Towards provenance-based intrusion detection

Thomas Pasquier, University of Bristol

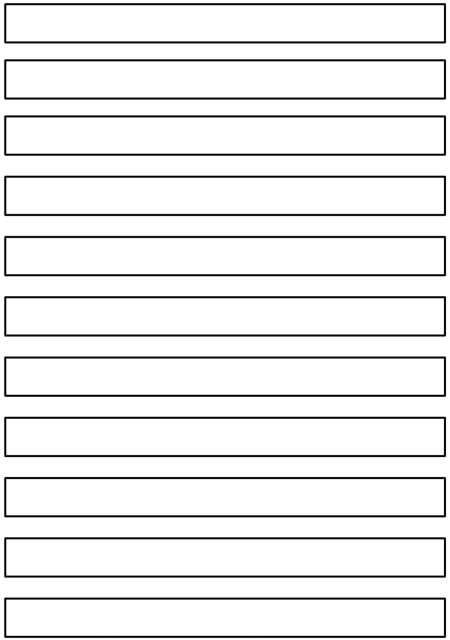
HP Labs, 06/06/2019

Talk loosely based on following publications

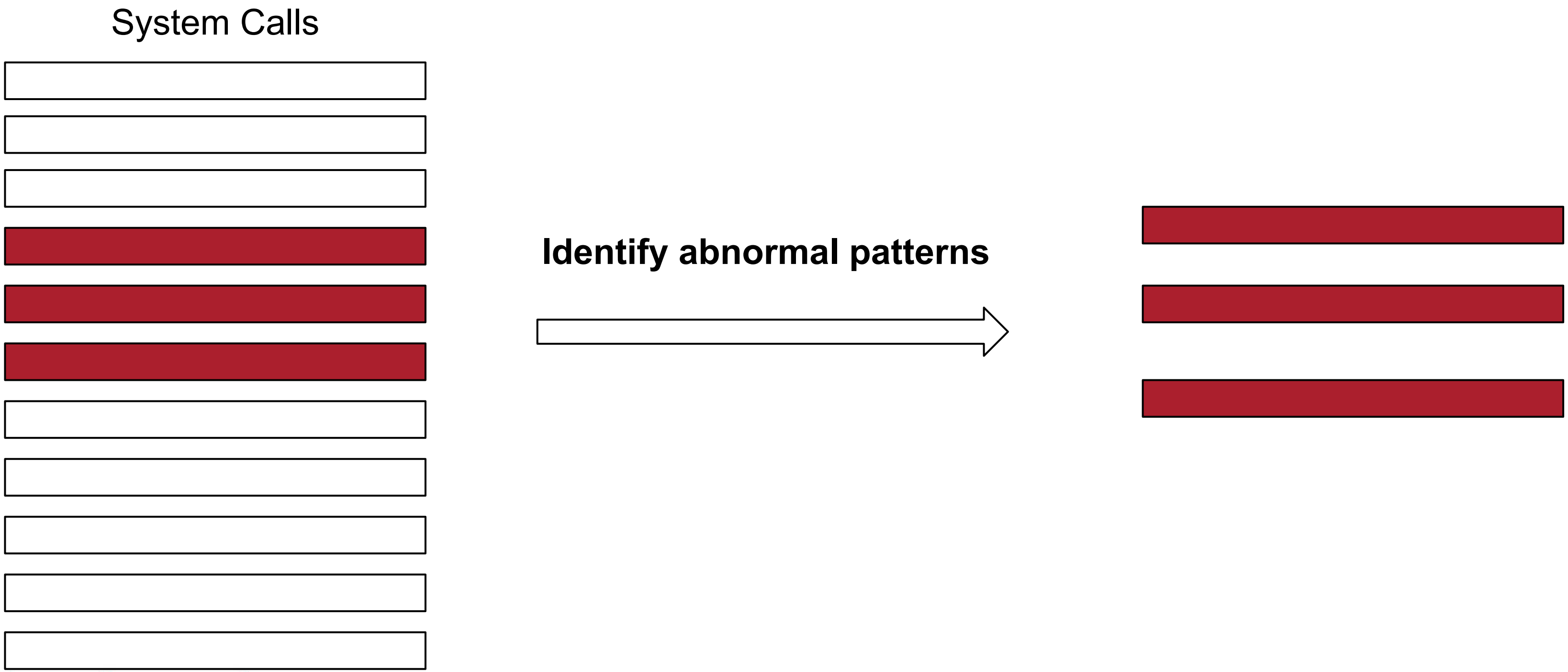
- Han et al. **“UNICORN: Revisiting Host-Based Intrusion Detection in the Age of Data Provenance”**, NDSS 2020
- Pasquier et al. **“Runtime Analysis of Whole-System Provenance”**, ACM CCS 2018
- Han et al. **“Provenance-based Intrusion Detection: Opportunities and Challenges”**, USENIX TaPP 2018
- Han et al. **“FRAPpuccino: Fault-detection through Runtime Analysis of Provenance”**, USENIX HotCloud 2017
- Pasquier et al. **“Practical Whole-System Provenance Capture”**, ACM SoCC 2017

System call based intrusion detection

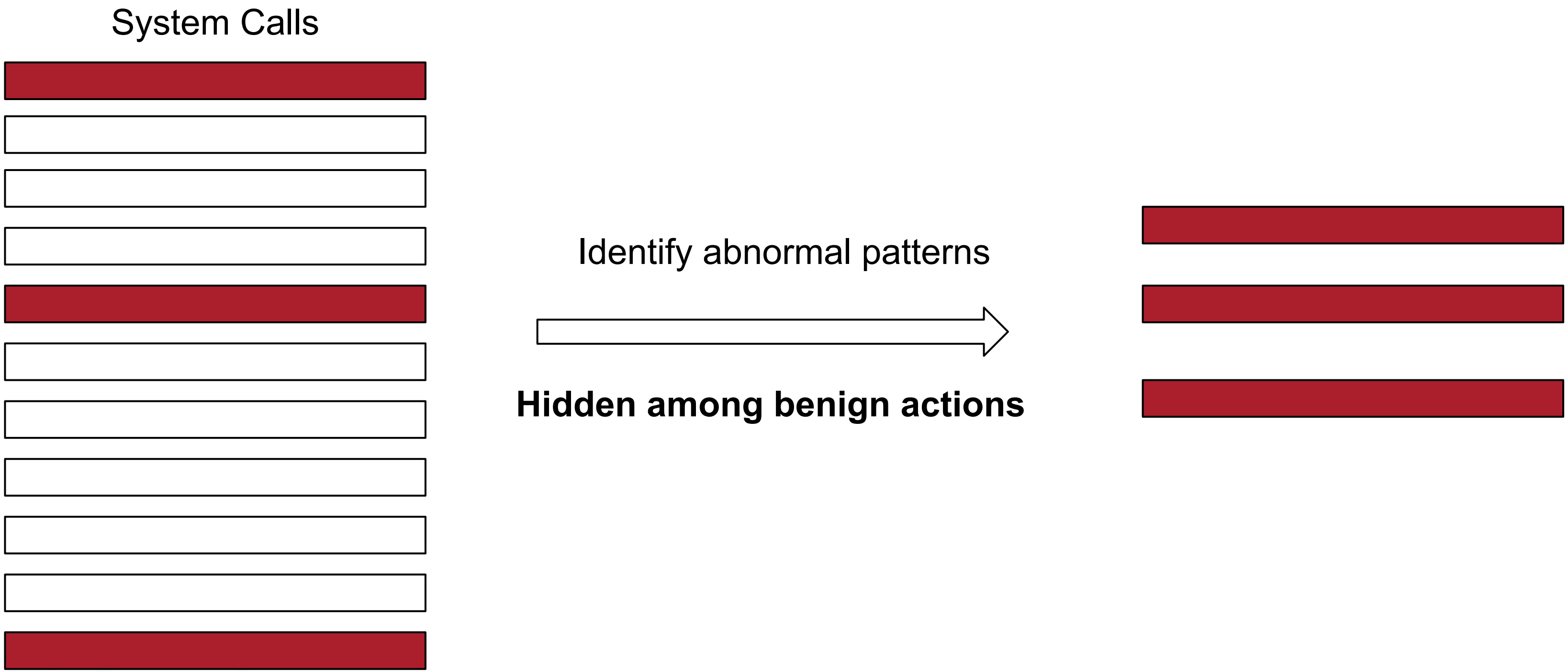
System Calls



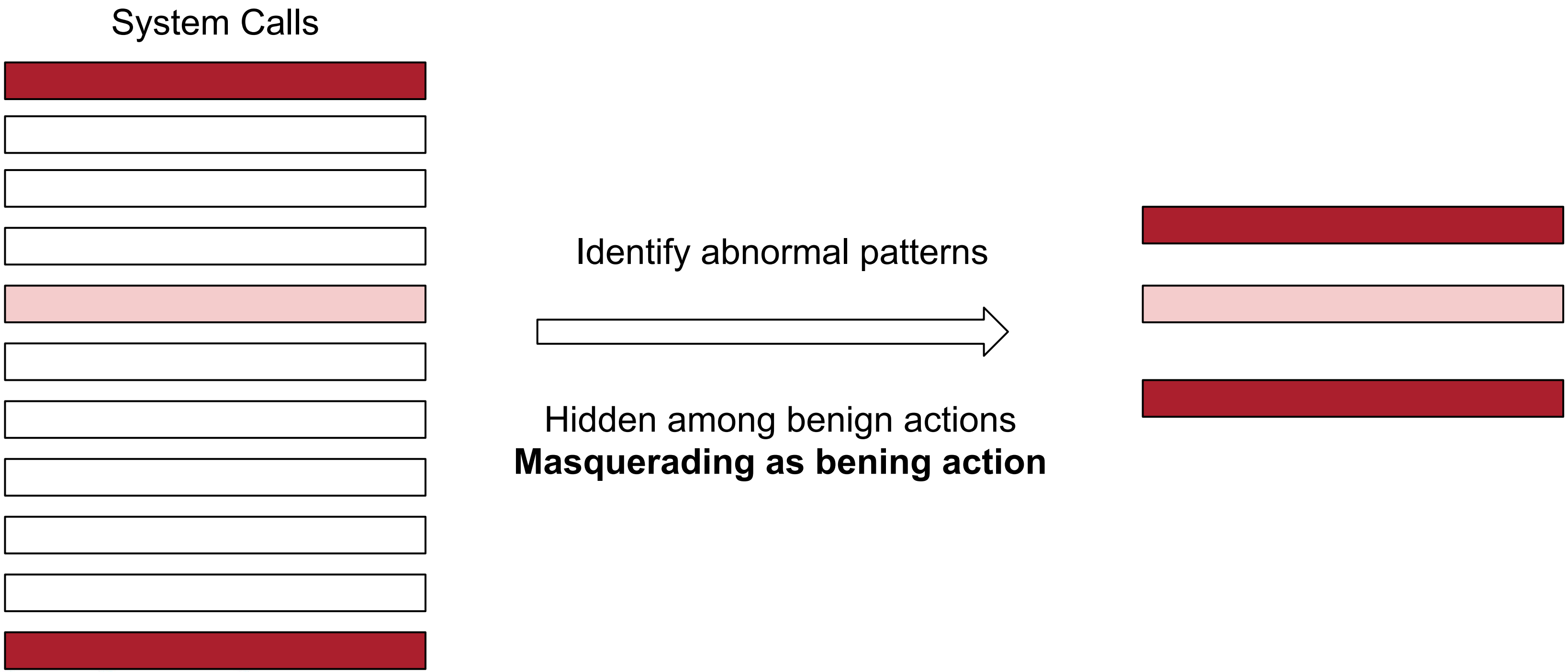
System call based intrusion detection



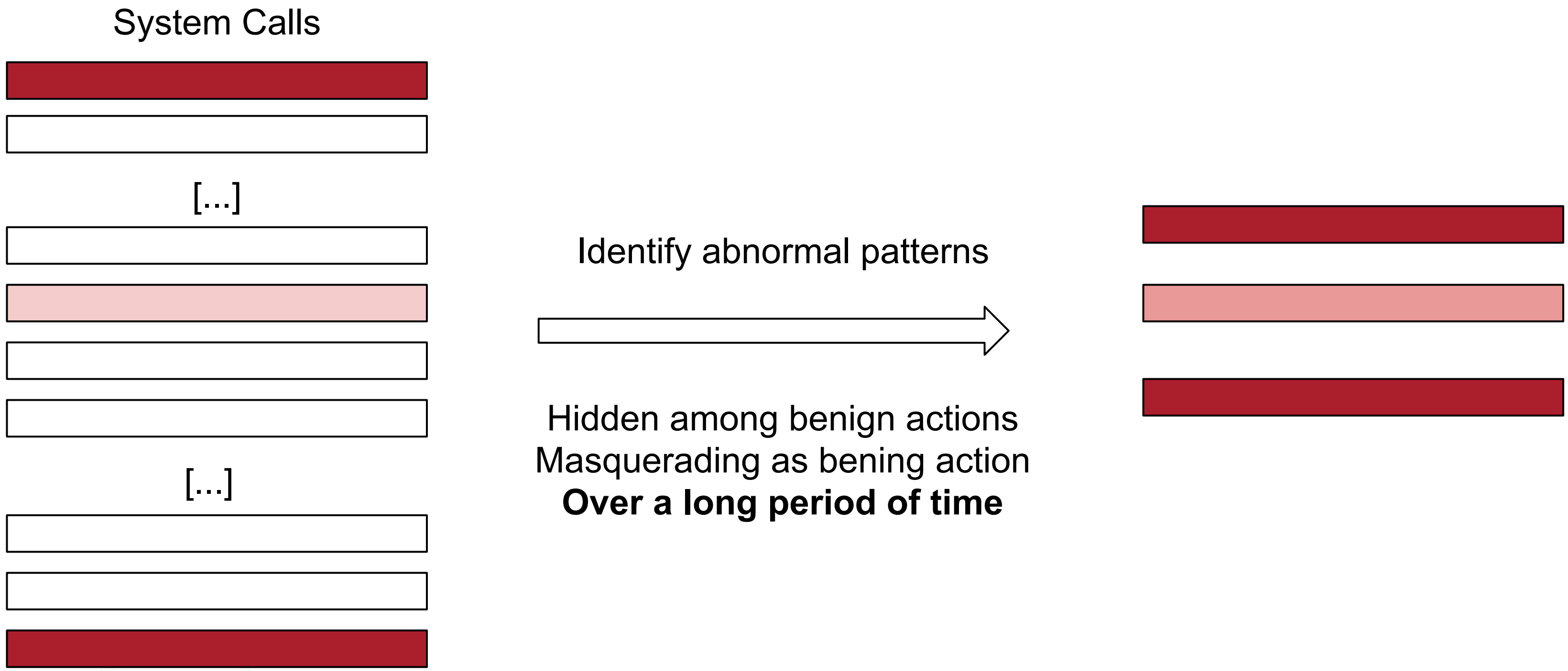
System call based intrusion detection



System call based intrusion detection

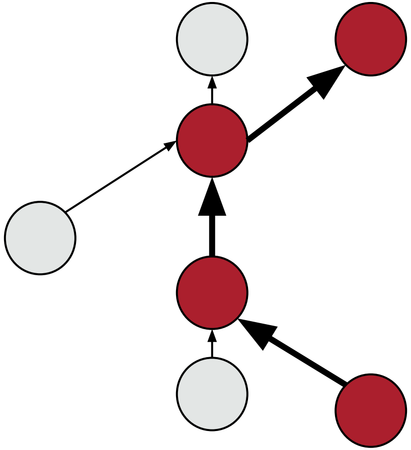


System call based intrusion detection



Provenance-based intrusion detection

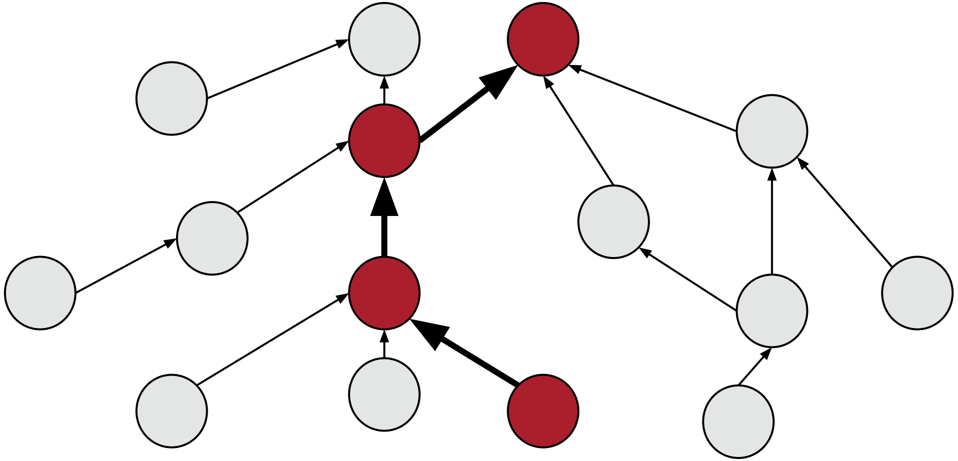
- **Intuition**: provenance graph **exposes causality relationships** between events



bristol.ac.uk

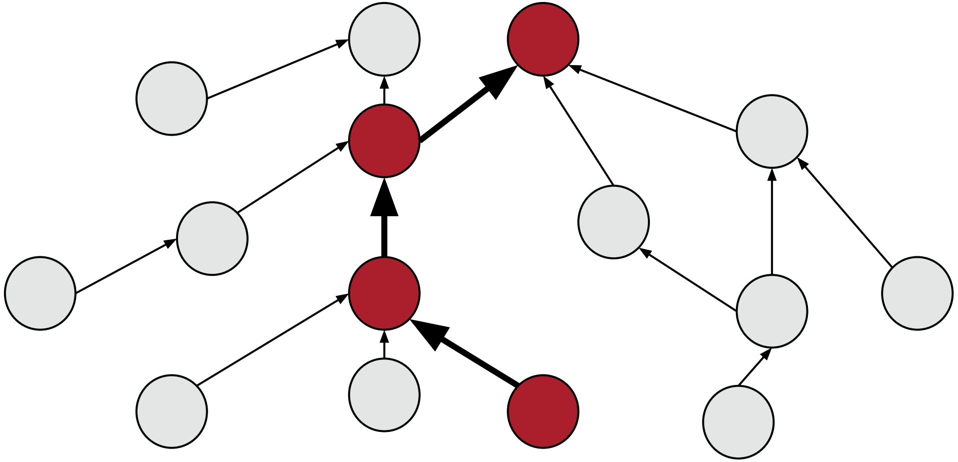
Provenance-based intrusion detection

- **Intuition**: provenance graph **exposes causality relationships** between events



Provenance-based intrusion detection

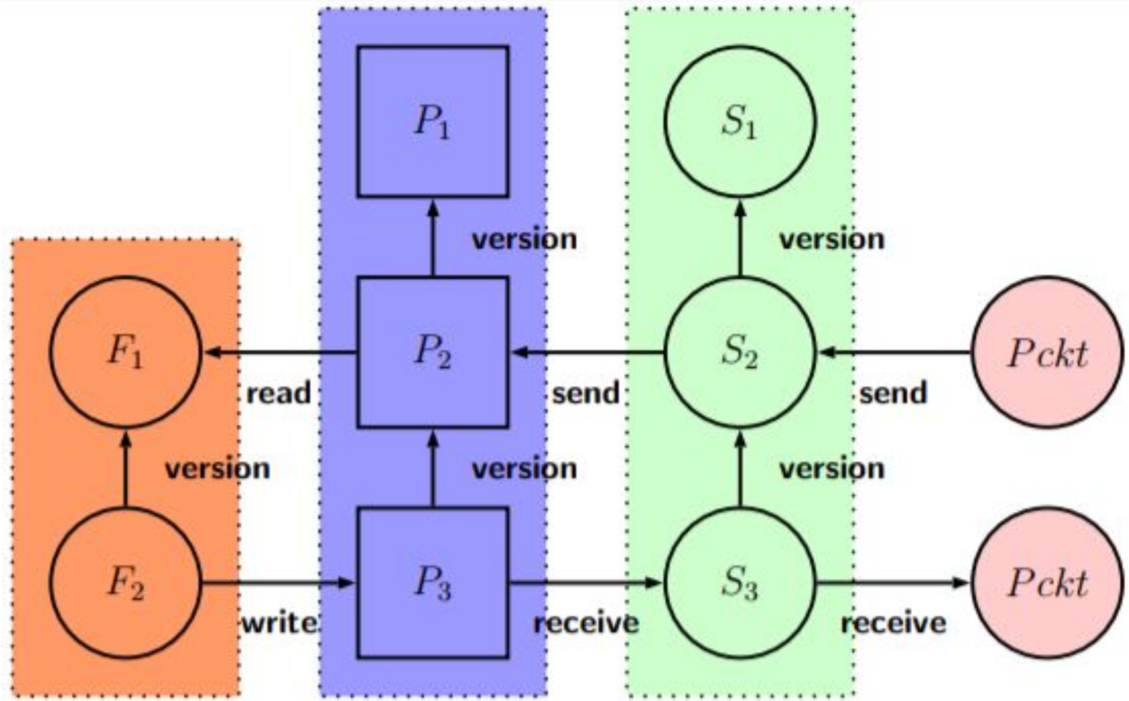
- Related events are connected even across long period of time



What is provenance in an operating system?

- Represent interactions between system objects
- Represented as a **directed acyclic graph**
- Information Flows
- **Relationship** between **kernel object states**
- History of a system execution

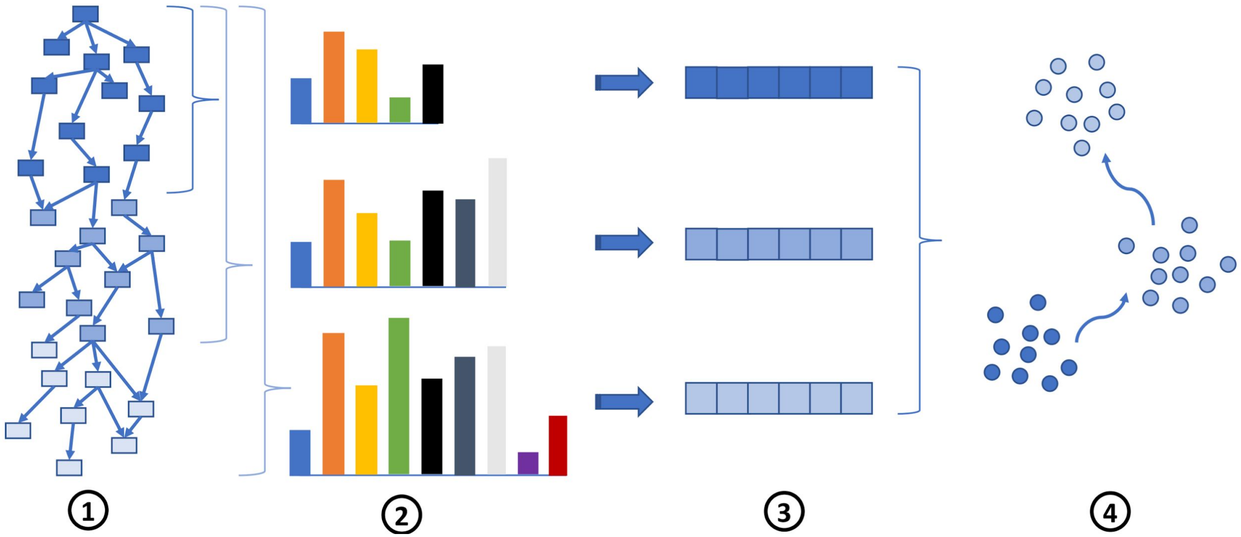
Example provenance



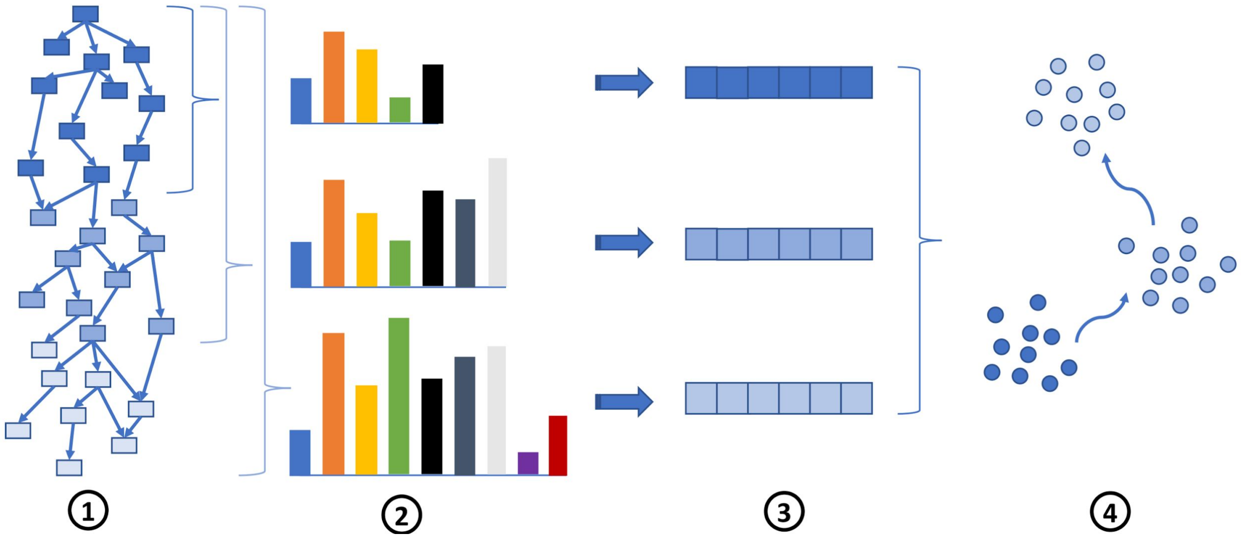
Provenance-based Intrusion Detection

- We target environment with minimal human intervention
  - Relatively well defined behaviour
- Build a model of system behaviour (unsupervised, batch training)
  - in a controlled environment
  - from a representative workload
- Detect deviation from the model
- Several approaches being explored…

Detecting intrusion

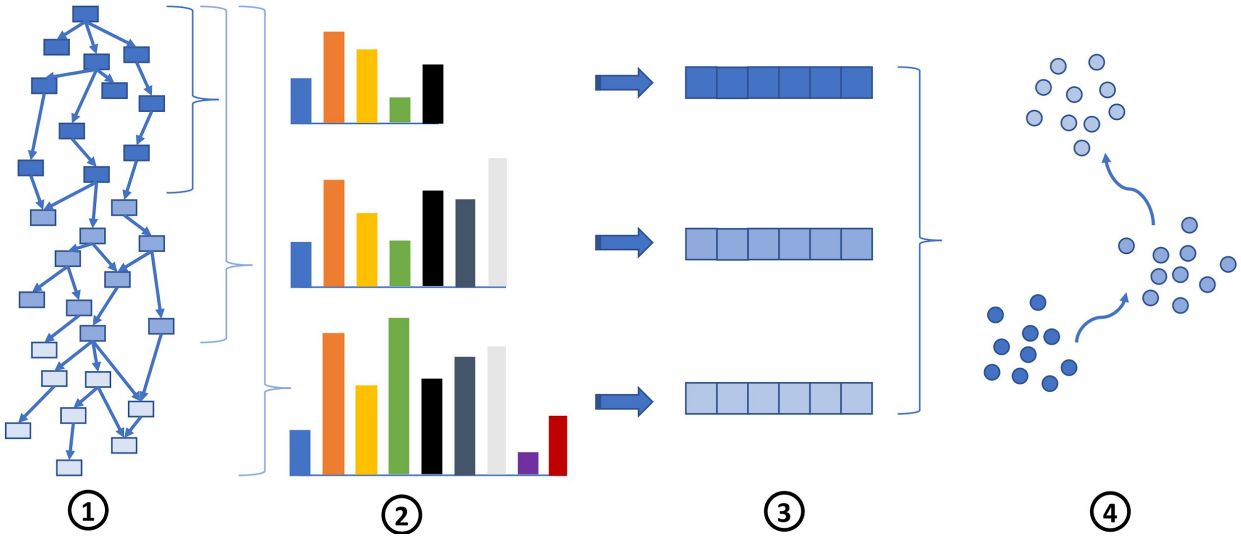


Detecting intrusion



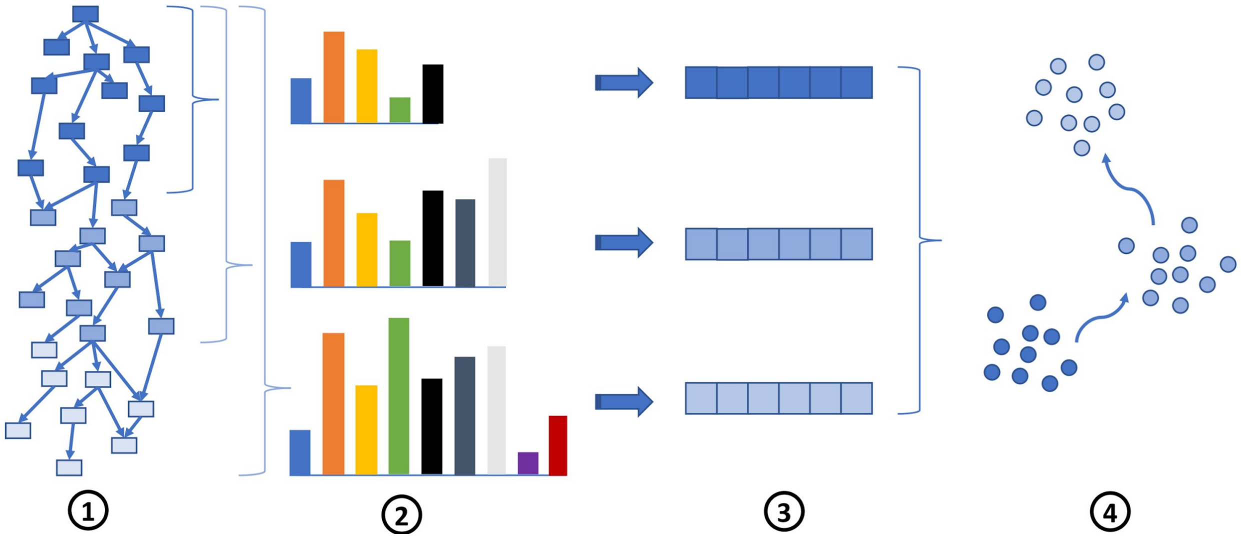
1) Graph streamed in, converted to histogram, labelled using struct2vec

Detecting intrusion



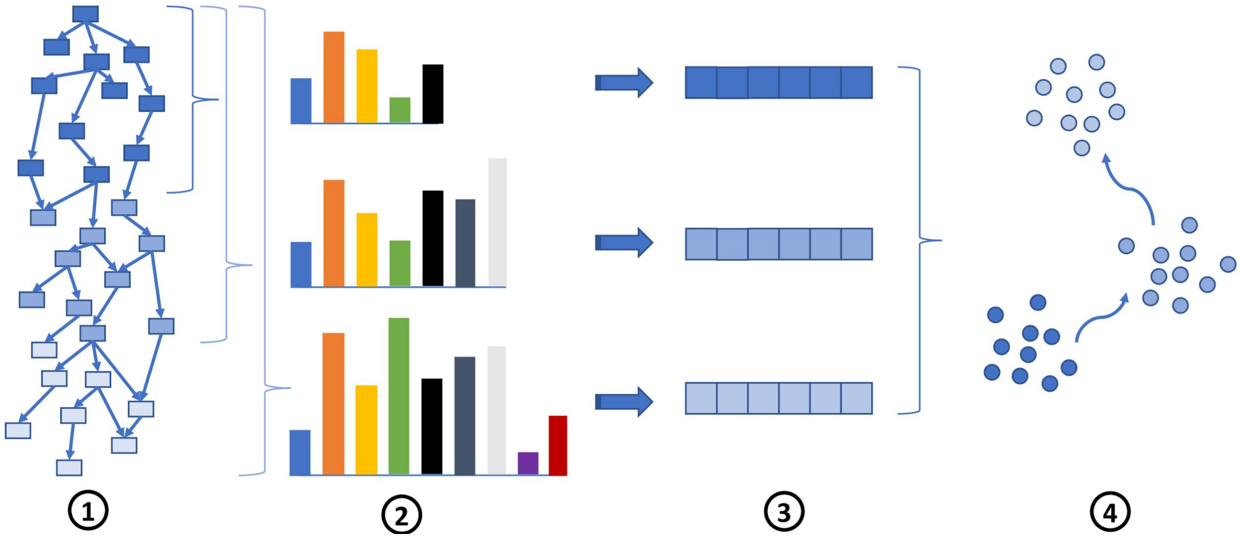
2) At regular interval, histogram converted to a fixed size vector using similarity preserving hashing

Detecting intrusion



3) Feature vectors are clustered

Detecting intrusion



4) Cluster forms "meta-state", transitions are modelled

In deployment, anomaly detected via clustering and "meta-state" model

How well does it work?

Experiment	Dataset	# of Graphs	Avg. |V|	Avg. |E|	Raw Data Size (GiB)
StreamSpot	YouTube	100	8,292	113,229	0.3
	Gmail	100	6,827	37,382	0.1
	Download	100	8,831	310,814	1
	VGame	100	8,637	112,958	0.4
	CNN	100	8,990	294,903	0.9
	Attack	100	8,891	28,423	0.1
DARPA CADETS	Benign	66	59,983	4,811,836	271
	Attack	8	386,548	5,160,963	38
DARPA ClearScope	Benign	43	2,309	4,199,309	441
	Attack	51	11,769	4,273,003	432
DARPA THEIA	Benign	2	19,461	1,913,202	4
	Attack	25	275,822	4,073,621	85
CamFlow CI	Benign	125	265,424	975,226	64
	Attack	25	257,156	957,968	12
CamFlow CI-2	Benign	125	238,338	911,153	59
	Attack	25	243,658	949,887	12

Table 1: Characteristics of graph datasets used in the experiments. The StreamSpot data sizes are significantly smaller, because we do not have access to the original raw data, just the processed data.

Experiment	Precision	Recall	Accuracy	F-Score
StreamSpot-Original	0.74	N/A	0.66	N/A
StreamSpot-Unicorn ($R = 1$)	0.51	1.0	0.52	0.67
StreamSpot-Unicorn ($R = 3$)	0.98	0.93	0.96	0.94
DARPA CADETS	0.98	1.0	0.99	0.99
DARPA ClearScope	0.98	1.0	0.98	0.99
DARPA THEIA	1.0	1.0	1.0	1.0
CamFlow CI	0.85	0.96	0.90	0.90
CamFlow CI-2	0.75	0.80	0.77	0.78

Table 2: Experimental results. We estimate StreamSpot's average accuracy and precision from the figure included in the paper [97], which does not report exact values. They did not report recall or F-score.

How well does it work?

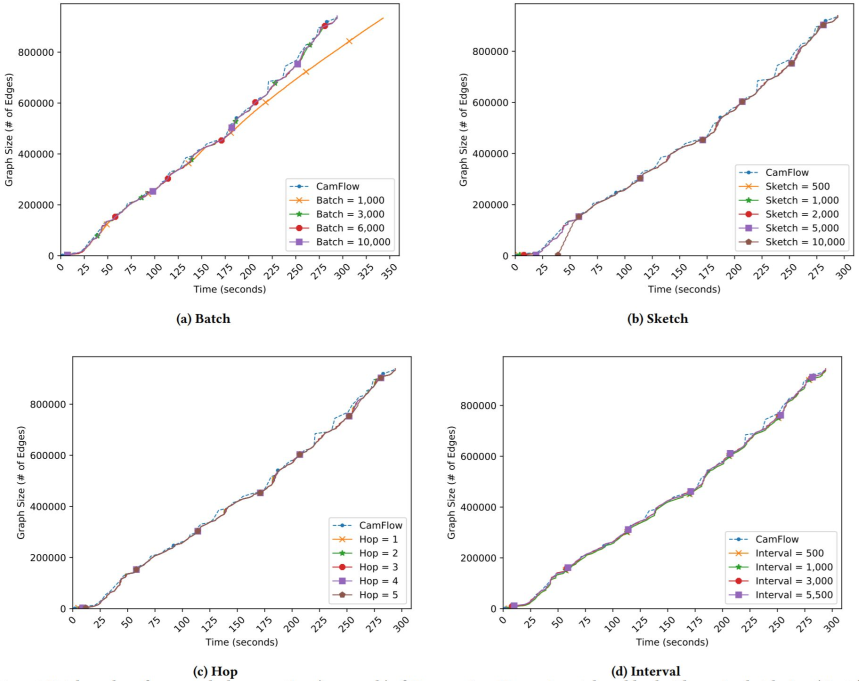


Figure 3: Total number of processed edges over time (in seconds) of UNICORN in a CI experimental workload with varying batch sizes (Fig. 3a), sketch sizes (Fig. 3b), hop counts (Fig. 3c), and intervals of sketch generation (Fig. 3d). Dashed blue line represents the speed of graph edges streamed into UNICORN for analysis. Red baseline has the same configurations as those used in our experiment and indicates the values of the controlled parameters (that remain constant) in each figure.

How well does it work?

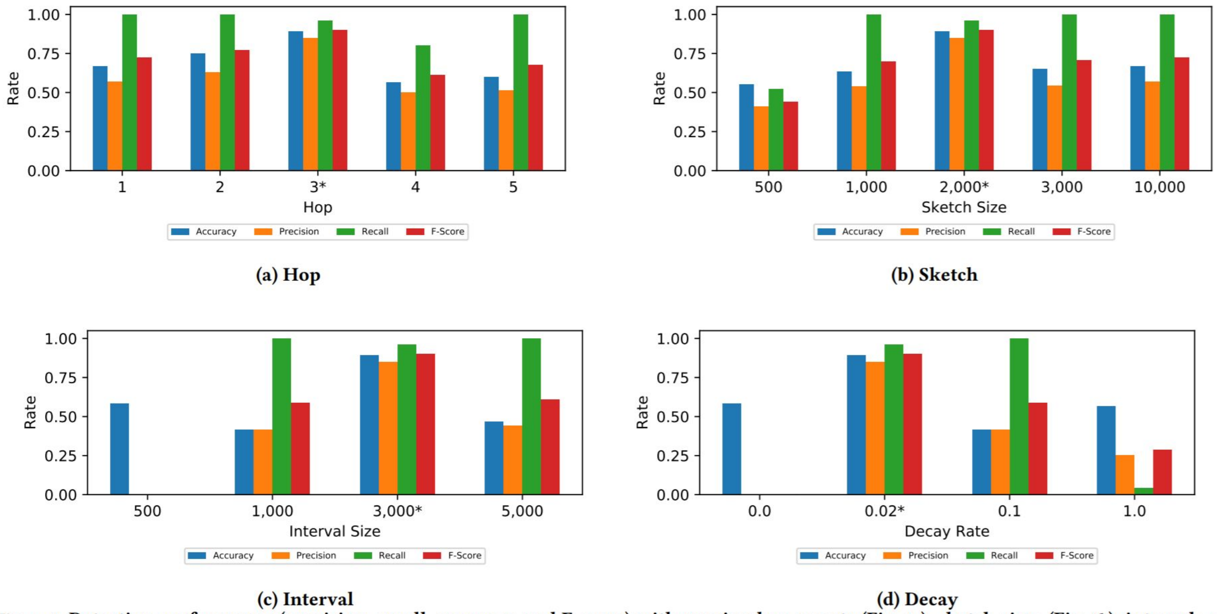


Figure 4: Detection performance (precision, recall, accuracy, and F-score) with varying hop counts (Fig. 4a), sketch sizes (Fig. 4b), intervals of sketch generation (Fig. 4c), and decay factor (Fig. 4d). Baseline values (*) are used by the controlled parameters (that remain constant) in each figure.

How well does it work?

Configuration Parameter	Parameter Value	Max Memory Usage (MB)
Hop Count	R = 1	562
	R = 2	624
	R = 3	687
	R = 4	749
	R = 5	812
Sketch Size	|S| = 500	312
	|S| = 1,000	437
	|S| = 2,000	687
	|S| = 5,000	1,374
	|S| = 10,000	2,498

Table 5: Memory usage with varying hop counts and sketch sizes.

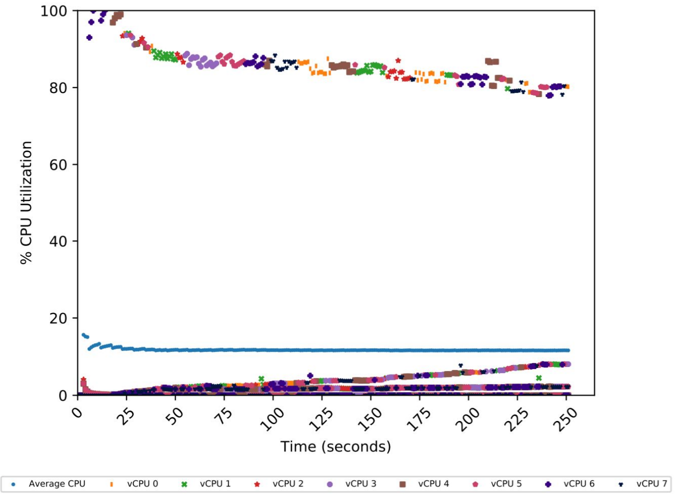


Figure 6: Per virtual CPU and average CPU utilization.

How well does it work?

Over long time period?

How well does it work?

CPU over long time period? 15% CPU time across cores

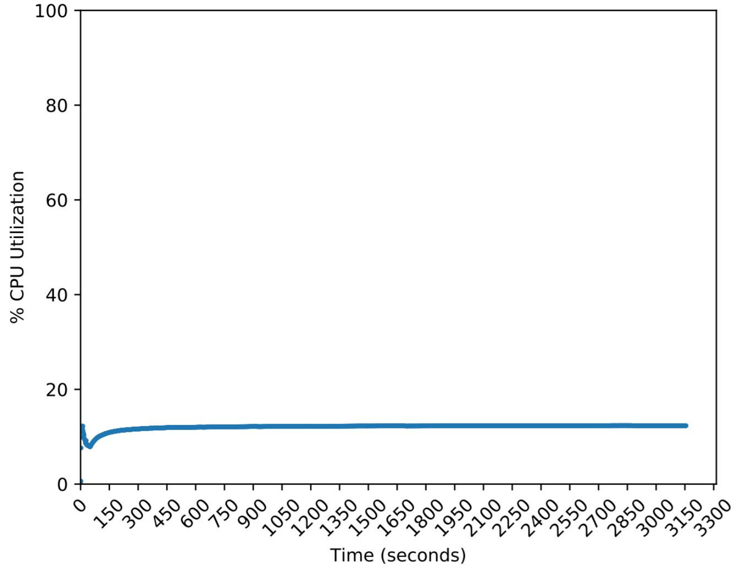


Figure 5: Average CPU utilization with the baseline configurations.

Some insights

- We can detect intrusion out of graph structure with little metadata
  - Vertex type (thread, file, socket etc…)
  - Edge type (read, write, connect etc…)
- Processing speed
  - Current prototype
  - Data generation speed < processing speed!

Some insights

- Doing proper evaluation is hard!
- Dataset are hard to generate
    - What is a good quality dataset?
- Hard to compare across papers, a lot is not available
    - Experiments (i.e. attacks)
    - Capture Mechanisms
    - Analysis pipelines
- Leads to unsatisfactory evaluation
    - I may be able to compare to similar techniques (may reuse dataset)
    - … very hard for unrelated one

Students working on closely related projects

Michael Han: currently on internship at NEC Labs America. Working on provenance-based IDS.

Abia Amin: working on IDS for smart-building settings.

Connor Goodwin: NCSC funded student starting in October working on automated intrusion report.

… more soon.

Collaborators

University of BRISTOL

THE UNIVERSITY OF BRITISH COLUMBIA

UNIVERSITY OF CAMBRIDGE

SRI International

HARVARD

bristol.ac.uk

Research Trajectory

- Capture system open-sourced (http://camflow.org)
  - Maintained for >4 years
  - Used in multiple research projects
  - Datasets soon to be released
  - Exploring means to perform **cross-evaluation** in an evolving “capture field”
- Extending to **distributed systems**
  - Capture system already support this
- Exploring more **advanced ML techniques**
  - Although the relatively simple approach presented bring good results
- Looking at **supply chain** attack/integrity.
- Looking for industry partners

University of BRISTOL

Thank you, questions?

tfjmp.org

camflow.org