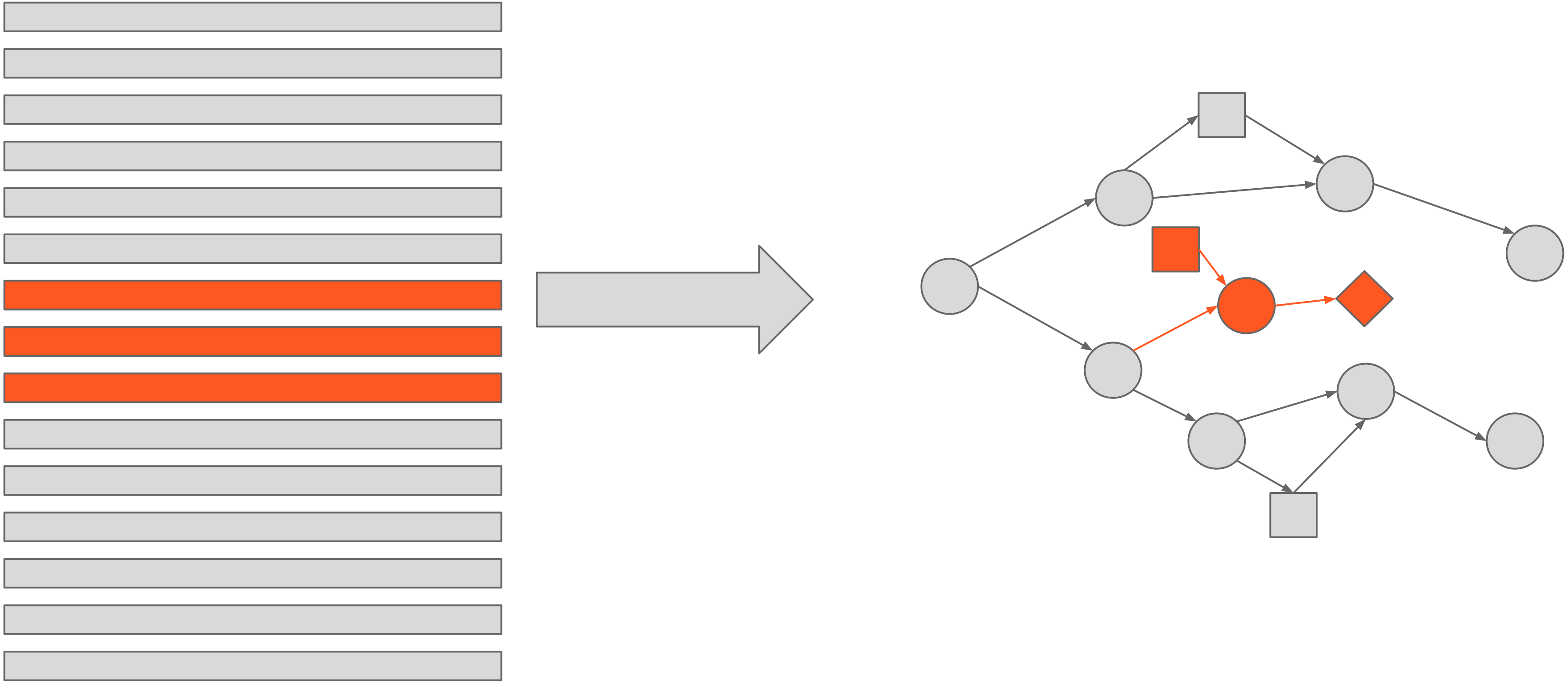
**Compiling the Linux Kernel:** ~2M graph elements

# What is the intuition behind PIDS?

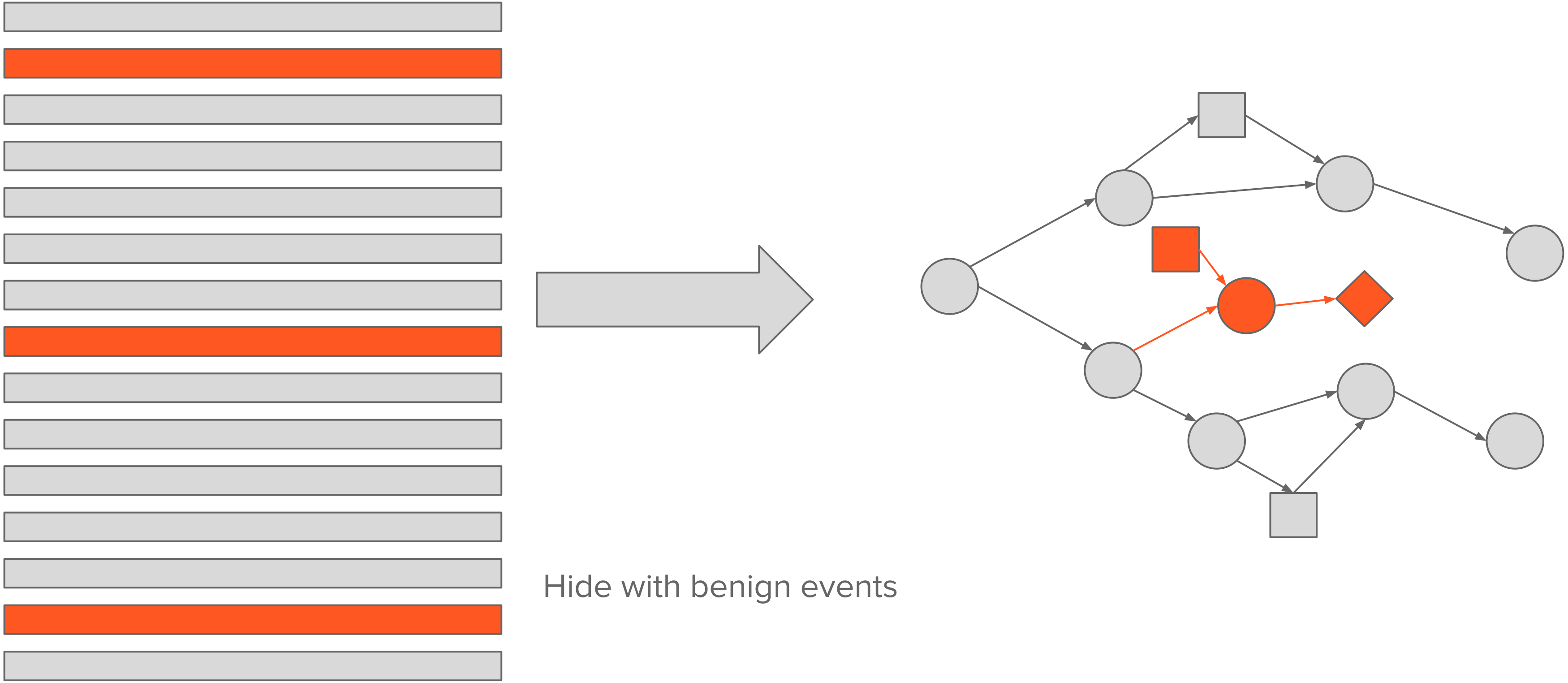
# What is the intuition behind PIDS?



# What is the intuition behind PIDS?

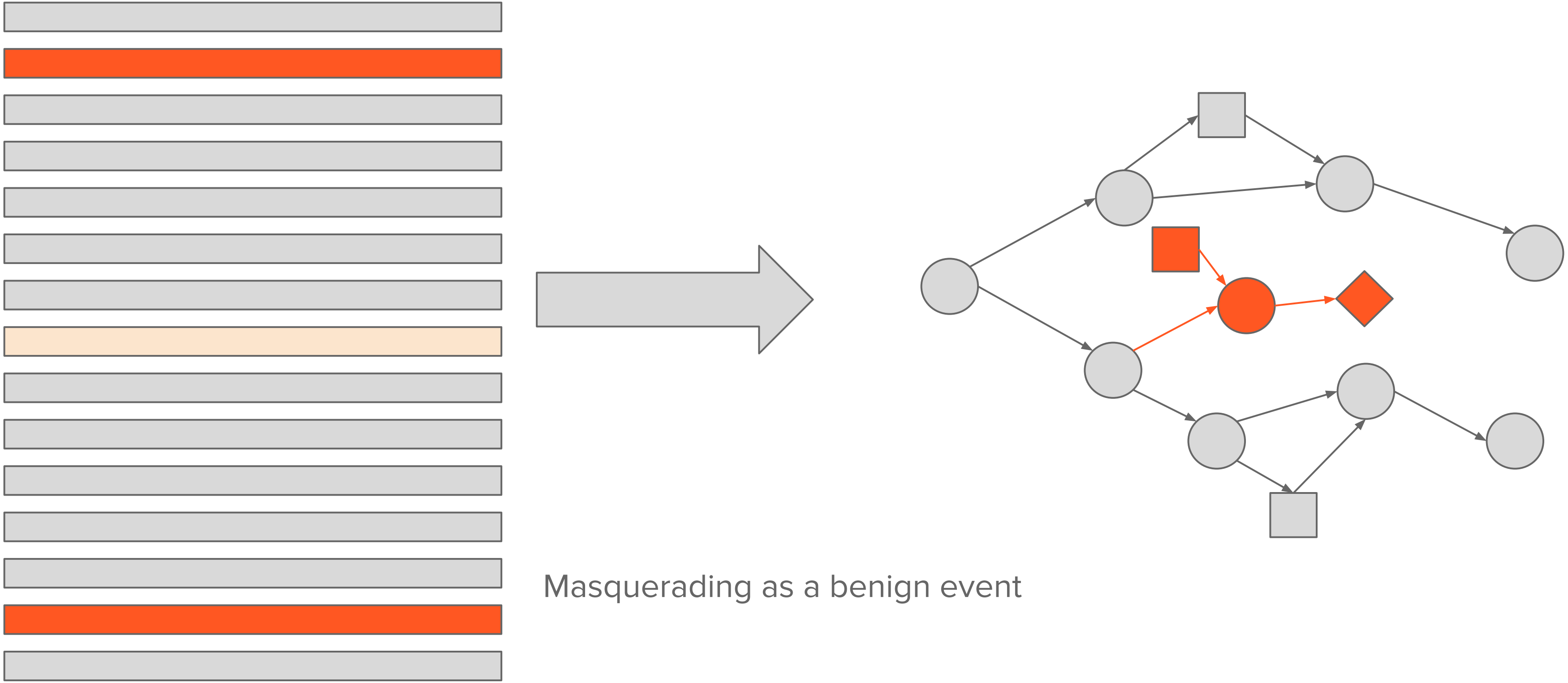


# What is the intuition behind PIDS?



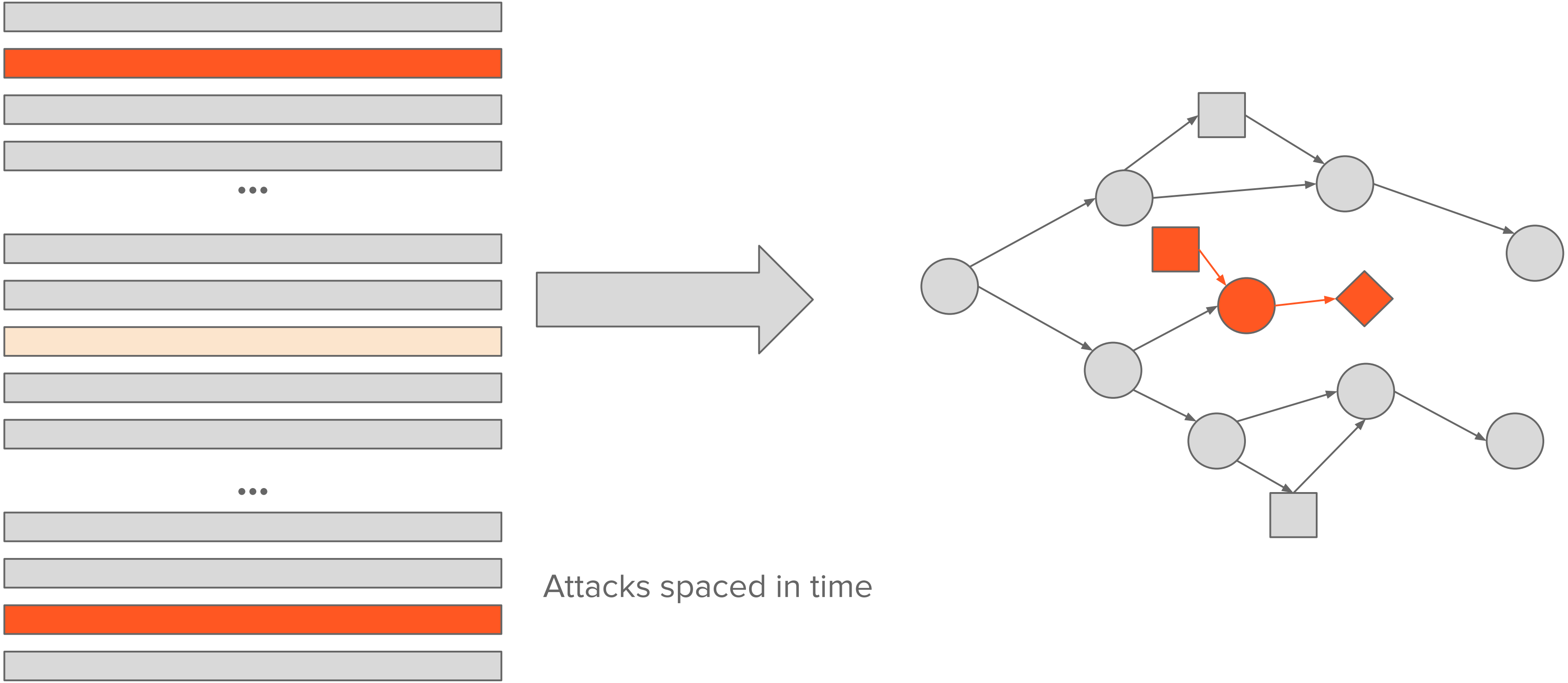
Hide with benign events

# What is the intuition behind PIDS?



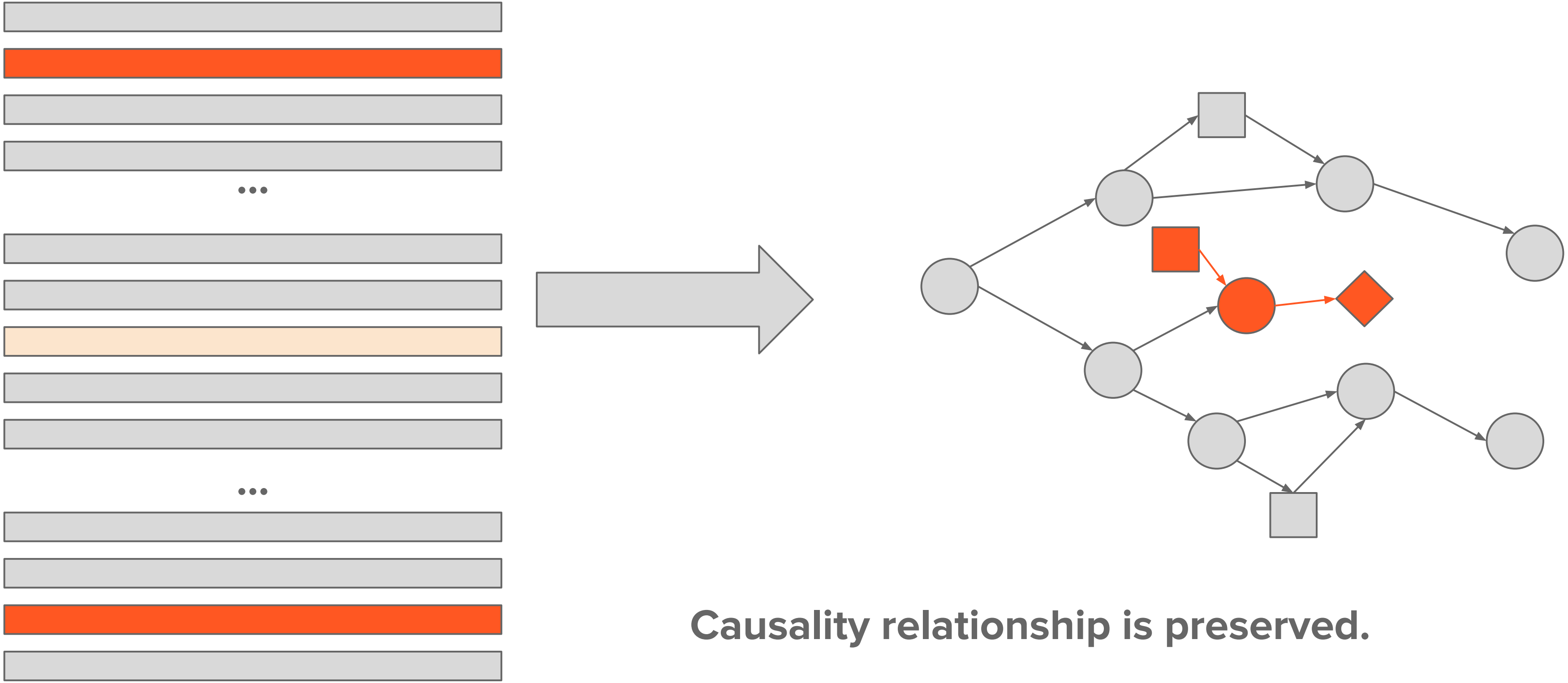
Masquerading as a benign event

# What is the intuition behind PIDS?



Attacks spaced in time

# What is the intuition behind PIDS?



Causality relationship is preserved.

# Two potential approaches

**Signature-based detection**

Match know malicious graph patterns

+ Higher precision

- Only detect known pattern

**Anomaly-based detection**

Detect pattern that deviate from normal behavior

+ Can detect unknown patterns

- Lower precision

# Two potential approaches

**Signature-based detection**

Match know malicious graph patterns

+ Higher precision
- Only detect known pattern

**Anomaly-based detection**

Detect pattern that deviate from normal behavior

+ Can detect unknown patterns
- Lower precision

**PIDS emerged to detect APT, zero-day exploit, etc.**

# Anomaly-based detection

- Train on past benign activity
    - Self-supervised learning
- During inference, perform anomaly detection
    - E.g., high reconstruction loss

# Limitation of SOTA methods

- Security analysts are overwhelmed with false positives
    - Alert Fatigue - Wajih et al., NDSS 2019
    - Burn out - Chandran et al., SOUPS 2015

# Our goal


- Security analysts are overwhelmed with false positives
  - Alert Fatigue - Wajih et al., NDSS 2019
  - Burn out - Chandran et al., SOUPS 2015

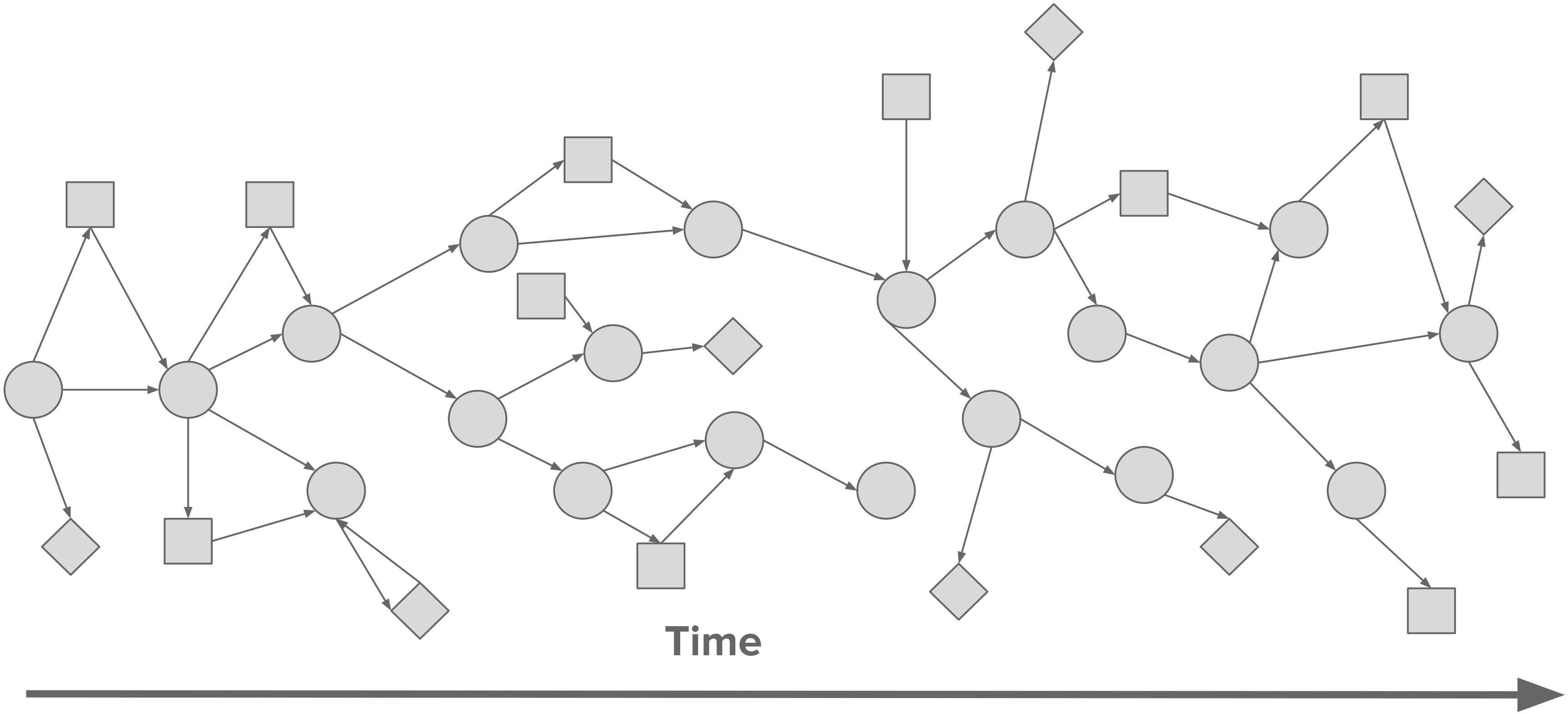
**Reduce the amount of information they need to go through**

# Understanding Attribution Quality

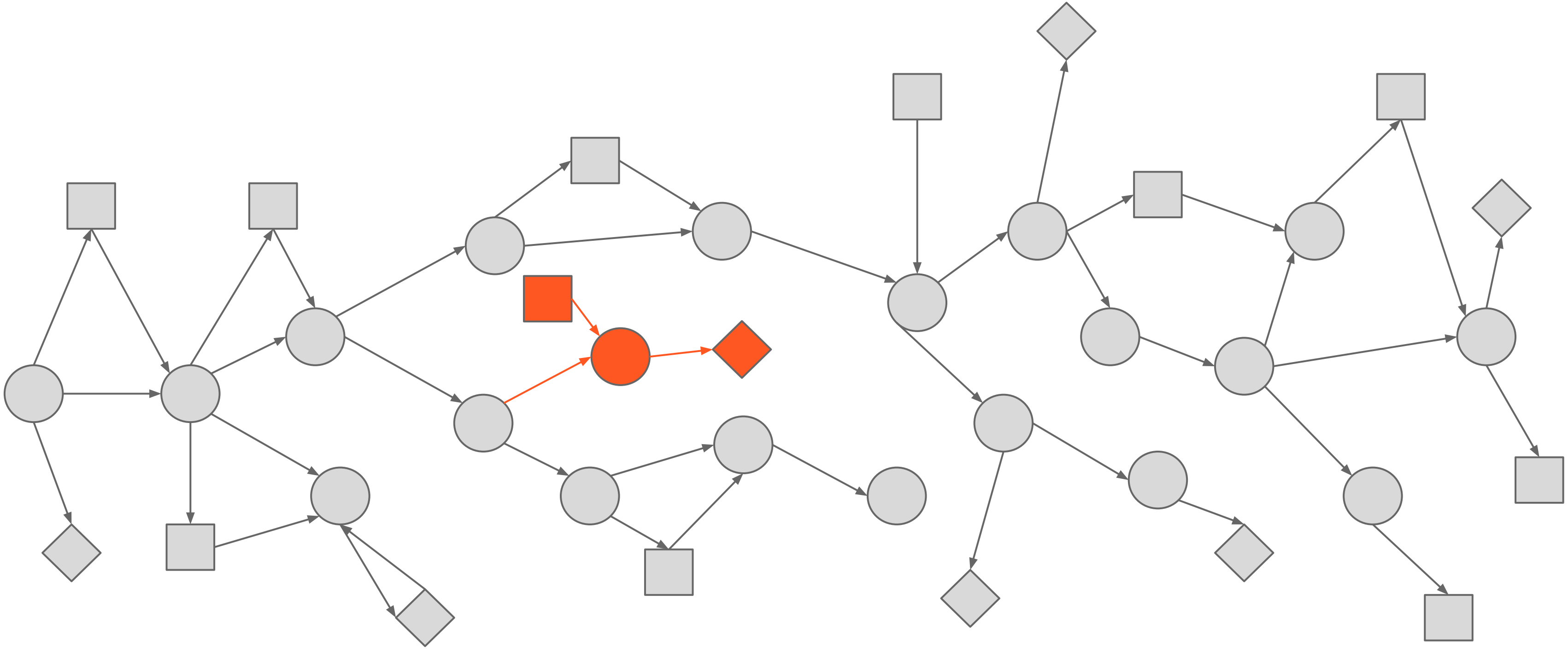
# Definition

*Attribution Quality* refers to the **amount of effort** required from a human analyst to investigate an IDS's detection output, uncover the root causes of an attack, and dismiss potential false alarms.

# Assume a provenance graph

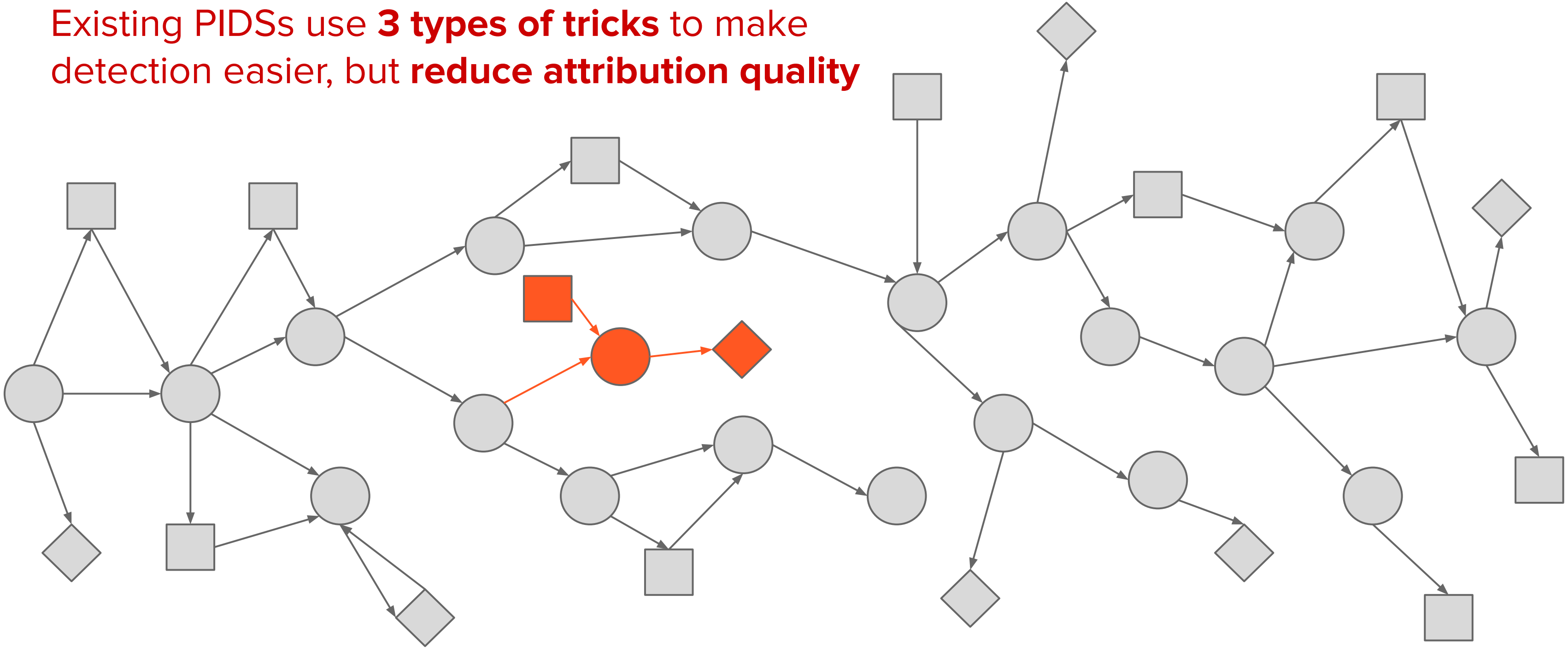


# Assume an attack



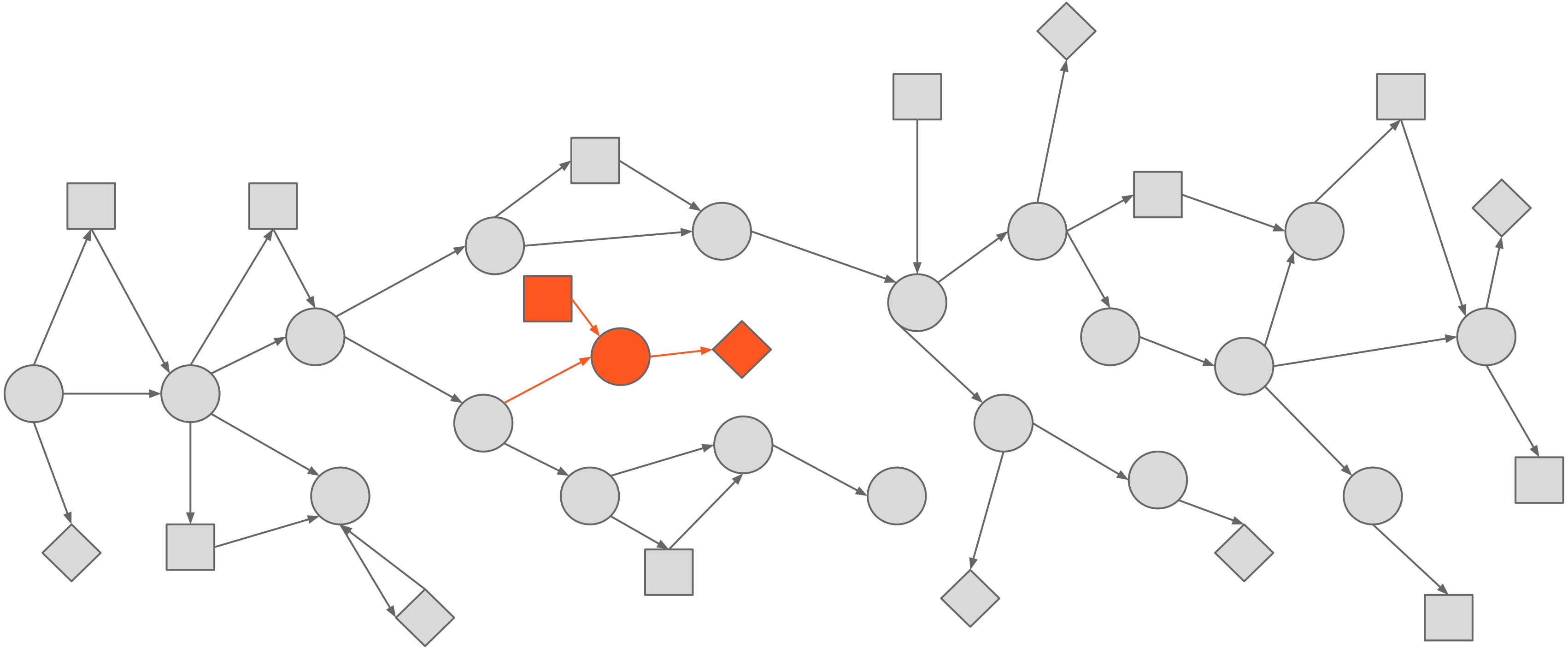
# Assume an attack

Existing PIDSs use **3 types of tricks** to make detection easier, but **reduce attribution quality**



# 1. Neighborhood Attribution

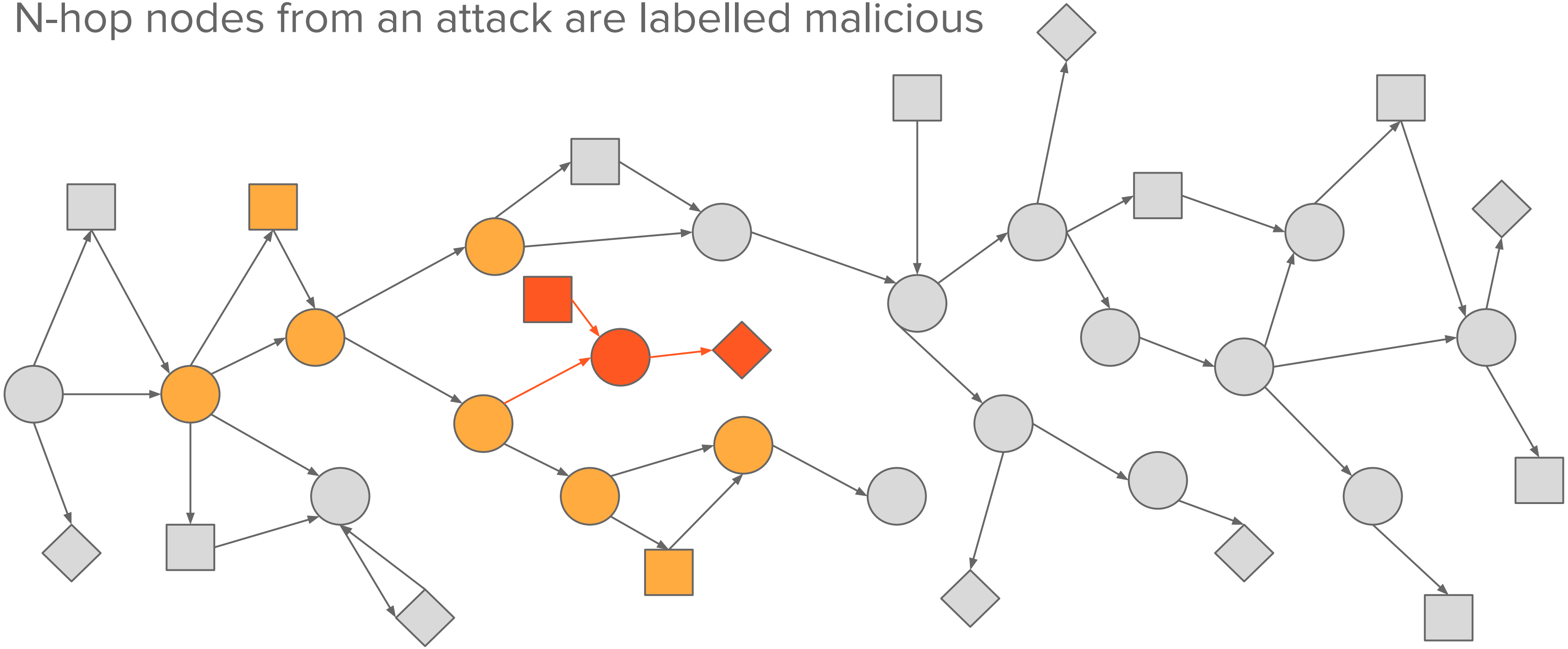
Concerned PIDSs:
**Flash**, S&P 2024
**MAGIC**, USENIX Sec 2024
**ThreaTrace**, TIFS 2022



# 1. Neighborhood Attribution

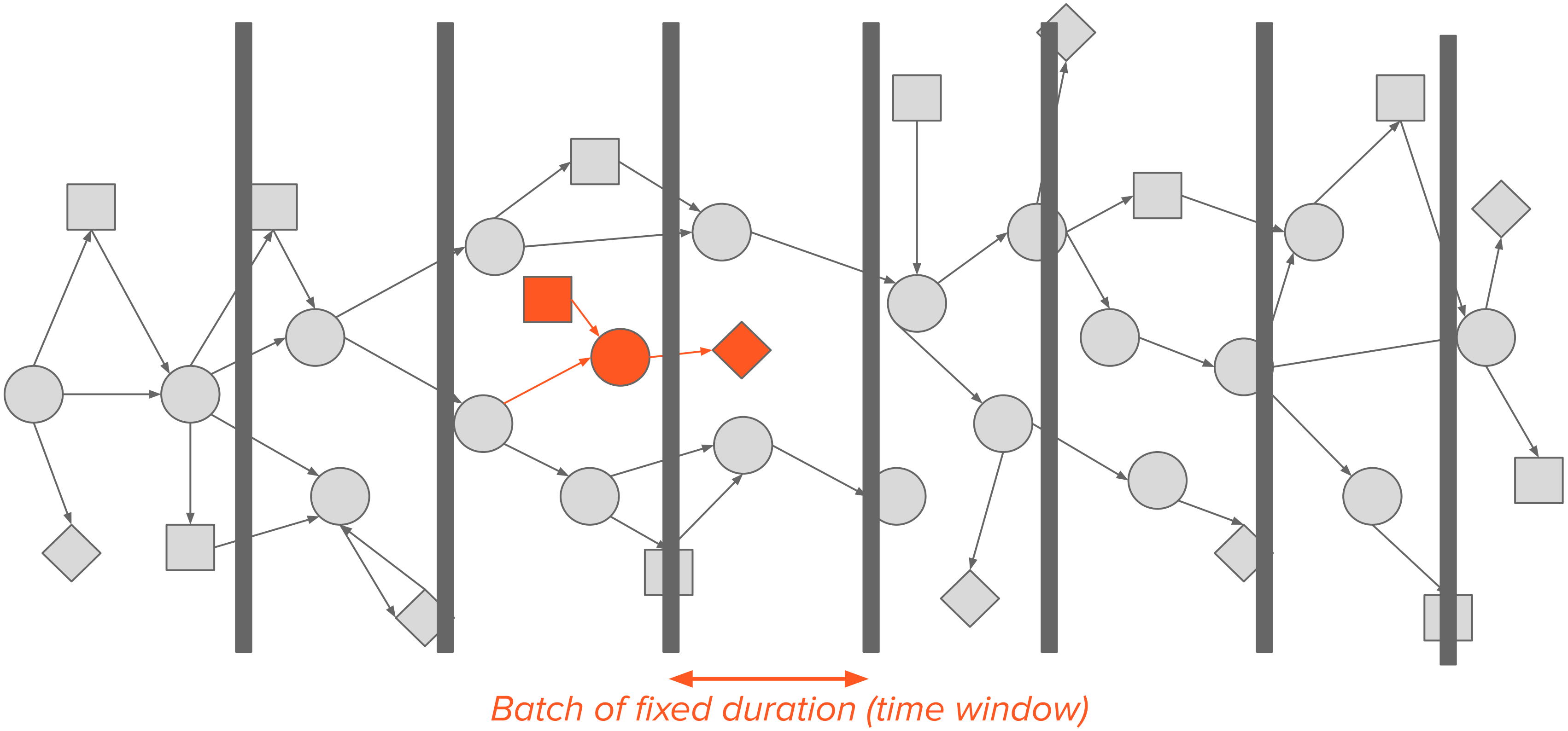
Concerned PIDSs:
**Flash**, S&P 2024
**MAGIC**, USENIX Sec 2024
**ThreaTrace**, TIFS 2022

N-hop nodes from an attack are labelled malicious



# 2. Batch Attribution

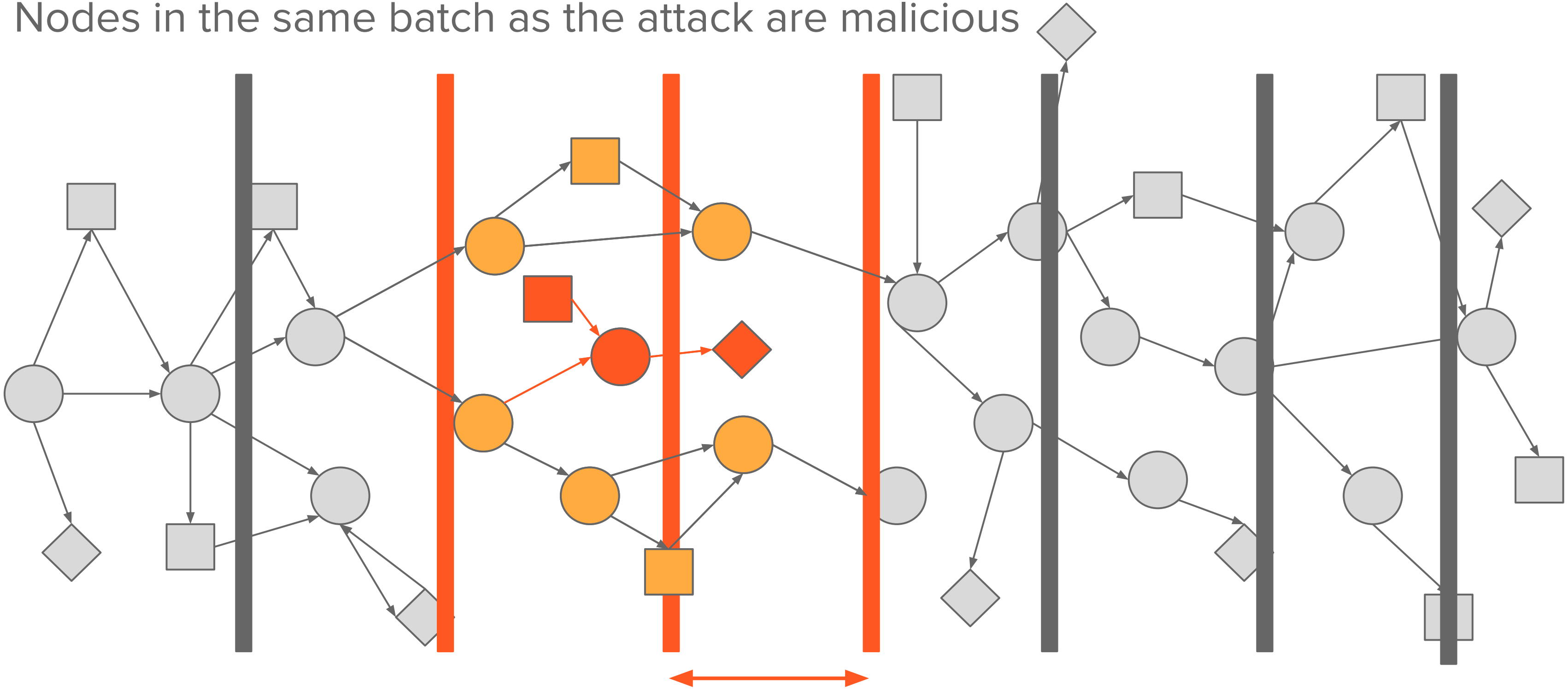
Concerned PIDSs:
**Kairos**, S&P 2024
**EdgeTorrent**, RAID's 2023



# 2. Batch Attribution

Concerned PIDSs:
**Kairos**, S&P 2024
**EdgeTorrent**, RAID's 2023

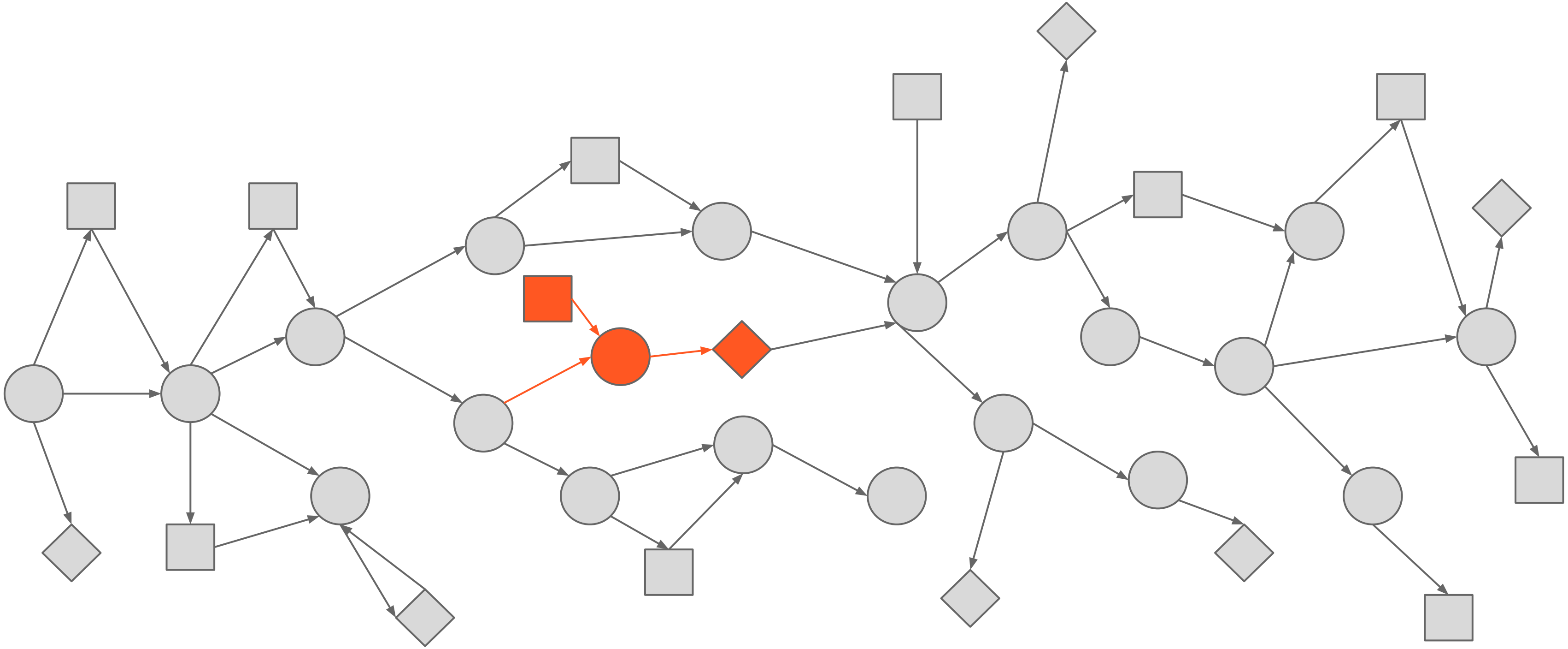
Nodes in the same batch as the attack are malicious



*Batch of fixed duration (time window)*

# 3. Source Attribution

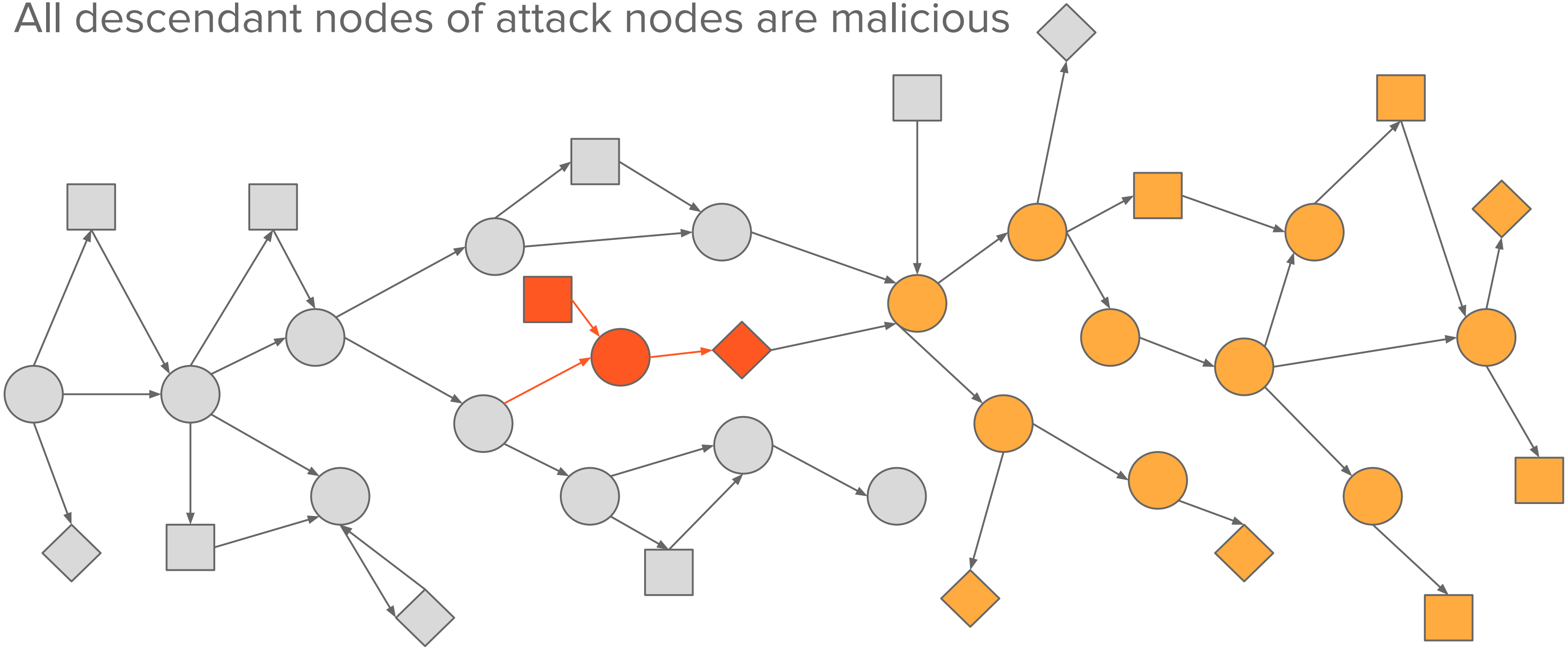
Concerned PIDS:
**R-Caid**, S&P 2024



# 3. Source Attribution

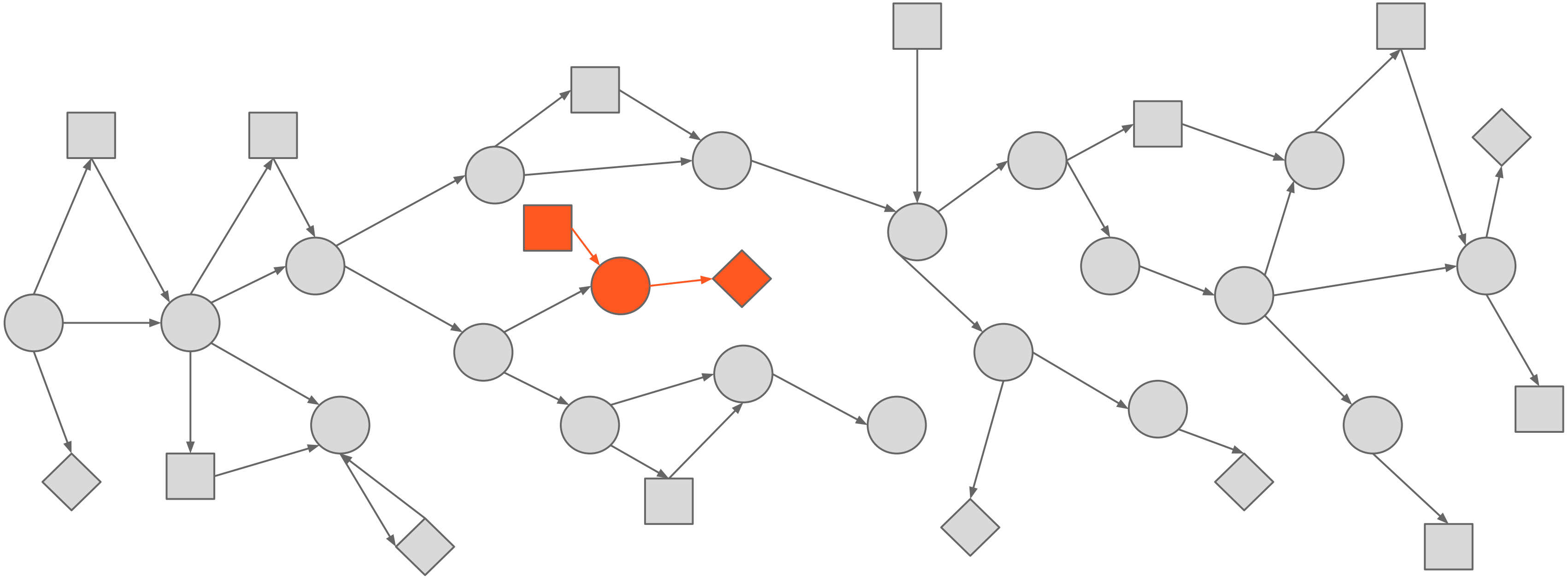
Concerned PIDS:
**R-Caid**, S&P 2024

All descendant nodes of attack nodes are malicious



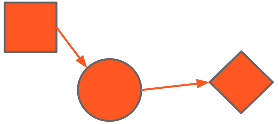
# Our Strategy

No tricks, we directly use the attack nodes from dataset ground truths



# Our Strategy

No tricks, we directly use the attack nodes from dataset ground truths



- Much harder detection (few nodes)
- Concise detection reports (less work for analysts)

# Idealized Detection Performance

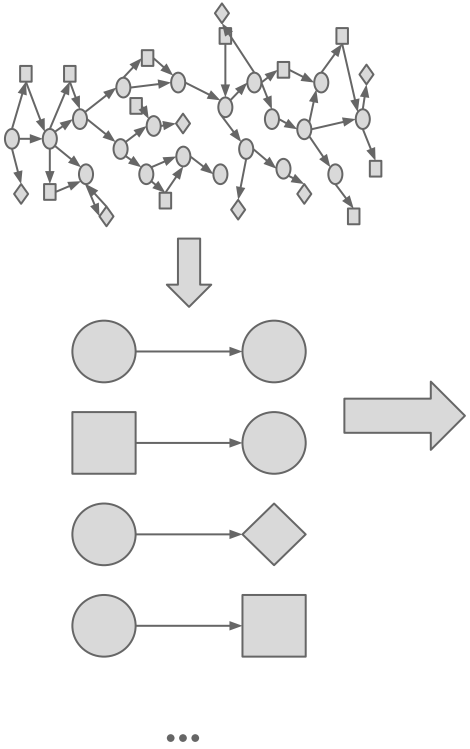
- Assuming perfect detection based on their design
- Past systems overwhelm security analyst with large alerts

**Number of attack nodes to detect per attribution strategy:**

| Dataset | Total Nodes | Neighborhood | Batch | Source | Node-level (Ours) |
|---|---|---|---|---|---|
| **E5-CADETS** | 7,632,792 | 20,524 | 717,783 | 401,065 | 123 |
| **E5-THEIA** | 1,728,121 | 162,714 | 61,368 | 9,374 | 69 |
| **E5-CLEARSCOPE** | 326,338 | 48,488 | 8,636 | 1,020 | 51 |

# Orthrus: a PIDS for node-level detection

# Orthrus design



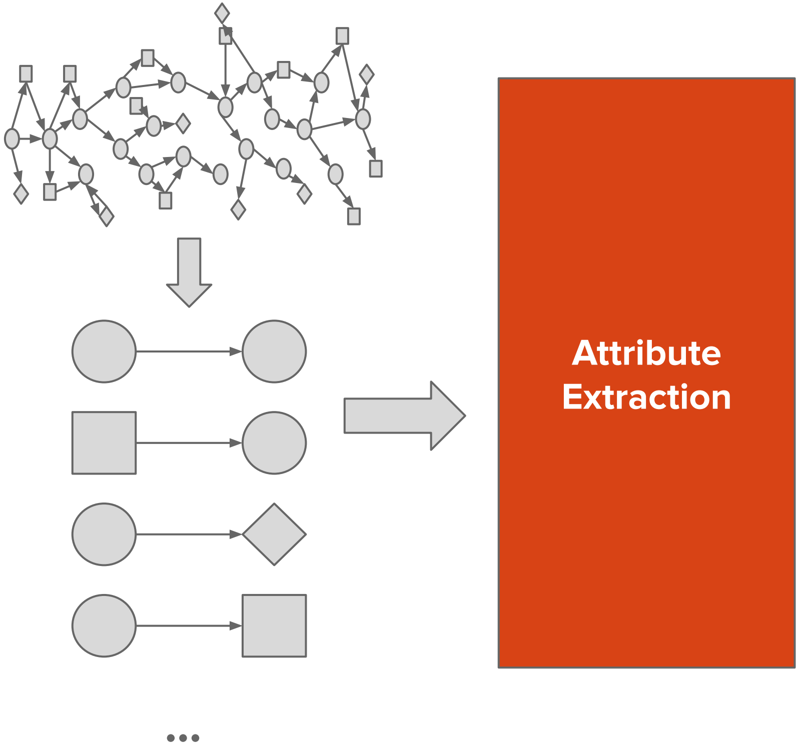
Attribute Extraction

Featurization

Graph Encoder

Detection Task

# Orthrus design

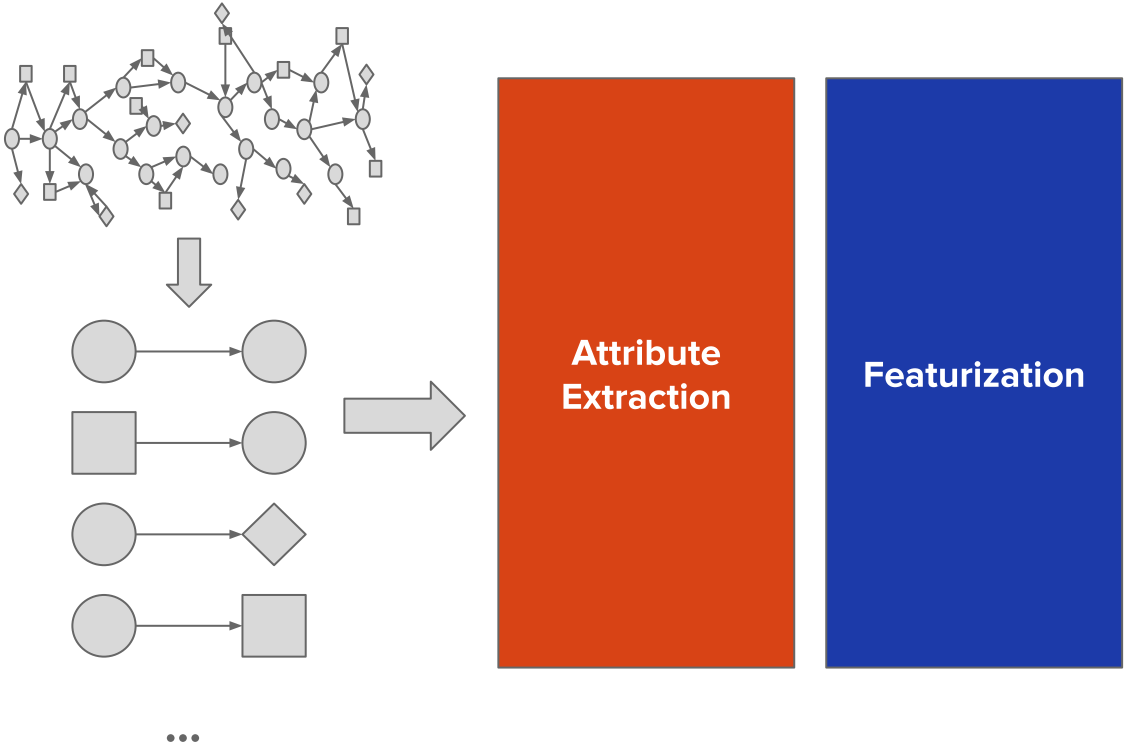


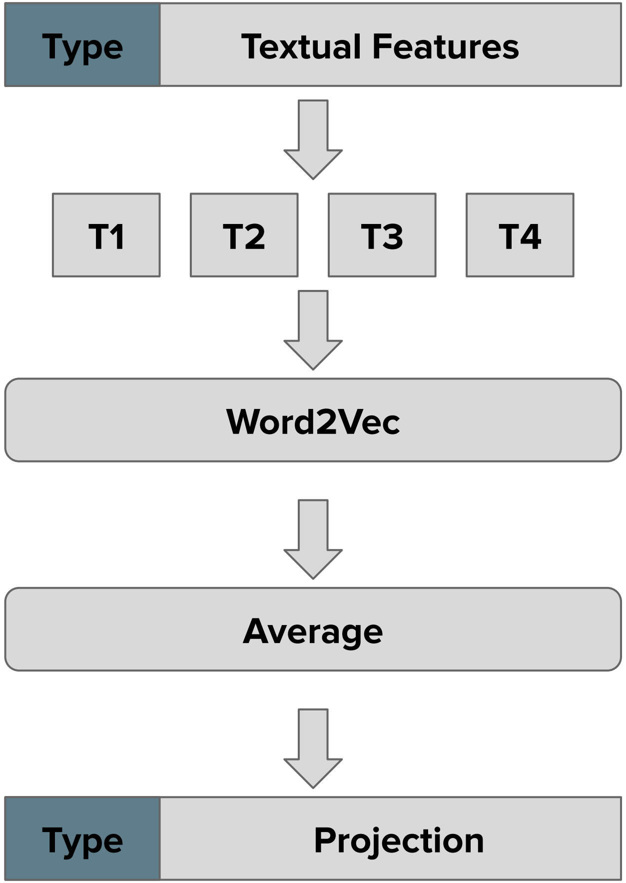

**File:** type + path (e.g. */etc/passwd)*

**Process:** type + cmd line (e.g. *ls -l -t -r)*

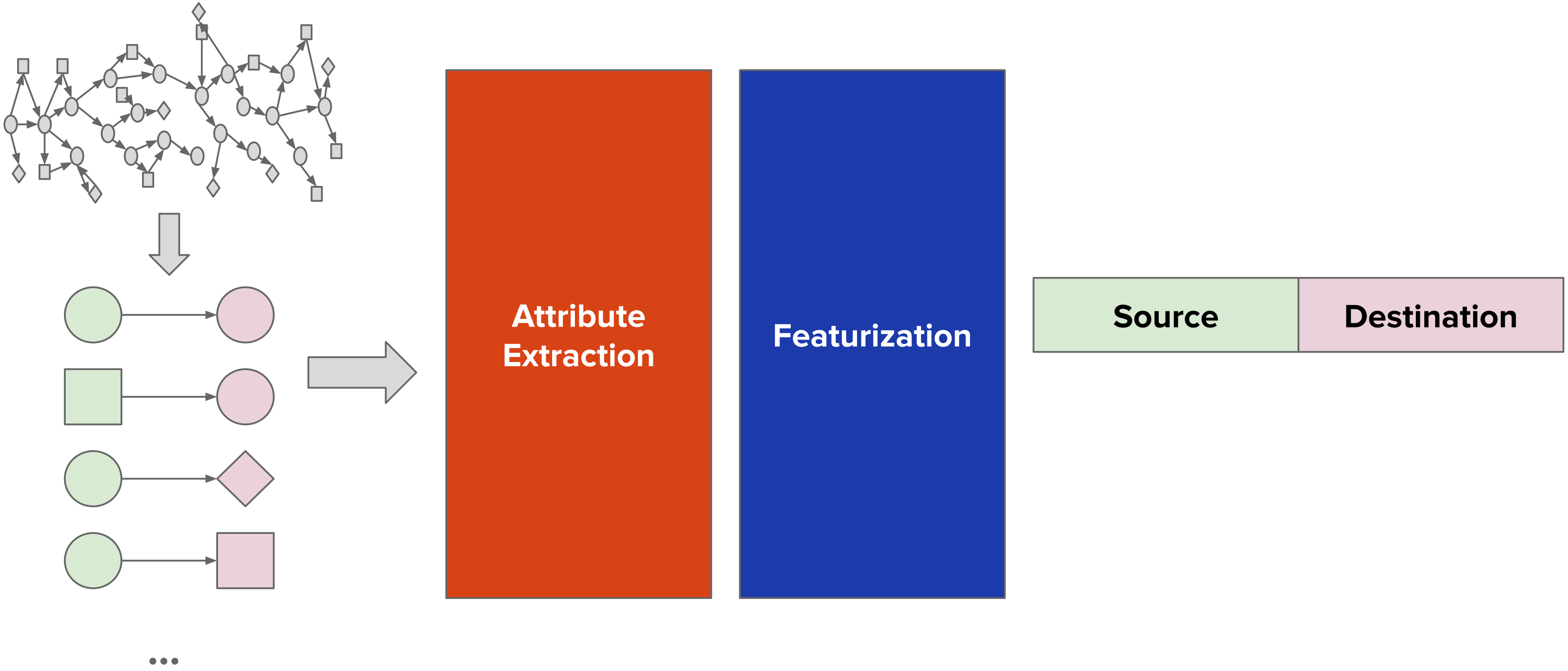
**Netflow:** type + IP + port (e.g. *192.168.1.2 80)*

# Orthrus design

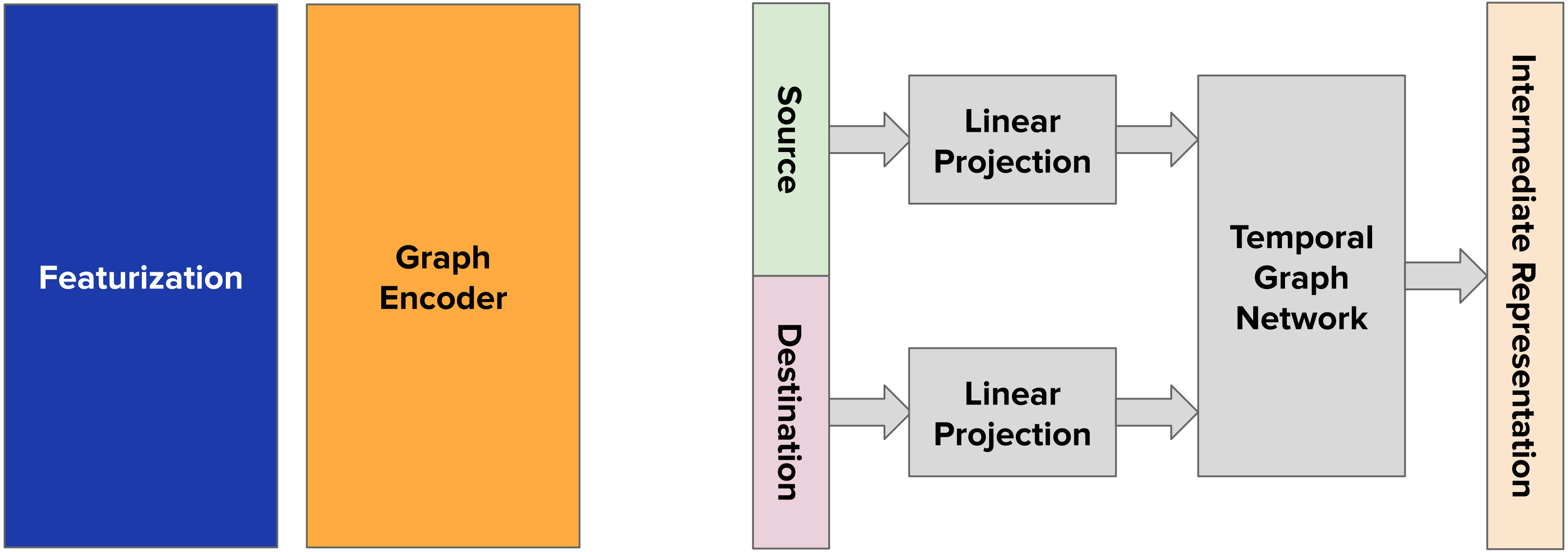


# Orthrus design

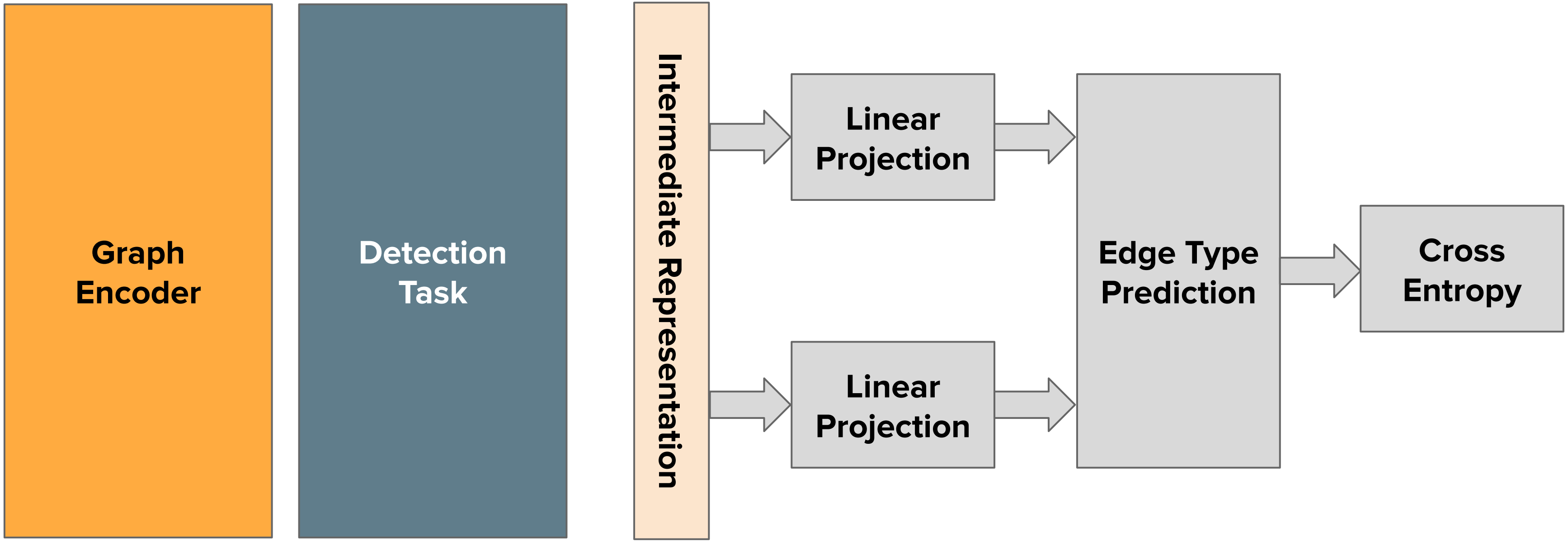


# Orthrus design




We made some changes to TGN to fit our problem.

# Orthrus design



# Evaluation

# Baselines

- We picked **5** graph learning-based **SOTA PIDSs**
- Hyperparameters tuned consistently
- Experiments based on our node-level labels

# Results

## E3 Datasets

| System | E3-CADETS (TP / FP / Prec) | E3-THEIA | E3-CLEARSCOPE |
|---|---|---|---|
| Kairos | 0 / 9 / 0.00 | 4 / 0 / **1.00** | 0 / 7 / 0.00 |
| Threatrace | 61 / 252k / 0.00 | 88 / 672k / 0.00 | 41 / 88k / 0.00 |
| SIGL | 0 / 80 / 0.00 | 1 / 29 / 0.03 | 1 / 11k / 0.00 |
| MAGIC | 63 / 80k / 0.00 | 115 / 395k / 0.00 | 40 / 102k / 0.00 |
| Flash | 13 / 2.4k / 0.01 | 22 / 32k / 0.00 | 0 / 15k / 0.00 |
| **Orthrus** | **10 / 0 / 1.00** | **8 / 0 / 1.00** | **1 / 1 / 0.50** |

## E5 Datasets

| System | E5-CADETS (TP / FP / Prec) | E5-THEIA | E5-CLEARSCOPE |
|---|---|---|---|
| Kairos | 0 / 6 / 0.00 | 0 / 2 / 0.00 | 1 / 7 / 0.25 |
| Threatrace | 91 / 3M / 0.00 | 66 / 739k / 0.00 | 41 / 142k / 0.00 |
| SIGL | 0 / 66 / 0.00 | 0 / 23 / 0.00 | 10 / 63 / 0.14 |
| MAGIC | 123 / 3M / 0.00 | 1 / 297k / 0.00 | 51 / 139k / 0.00 |
| Flash | 45 / 34k / 0.00 | 43 / 296k / 0.00 | 15 / 4.6k / 0.00 |
| **Orthrus** | **1 / 5 / 0.17** | **2 / 0 / 1.00** | **2 / 3 / 0.22** |

# Results

- Orthrus can detect **all attacks** in each dataset
- It detects a few nodes only, but with high precision
- Attack reconstruction algorithms for provenance graphs can be used

# Toward More Practical PIDSs

# PIDSMaker
## A framework to design PIDSs

We reimplemented 7 SOTA PIDSs in a **unified** (open source) **framework**

1. Kairos
2. ThreaTrace
3. NodLink
4. Magic
5. Flash
6. R-Caid
7. SIGL
8. Orthrus


```yaml
featurization:
 feat_training:
   emb_dim: 16
   training_split: all
   used_method: hierarchical_hashing
detection:
 gnn_training:
   used_method: default
   use_seed: True
   deterministic: False
   num_epochs: 12
   patience: 3
   lr: 0.00005
   weight_decay: 0.01
   node_hid_dim: 100
   node_out_dim: 100
   grad_accumulation: 1
   encoder:
     dropout: 0.0
     used_methods: graph_attention,tgn
     graph_attention:
       activation: relu
       num_layers: 2
     tgn:
       tgn_memory_dim: 100
       tgn_time_dim: 100
       use_node_feats_in_gnn: False
       use_memory: True
       use_time_order_encoding: False
       project_src_dst: True
   decoder:
     used_methods: predict_edge_type
     predict_edge_type:
       balanced_loss: False
       use_triplet_types: False
       decoder: edge_mlp
       edge_mlp:
         architecture_str: linear(2) | dropout(0.5) | tanh |
linear(0.25) | dropout(0.25) | tanh | linear(0.25) | dropout(0.5) | tanh
         src_dst_projection_coef: 2
 evaluation:
   used_method: node_evaluation
   node_evaluation:
     threshold_method: max_val_loss
     use_dst_node_loss: True
     use_kmeans: False
```


10 text encoders

YAML

11 graph encoders

6 decoders

# PIDSMaker
## A framework to design PIDSs

**Goal**:

- Combinatorial architecture search
- Consistent evaluation across papers

```yaml
featurization:
 feat_training:
   emb_dim: 16
   training_split: all
   used_method: hierarchical_hashing
detection:
 gnn_training:
   used_method: default
   use_seed: True
   deterministic: False
   num_epochs: 12
   patience: 3
   lr: 0.00005
   weight_decay: 0.01
   node_hid_dim: 100
   node_out_dim: 100
   grad_accumulation: 1
   encoder:
     dropout: 0.0
     used_methods: graph_attention,tgn
     graph_attention:
       activation: relu
       num_layers: 2
     tgn:
       tgn_memory_dim: 100
       tgn_time_dim: 100
       use_node_feats_in_gnn: False
       use_memory: True
       use_time_order_encoding: False
       project_src_dst: True
   decoder:
     used_methods: predict_edge_type
     predict_edge_type:
       balanced_loss: False
       use_triplet_types: False
       decoder: edge_mlp
       edge_mlp:
         architecture_str: linear(2) | dropout(0.5) | tanh |
linear(0.25) | dropout(0.25) | tanh | linear(0.25) | dropout(0.5) | tanh
         src_dst_projection_coef: 2
 evaluation:
   used_method: node_evaluation
   node_evaluation:
     threshold_method: max_val_loss
     use_dst_node_loss: True
     use_kmeans: False
```

10 text encoders

YAML

11 graph encoders

6 decoders

# Addressing Existing Shortcomings

- We found 9 key shortcomings that hinder practicality of SOTA PIDSs

| System | SC1 | SC2 | SC3 | SC4 | SC5 | SC6 | SC7 | SC8 | SC9 |
|---|---|---|---|---|---|---|---|---|---|
| SIGL | | | ✔ | | | ✔ | | | |
| THREATRACE | | | | | | ✔ | | | |
| NODLINK | ✔ | | ✔ | | | ✔ | | | |
| MAGIC | | | | | | ✔ | | | |
| KAIROS | | | | | | ✔ | | | |
| FLASH | | | | | | ✔ | | | |
| R-CAID | | | ✔ | | | ✔ | | | |
| ORTHRUS | ✔ | ✔ | | | | | | | |

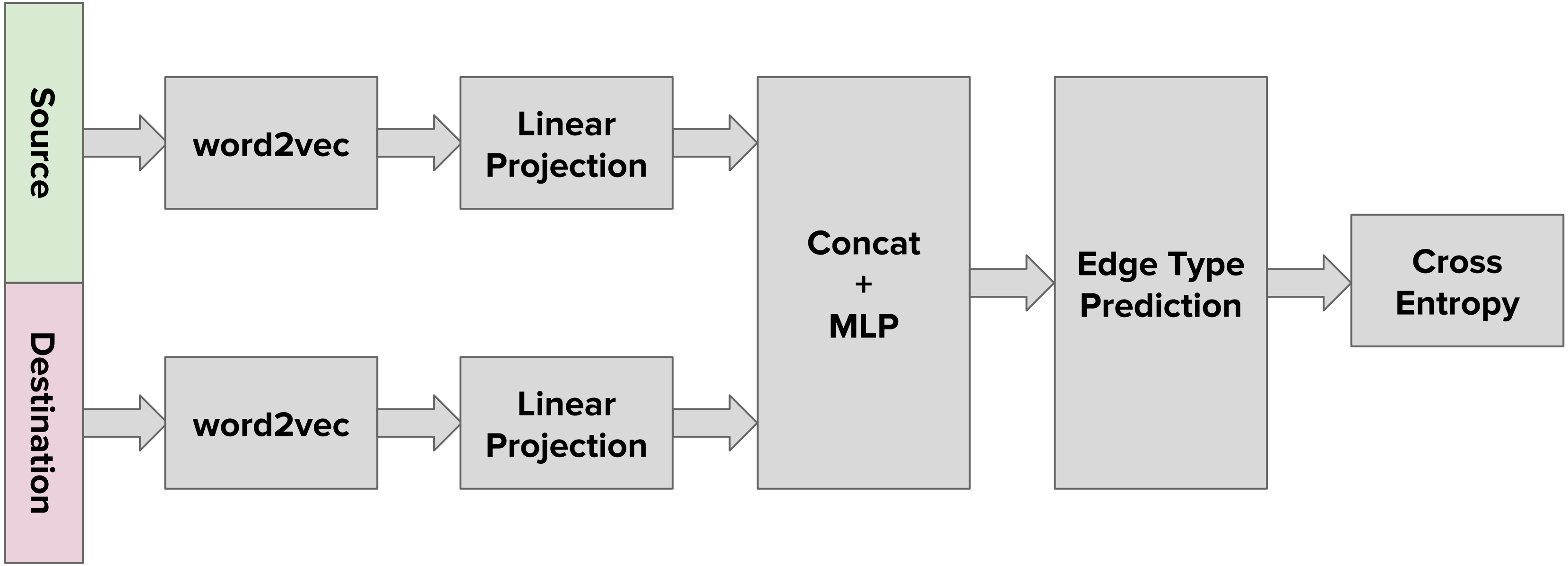
# Addressing Existing Shortcomings

- We found 9 key shortcomings that hinder practicality of SOTA PIDSs

| System | SC1 | SC2 | SC3 | SC4 | SC5 | SC6 | SC7 | SC8 | SC9 |
|---|---|---|---|---|---|---|---|---|---|
| SIGL | | | ✔ | | | ✔ | | | |
| THREATRACE | | | | | | ✔ | | | |
| NODLINK | ✔ | | ✔ | | | ✔ | | | |
| MAGIC | | | | | | ✔ | | | |
| KAIROS | | | | | | ✔ | | | |
| FLASH | | | | | | ✔ | | | |
| R-CAID | | | ✔ | | | ✔ | | | |
| ORTHRUS | ✔ | ✔ | | | | | | | |
| VELOX | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

We designed **Velox**, a PIDS that addresses all these shortcomings

# Velox design

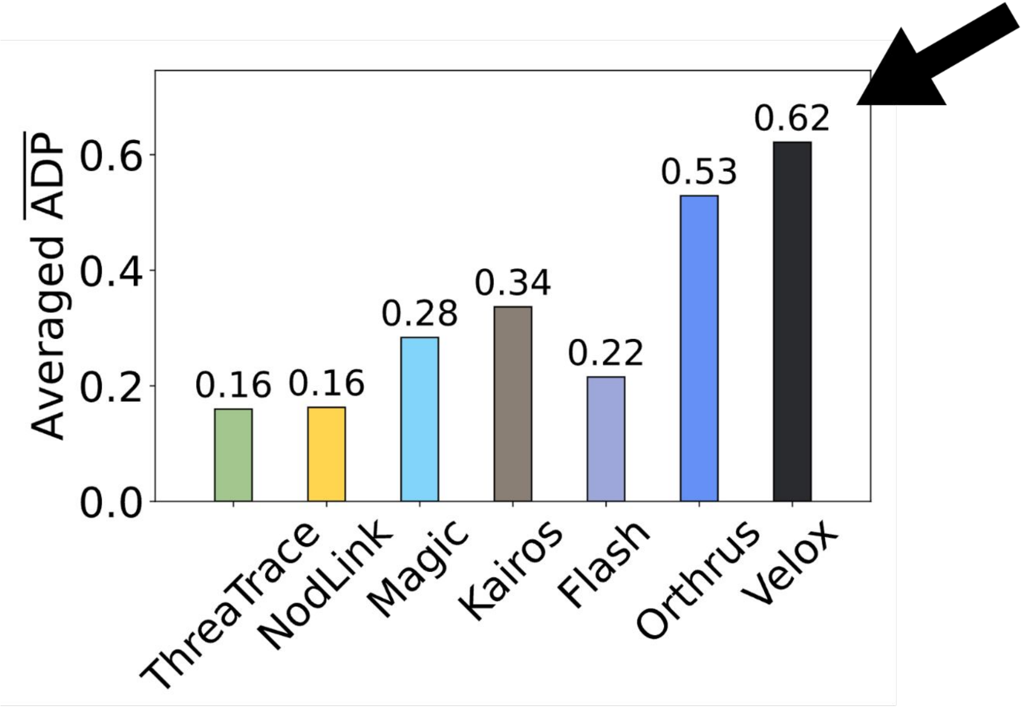



- This design has been selected after **~453 days of GPU compute**
- Unlike all other PIDSs, no complex graph encoder is used

# An Unexpected Discovery

Velox has the highest average **Attack Detection Precision (ADP)** across 9 DARPA datasets

**ADP:** measures the ability to detect all attacks in a dataset with high precision




*ADP averaged across 9 DARPA datasets*

# Discussion

- Complex architectures are not always needed
- APTs can be detected using textual features features only!
  - Realistic attacks?

# Future Directions

# Limitations & Future Work

| Limitation | | Future Work |
|---|---|---|
| • Benign anomalies could be false positives | → | Design datasets with benign anomalies |
| • Training instability | → | Add some supervision during training |

# Want to play with PIDSMaker?

- **PIDSMaker is open source**
- 8 PIDSs (incl. Orthrus & Velox)
- 9 pre-processed datasets
- You can build your own models

https://github.com/ubc-provenance/PIDSMaker

README | Contributing | Apache-2.0 license

# PIDSMaker

docs online | DOI 10.5281/zenodo.15603122 | license Apache-2.0 | release v1.0.1 | Stars 45

Paper | Documentation | Installation

The first framework designed to build and experiment with provenance-based intrusion detection systems (PIDSs) using deep learning architectures. It provides a single codebase to run most recent state-of-the-arts systems and easily customize them to develop new variants.

**Currently supported PIDSs**:

- **Velox** (USENIX Sec'25): Sometimes Simpler is Better: A Comprehensive Analysis of State-of-the-Art Provenance-Based Intrusion Detection Systems
- **Orthrus** (USENIX Sec'25): ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems
- **R-Caid** (IEEE S&P'24): R-CAID: Embedding Root Cause Analysis within Provenance-based Intrusion Detection
- **Flash** (IEEE S&P'24): Flash: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning
- **Kairos** (IEEE S&P'24): Kairos: Practical Intrusion Detection and Investigation using Whole-system Provenance
- **Magic** (USENIX Sec'24): MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning
- **NodLink** (NDSS'24): NODLINK: An Online System for Fine-Grained APT Attack Detection and Investigation
- **ThreaTrace** (IEEE TIFS'22): THREATRACE: Detecting and Tracing Host-Based Threats in Node Level Through Provenance Graph Learning

# Collaborations

## Want to work with us?

We are looking for motivated students to collaborate!

- Fully-funded PhD offers are still available in our team (UBC x Amazon)
- Possible to join as a visiting research student too

Drop me an e-mail: tristan.bilot@universite-paris-saclay.fr