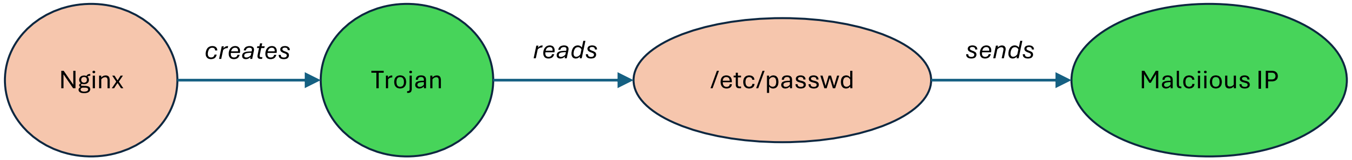
System-level Intrusion Detection with Graph Neural Networks

Tristan Bilot

06/21/2024

Motivating Example

- An attacker leverages a vulnerable version of Nginx to **gain elevated privileges** on a victim's machine
- Passwords stored in **/etc/passwd** are **stolen**
- These passwords are **sent to a Command & Control** (C2) server



Motivating Example

Goal:

- Detect such attacks at the system-level
- Without any labels
- On large-scale data
- In a near real-time setting
- With low false positive rate

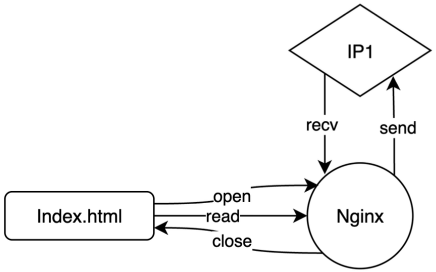
Provenance Graphs

- **System Provenance** records causality relationships between system objects (e.g. Processes, Files, or Network sockets, etc.) and represents system execution flow as a **directed** and **attributed graph**
- We use **provenance graphs** to model the interactions between system entities

System Events

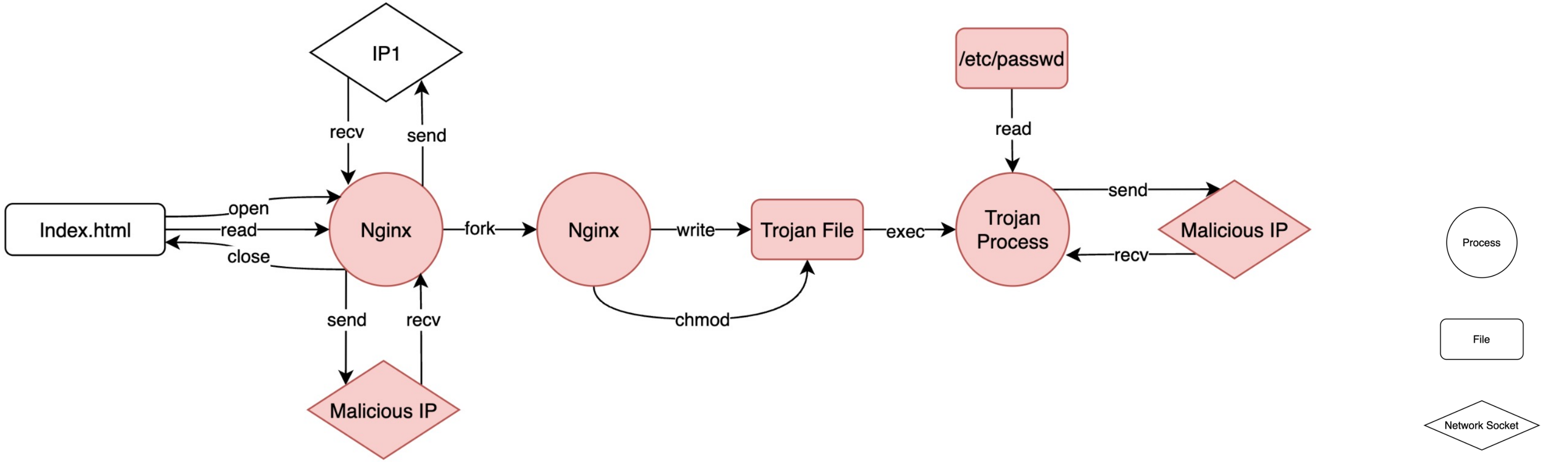
<Timestamp 1> Nginx, receive, IP1
<Timestamp 2> Nginx, open, index.html
<Timestamp 3> Nginx, read, index.html
<Timestamp 4> Nginx, send, IP1
<Timestamp 5> Nginx, close, Index.html
……

System Provenance Graph



Provenance Graphs

- The **previous attack** can be easily represented as a **provenance graph**
- **Nodes** represent **system entities**
- **Edges** represent **system calls**



Provenance-based Intrusion Detection System

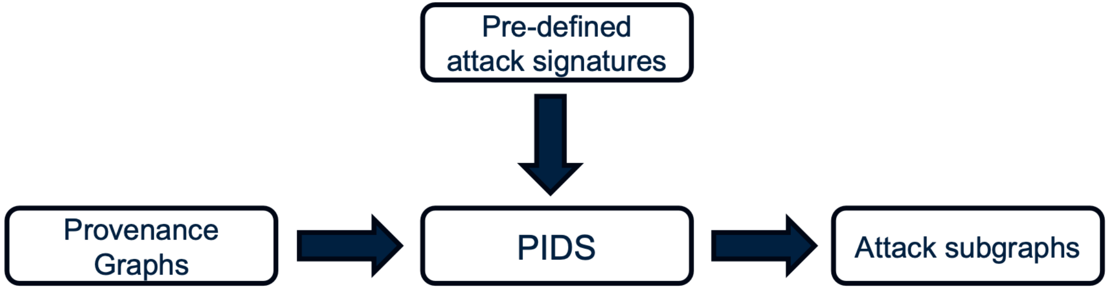
- A Provenance-based Intrusion Detection System (**PIDS**) aims to detect the malicious system behaviors in provenance graphs
- Category:
  - **Signature-based PIDS**
  - **Anomaly-based PIDS**

Provenance-based Intrusion Detection System

- A Provenance-based Intrusion Detection System (**PIDS**) aims to detect the malicious system behaviors in provenance graphs

Signature-based PIDS

- **Advantage**: The PIDS can report the complete attack subgraphs from known attacks
- **Disadvantage**: Any unknown attacks are unable to be detected

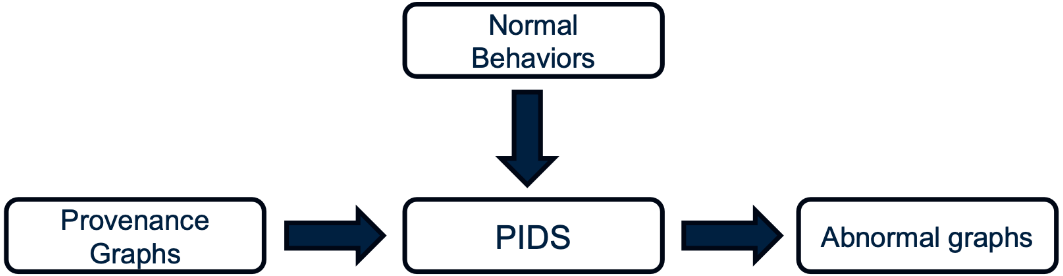


Provenance-based Intrusion Detection System

- A Provenance-based Intrusion Detection System (**PIDS**) aims to detect the malicious system behaviors in provenance graphs

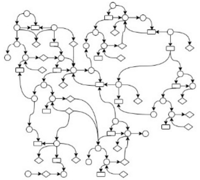
Anomaly-based PIDS

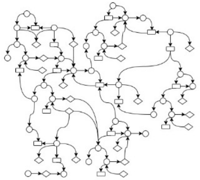
- **Advantage**: The PIDS can detect unknown attacks
- **Disadvantage:** Higher false positives as it detects anomalies

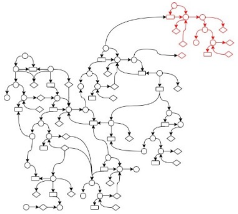


Motivation

Prior work neglects the quality of detection report generated by PIDSes.







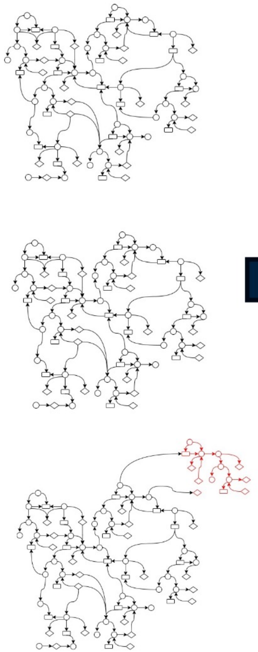
StreamSpot (KDD 16') [1]
or
Unicorn (NDSS 20') [2]

Detection reports:
Graph 1: normal
Graph 2: normal
Graph 3: abnormal

All colored nodes are considered as anomalies by the systems

Motivation

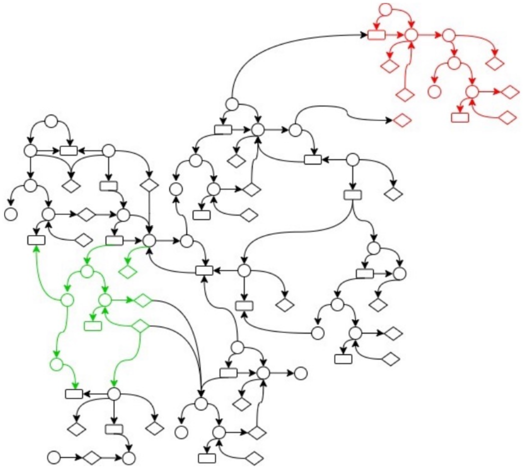
Prior work neglects the quality of detection report generated by PIDSes.



ThreaTrace (TIFS 22') [3]

NodLink (NDSS 24') [4]

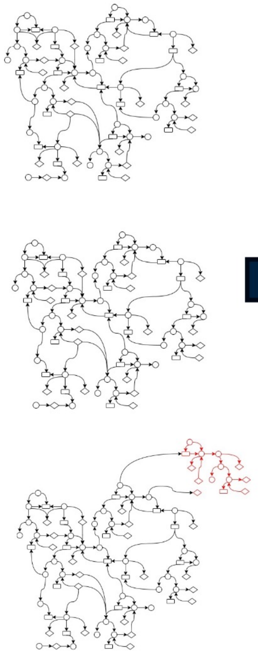
Detection reports:
Graph 3: abnormal



All colored nodes are considered as anomalies by the systems

Motivation

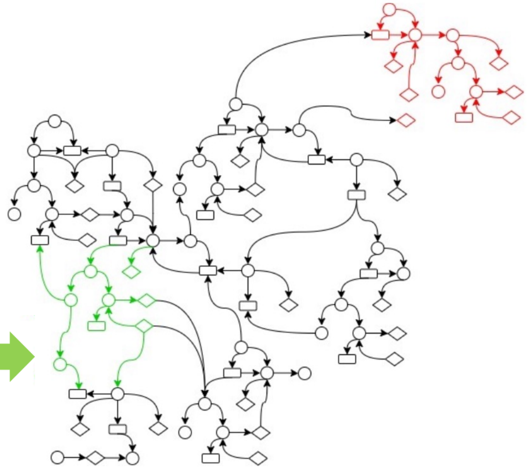
Prior work neglects the quality of detection report generated by PIDSes.



ThreaTrace (TIFS 22') [3]

NodLink (NDSS 24') [4]

Detection reports:
Graph 3: abnormal



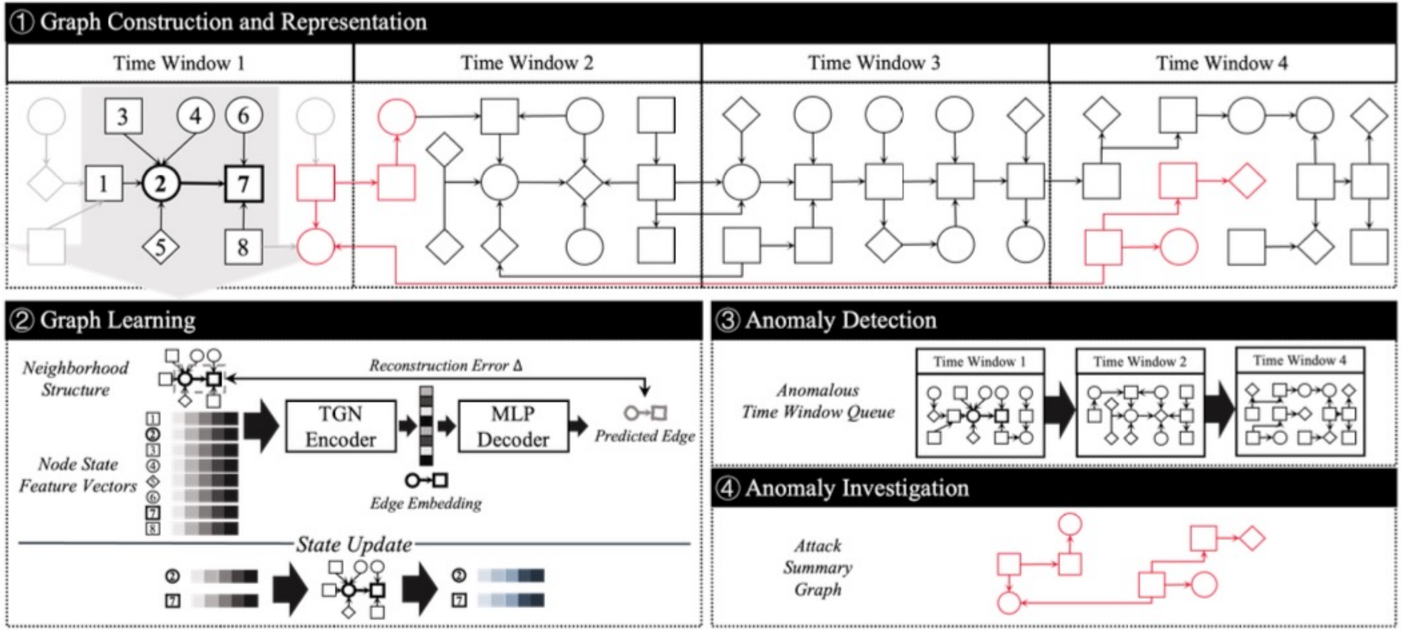
False
Positives

All colored nodes are considered as anomalies by the systems

Proposition: Kairos

- **Kairos (S&P 2024)** [5] is a PIDS based on Graph Neural Networks (**GNNs**)
- It offers a more **fine-grained** summary of detected attacks
- It captures **long-term dependencies** within attacks
- It has a **low false positive rate**

Proposition: Kairos



1. Graph construction

- We encode information within the graph structure using **node and edge features**
- Node features
  - **Process**: executable name
  - **File**: file name
  - **Network socket**: src+dst IP addess, src+dst port
- Edge features
  - One-hot encoded system call type
- String features (e.g. file names) are transformed into vectors using **hierarchical feature hashing [6]**

Subject	Object	Relationships	Entity Attributes
Process	Process	Start, Close, Clone	Image pathname
	File	Read, Write, Open, Exec	File pathname
	Socket	Send, Receive	Src/Dst IP/port

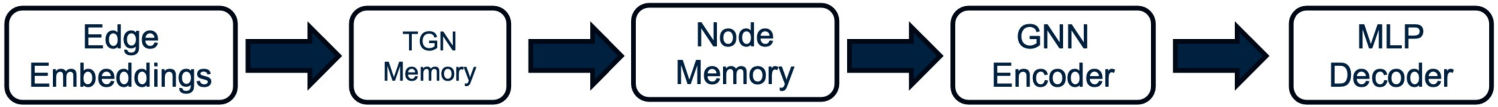
1. Graph Construction

- After generating **node feature vectors**, Kairos generates an **edge feature vector** based on its source node, destination node and edge type

$$Vec_{edge} = CONCAT(\ Vec_{Src},\ One\text{-}Hot\text{-}Encoding_{EdgeType},\ Vec_{Dst}\ )$$

2. Graph Learning

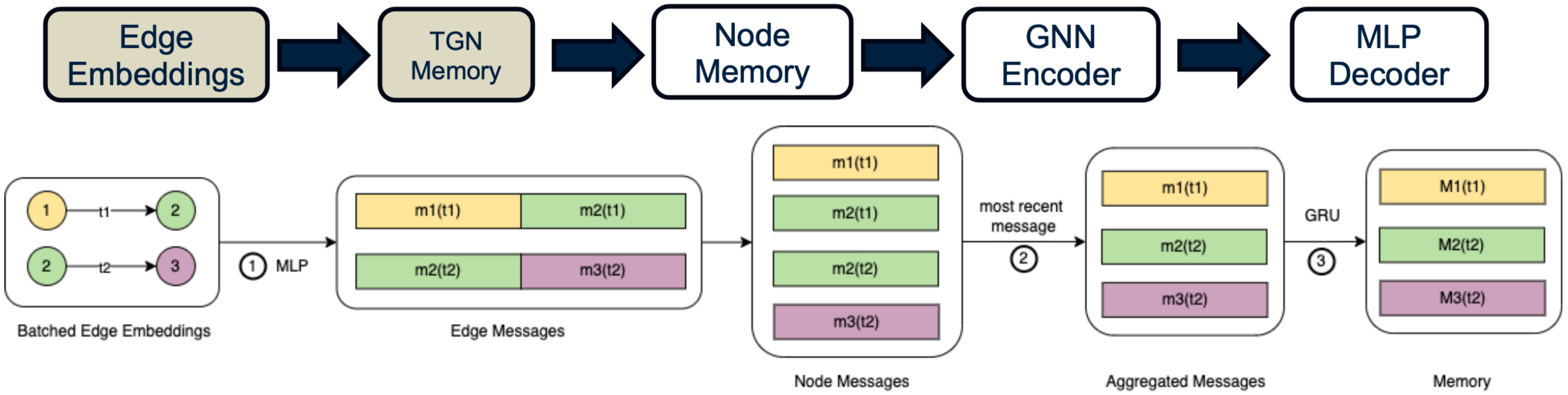
Framework: Temporal Graph Network [7]



- **TGN Memory**: Updates node embeddings given the temporal information of a node.
- **GNN Encoder**: Aggregates node embeddings based on graph structure to generate embeddings.
- **MLP Decoder**: Predicts the edge type between any two connected nodes based on their embeddings.

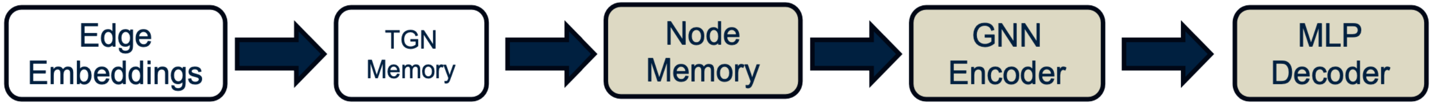
2. Graph Learning

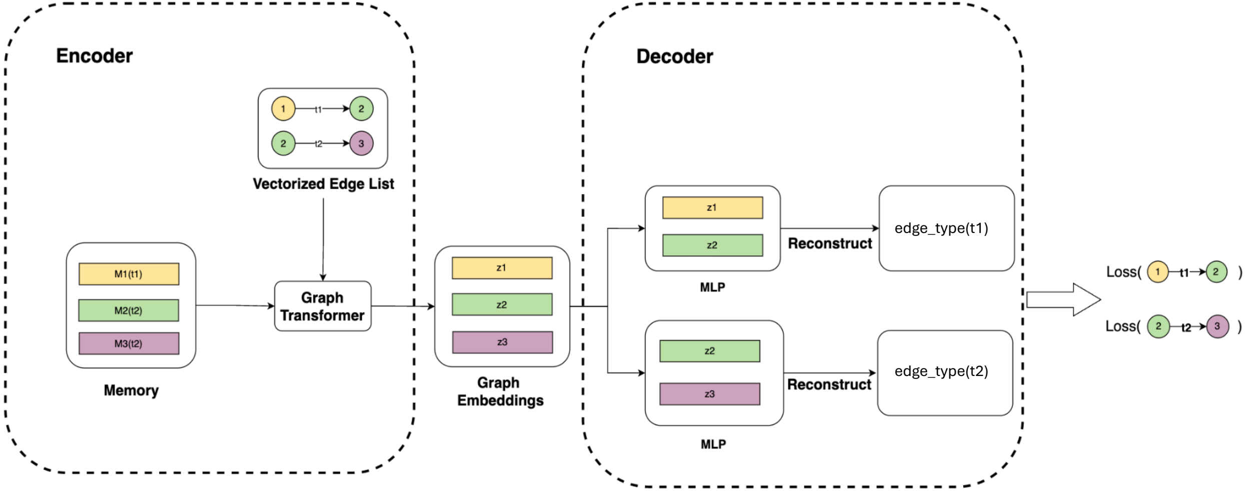
Framework: Temporal Graph Network [7]



2. Graph Learning

Framework: **Temporal Graph Network [7]**



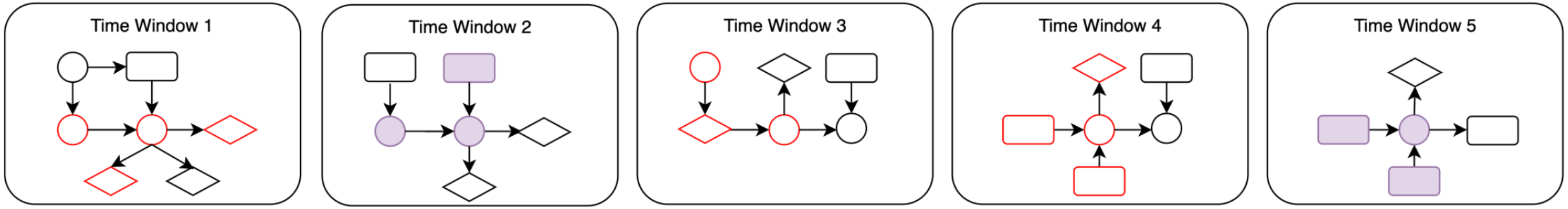


3. Anomaly Detection

- **Threshold** = Max(Validation Losses)
- **Detection**:
  - **Edge**: If Loss(edge) > Threshold, the edge will be alerted as abnormal.
  - **Node**: The source and destination nodes of abnormal edges will be alerted.

4. Alert Correlation

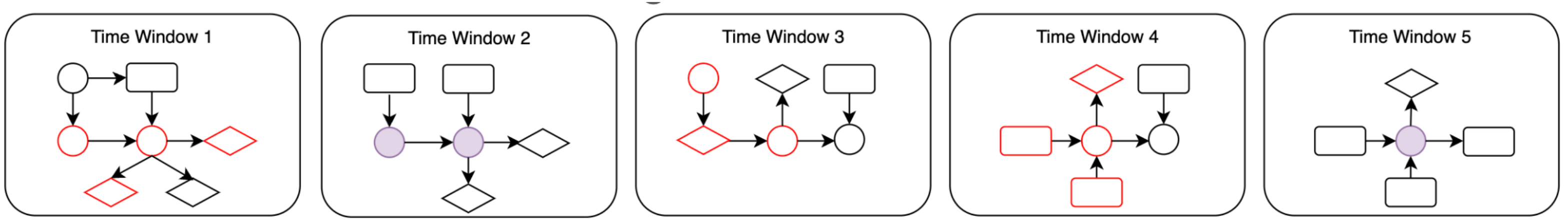
- Malicious activity from a node in a given time window can **yield false positives** in subsequent time windows
- This is a main **consequence** of using **temporal-based models**
- Malicious time windows are correlated through their suspicious nodes with a **malicious queue**
- **A queue captures the activity of suspicious nodes over time and between each other.**



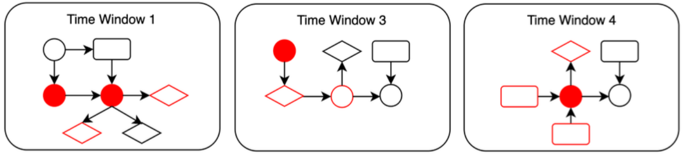
Red Nodes are True Positive Nodes
Purple Nodes are False Positive Nodes

4. Alert Correlation

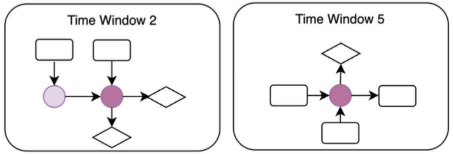
Time windows with common suspicious nodes are fused into a same queue for detection



Queue 1



Queue 2



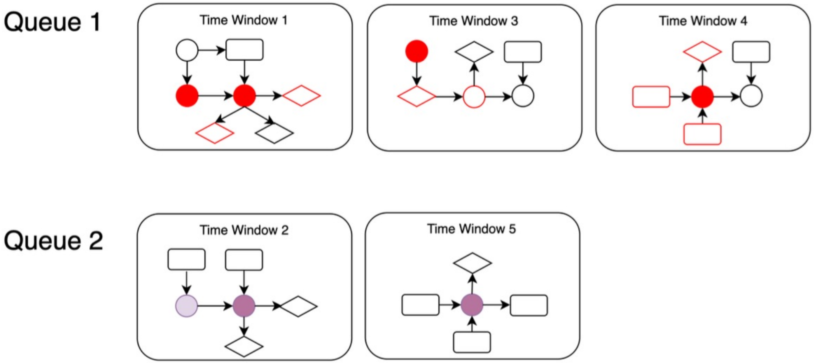
A new window T_{new} is added into a queue Q if

$$\exists\, T \in Q : AN_{Tnew} \cap AN_T \neq \emptyset$$

- AN_{Tnew}: The abnormal nodes in Tnew
- AN_T: The abnormal nodes in T

4. Alert Correlation

A queue anomaly score is calculated with the cumulative product of each time windows's anomaly score



$$AnomalyScore(Q) = \prod_{i=1}^{n} (1 + AnomalyScore(Ti))$$

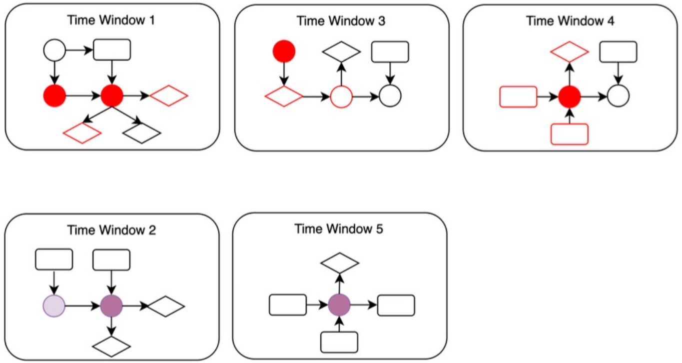
$$AnomalyScore(T) = \frac{1}{n} \sum_{Abnormal\ Edge\ e} Loss(e)$$

4. Alert Correlation

A queue is detected as anomalous if its anomaly score is above the calculated threshold

Queue 1

Queue 2



$$AnomalyScore(Q1) > Threshold$$

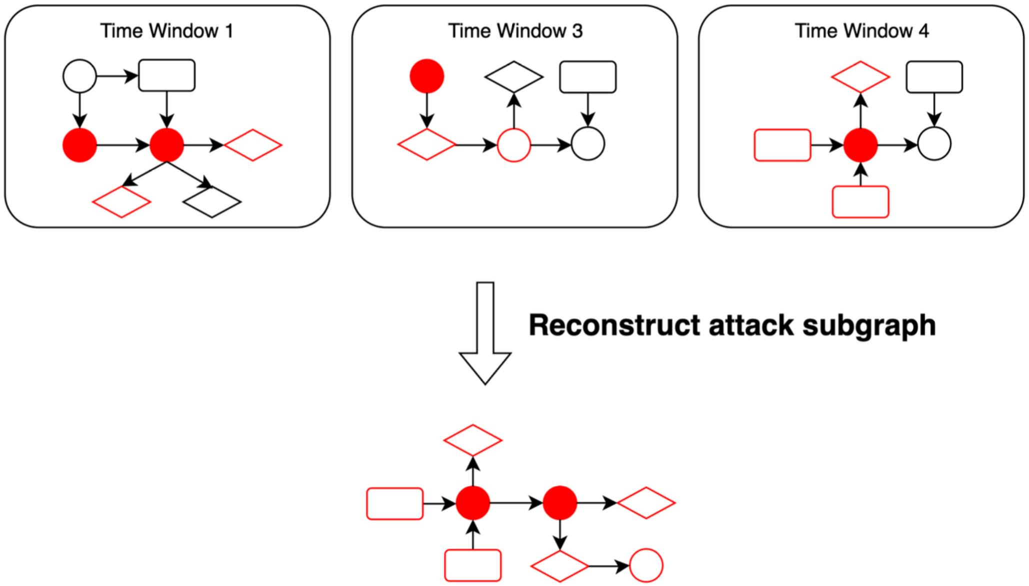
$$AnomalyScore(Q2) < Threshold$$

4. Attack Summary Graph

A summary graph of the attack can be generated from the predicted time windows and nodes

Time-window-level detection

Node-level detection



Evaluation

- Kairos has been evaluated on **8 large imbalanced datasets**
- Most benchmark datasets were published by DARPA's Transparent Computing (TC) programs
- TC organized several adversarial engagements that simulated real-world APTs on enterprise networks.
- **Simulation Duration:** two weeks
- **Benign activities:** browse website, check emails, SSH connection, etc.
- **Attack activities:** browser vulnerability exploitation, malicious process execution, sensitive data leakage.
- Benign data were used for training and validation
- Attack data and benign data were used for testing

Dataset	# of Nodes	# of Edges (in millions)	# of Attack Edges	% of Attack Edges
Manzoor et al.	999,999	89.8	2,842,345	3.165%
DARPA-E3-THEIA	690,105	32.4	3,119	0.010%
DARPA-E3-CADETS	178,965	10.1	1,248	0.012%
DARPA-E3-ClearScope	68,549	9.7	647	0.006%
DARPA-E5-THEIA	739,329	55.4	86,111	0.156%
DARPA-E5-CADETS	90,397	26.5	793	0.003%
DARPA-E5-ClearScope	91,475	40.0	4,044	0.010%
DARPA-OpTC	9,485,265	75.0	33,504	0.045%

Evaluation

Kairos achieves very high precision despite the imbalanced nature of datasets

Datasets	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**Accuracy**	**AUC**
Manzoor et al.	100	375	0	0	1.000	1.000	1.000	1.000
E3-THEIA	**10**	**216**	**1**	**0**	**0.909**	**1.000**	**0.996**	**0.998**
E3-CADETS	4	174	1	0	0.800	1.000	0.994	0.997
E3-ClearScope	5	112	2	0	0.714	1.000	0.983	0.991
E5-THEIA	2	173	1	0	0.667	1.000	0.994	0.997
E5-CADETS	**16**	**238**	**0**	**0**	**1.000**	**1.000**	**1.000**	**1.000**
E5-ClearScope	10	217	5	0	0.667	1.000	0.978	0.989
OpTC	**32**	**1210**	**6**	**0**	**0.842**	**1.000**	**0.995**	**0.998**

Evaluation

Kairos consistently outperforms 2 other SOTA models: Unicorn [2] and ThreaTrace [3]

Table 7. Comparison study between Unicorn and Kairos.

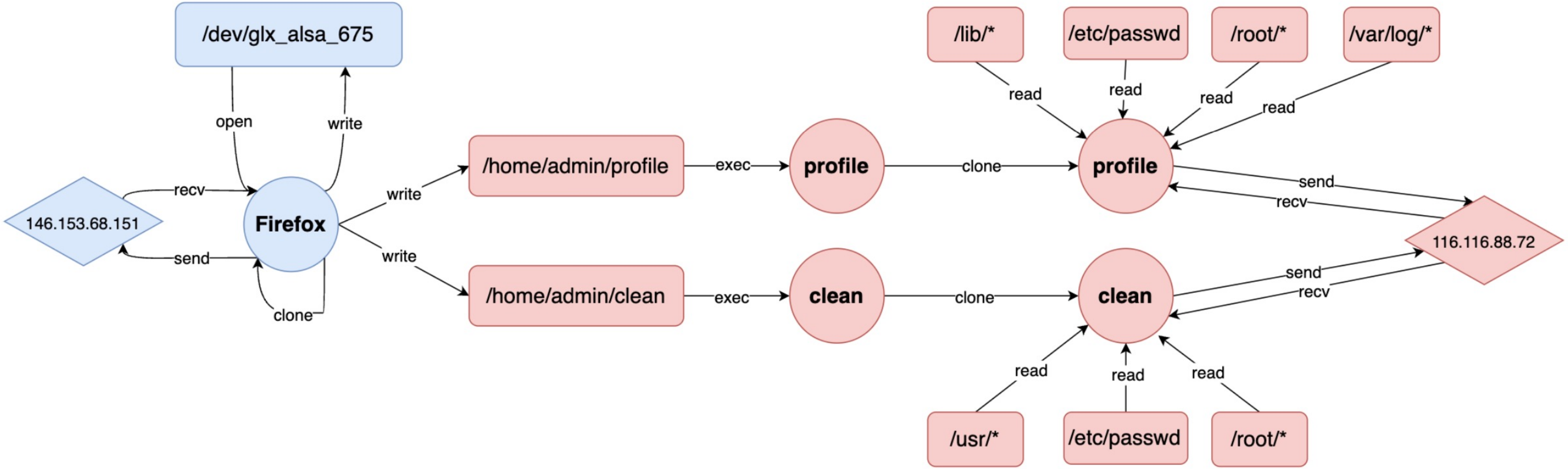
Datasets	System	Precision	Recall	Accuracy
Manzoor et al.	Unicorn	0.98	0.93	0.96
	Kairos	**1.00**	**1.00**	**1.00**
E3-CADETS	Unicorn	0.98	1.00	0.99
	Kairos	**1.00**	1.00	**1.00**
E3-THEIA	Unicorn	1.00	1.00	1.00
	Kairos	1.00	1.00	1.00
E3-ClearScope	Unicorn	0.98	1.00	0.98
	Kairos	**1.00**	1.00	**1.00**

Table 8. Comparison study between ThreaTrace and Kairos.

Datasets	System	Precision	Recall	Accuracy
Manzoor et al.	ThreaTrace	0.98	0.99	0.99
	Kairos	**1.00**	**1.00**	**1.00**
E3-CADETS	ThreaTrace	0.90	0.99	0.99
	Kairos	**1.00**	0.95	0.99
E3-THEIA	ThreaTrace	0.87	0.99	0.99
	Kairos	**1.00**	0.95	0.99
E5-CADETS	ThreaTrace	0.63	0.86	0.97
	Kairos	**1.00**	0.85	**0.98**
E5-THEIA	ThreaTrace	0.70	0.92	0.99
	Kairos	**1.00**	0.92	0.99

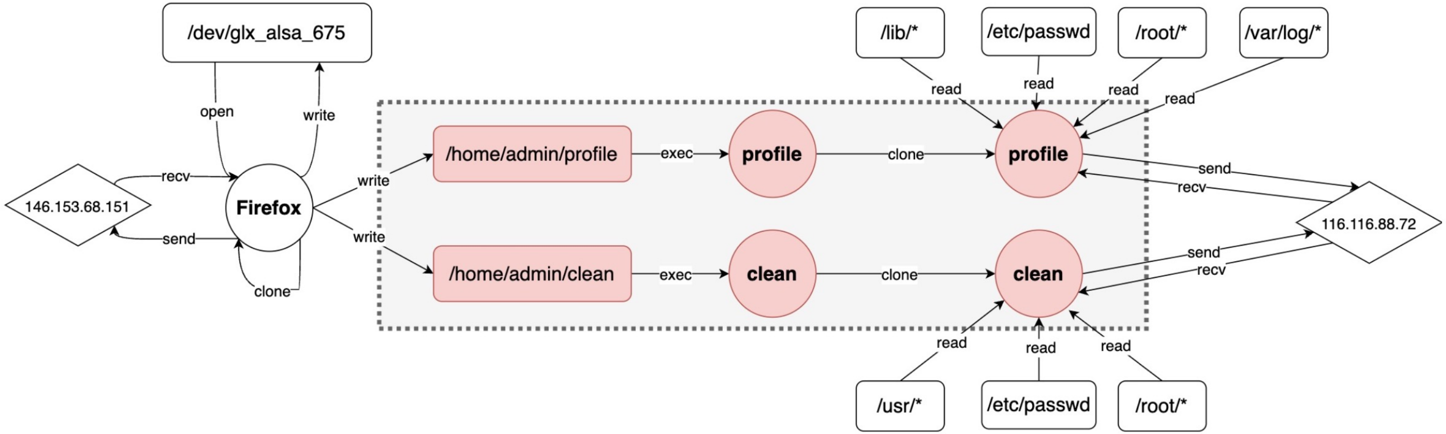
Explainability

Detection Report of THEIA E3. Red nodes are malicious nodes detected by Kairos; Blue nodes are malicious nodes missed by Kairos.



Explainability

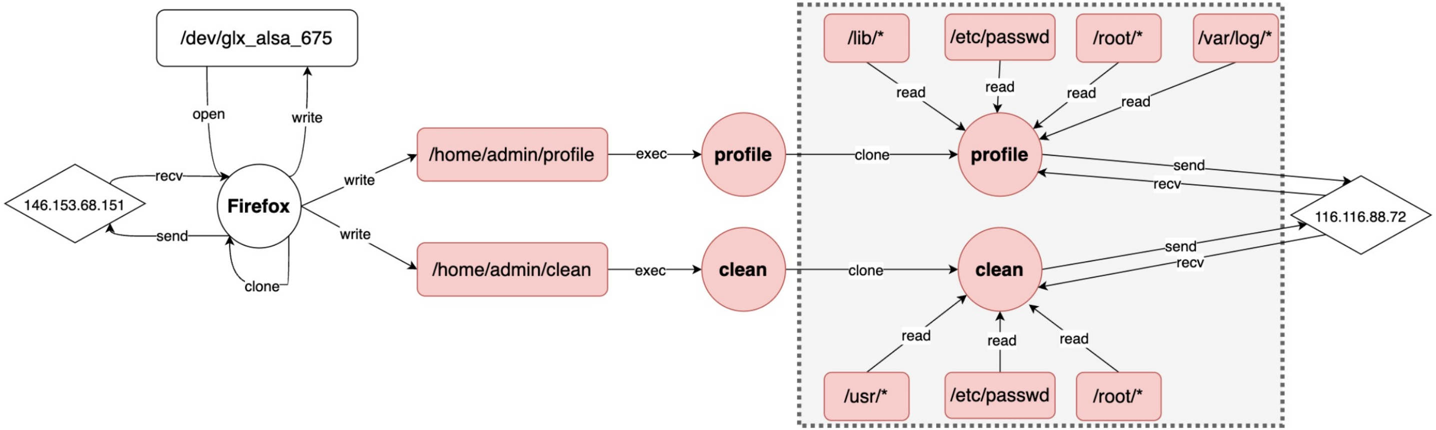
Kairos is able to detect various steps of the attack chain.



Malicious Process Execution

Explainability

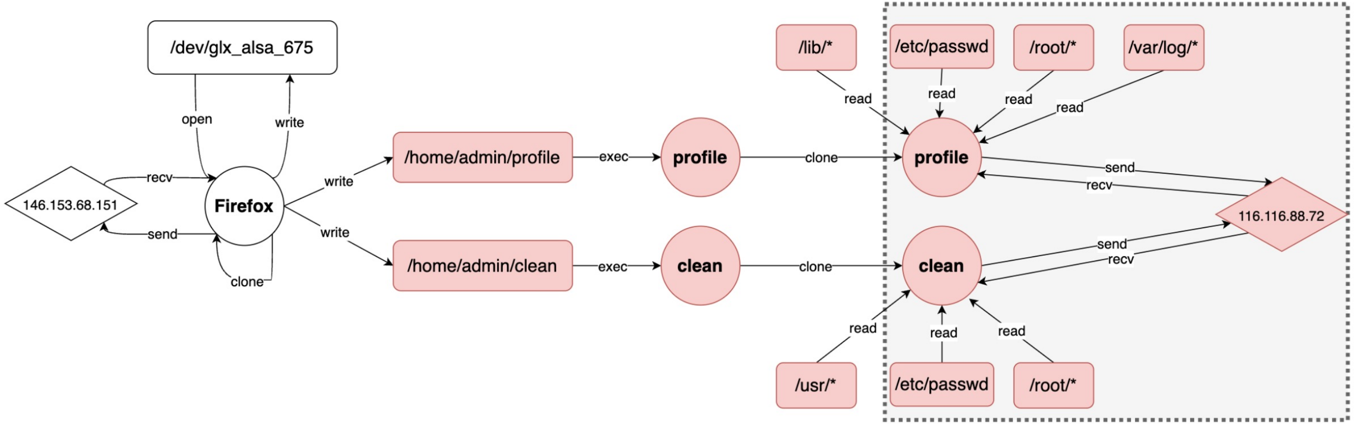
Kairos is able to detect various steps of the attack chain.



Sensitive information read

Explainability

Kairos is able to detect various steps of the attack chain.



Sensitive information leakage

Limitations

- Kairos **always misses the attacks in the initial stage**, because malicious behaviors might not have much difference from benign behaviors. So Kairos cannot identify the malicious subgraphs in initial stage.
- Using a time window-based detection, it is required **to wait upon the end of a time window** to perform detection

Ongoing Work

- We improved Kairos for a more fine-grained detection at the **node-level instead of the queue level**
- We also **improved the detection performance** of Kairos by updating the model architecture and features
- We are currently working on a **near real-time model** able to infer detection after each new edge appearing in the graph

Future Work

- **Reliability:** Study the robustness of such models to adversarial attacks
- **Production usage:** Apply such models in real-life scenarios
- **Inductive bias:** Train a model on one dataset and do inference on another dataset

The End

Do you have any questions?

Contact: Tristan BILOT, tristan.bilot@universite-paris-saclay.fr

References

[1] Manzoor, Emaad, Sadegh M. Milajerdi, and Leman Akoglu. "Fast memory-efficient anomaly detection in streaming heterogeneous graphs." *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016.
[2] Han, Xueyuan, et al. "Unicorn: Runtime provenance-based detector for advanced persistent threats." *arXiv preprint arXiv:2001.01525* (2020).
[3] Wang, Su, et al. "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning." *IEEE Transactions on Information Forensics and Security* 17 (2022): 3972-3987.
[4] Li, Shaofei, et al. "NODLINK: An Online System for Fine-Grained APT Attack Detection and Investigation." *arXiv preprint arXiv:2311.02331* (2023).
[5] Cheng, Zijun, et al. "Kairos:: Practical Intrusion Detection and Investigation using Whole-system Provenance." *arXiv preprint arXiv:2308.05034* (2023).
[6] Zhang, Zhaoqi, Panpan Qi, and Wei Wang. "Dynamic malware analysis with feature engineering and feature learning." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. No. 01. 2020.
[7] Rossi, Emanuele, et al. "Temporal graph networks for deep learning on dynamic graphs." *arXiv preprint arXiv:2006.10637* (2020).