University of BRISTOL

Provenance-based Intrusion Detection

Thomas Pasquier, University of Bristol

UK Cyber Security Winter School, Newcastle, 15/01/2020

bristol.ac.uk

EPSRC
Pioneering research and skills

DARPA

NSF

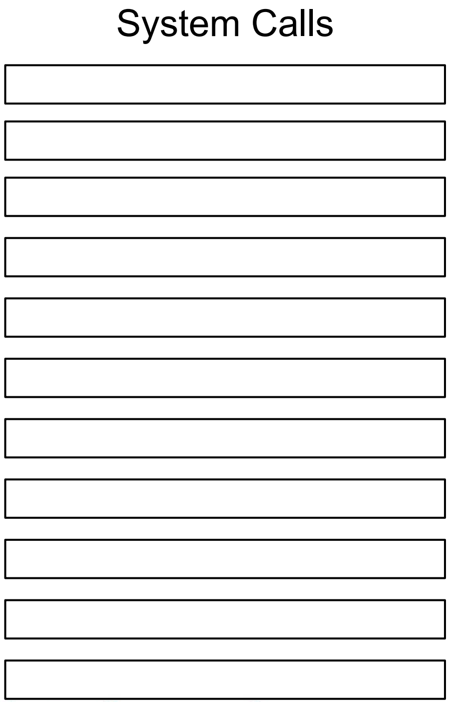
UNIVERSITY OF CAMBRIDGE

HARVARD UNIVERSITY

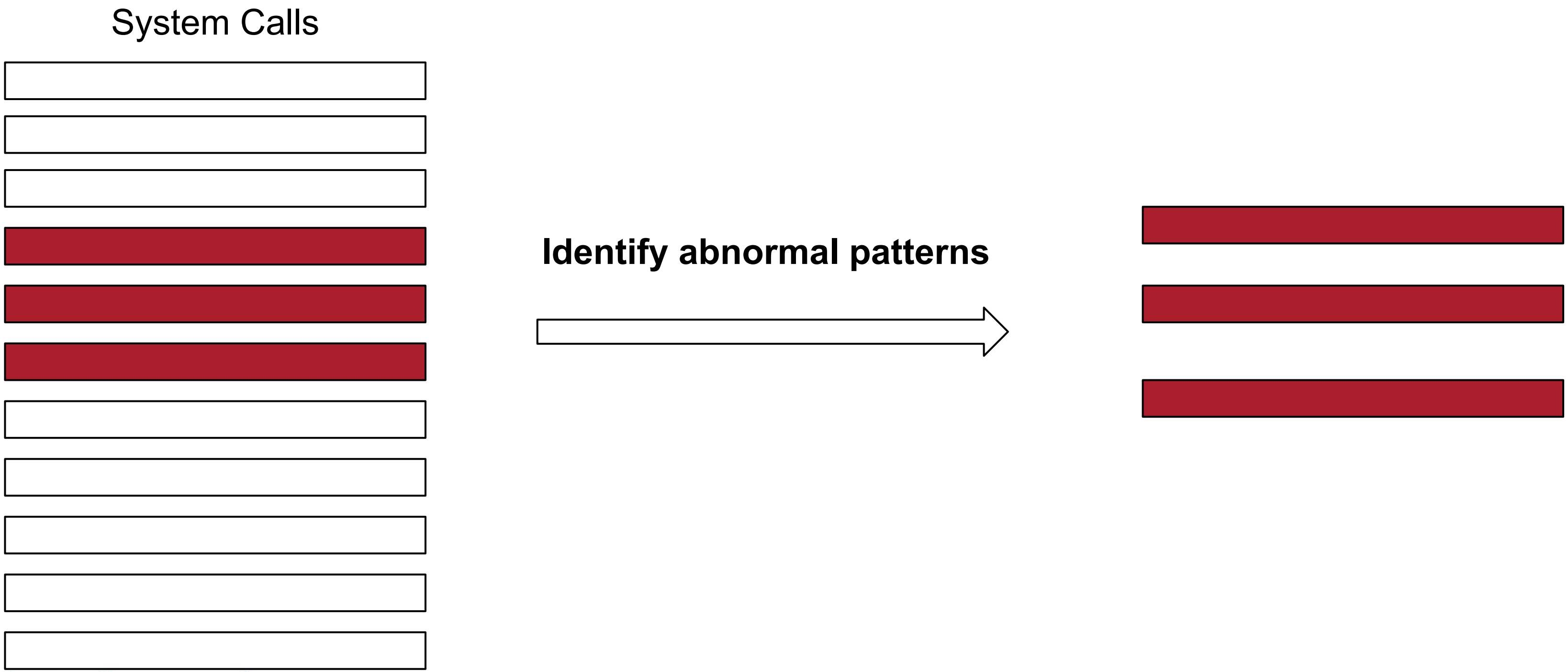
University of BRISTOL

bristol.ac.uk

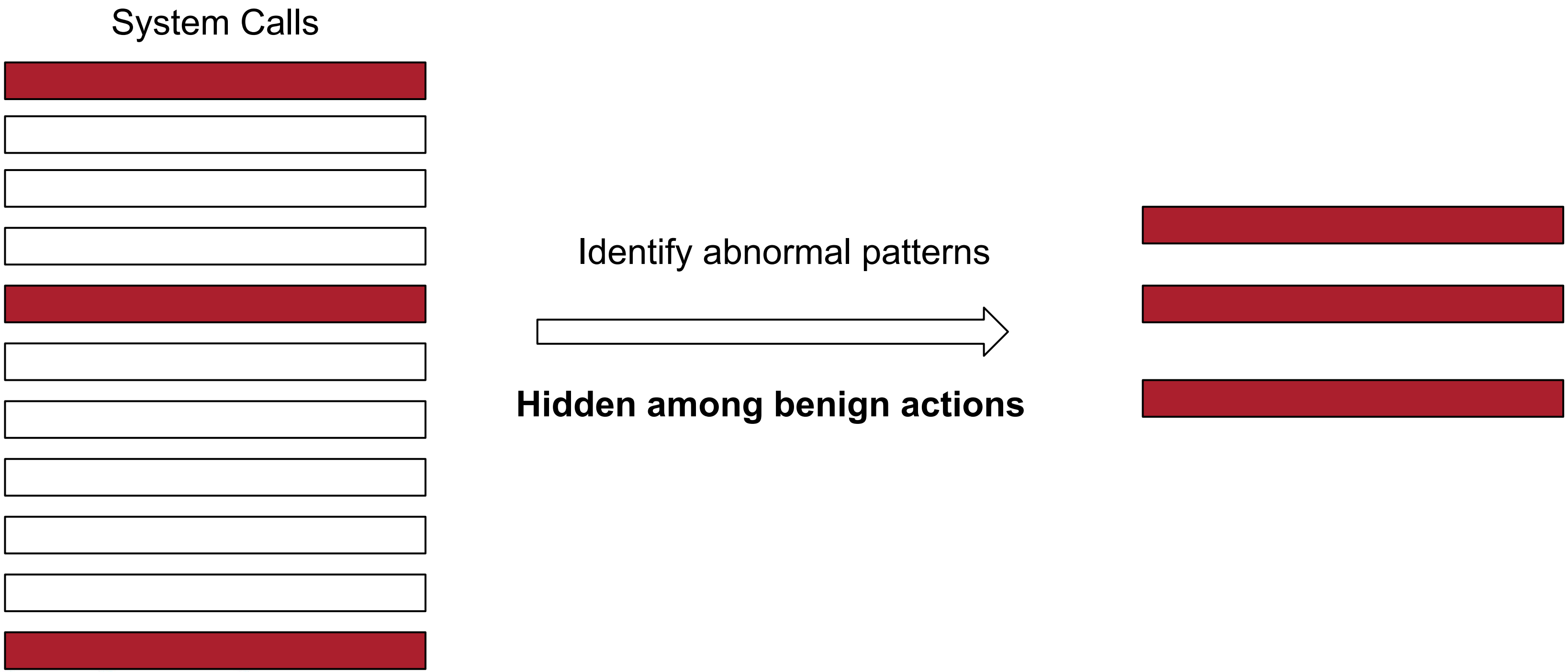
System call based intrusion detection



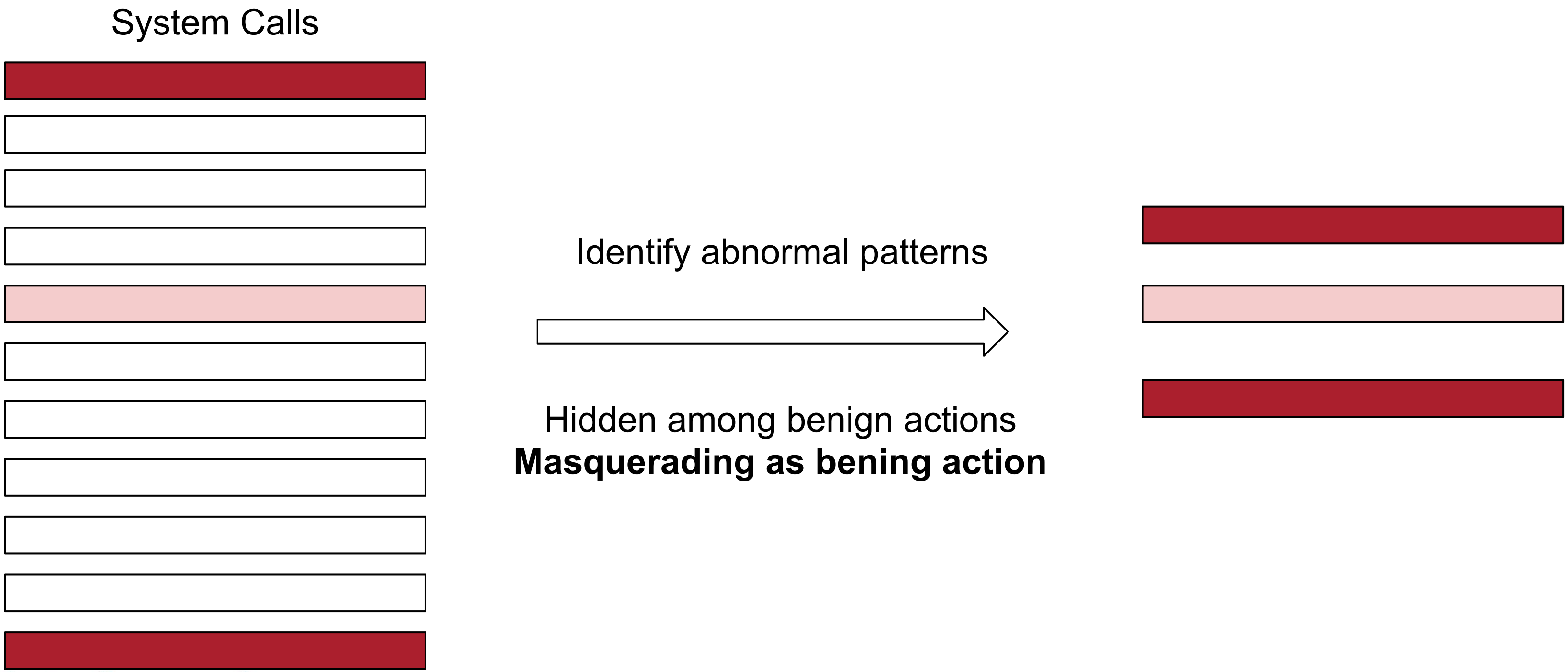
System call based intrusion detection



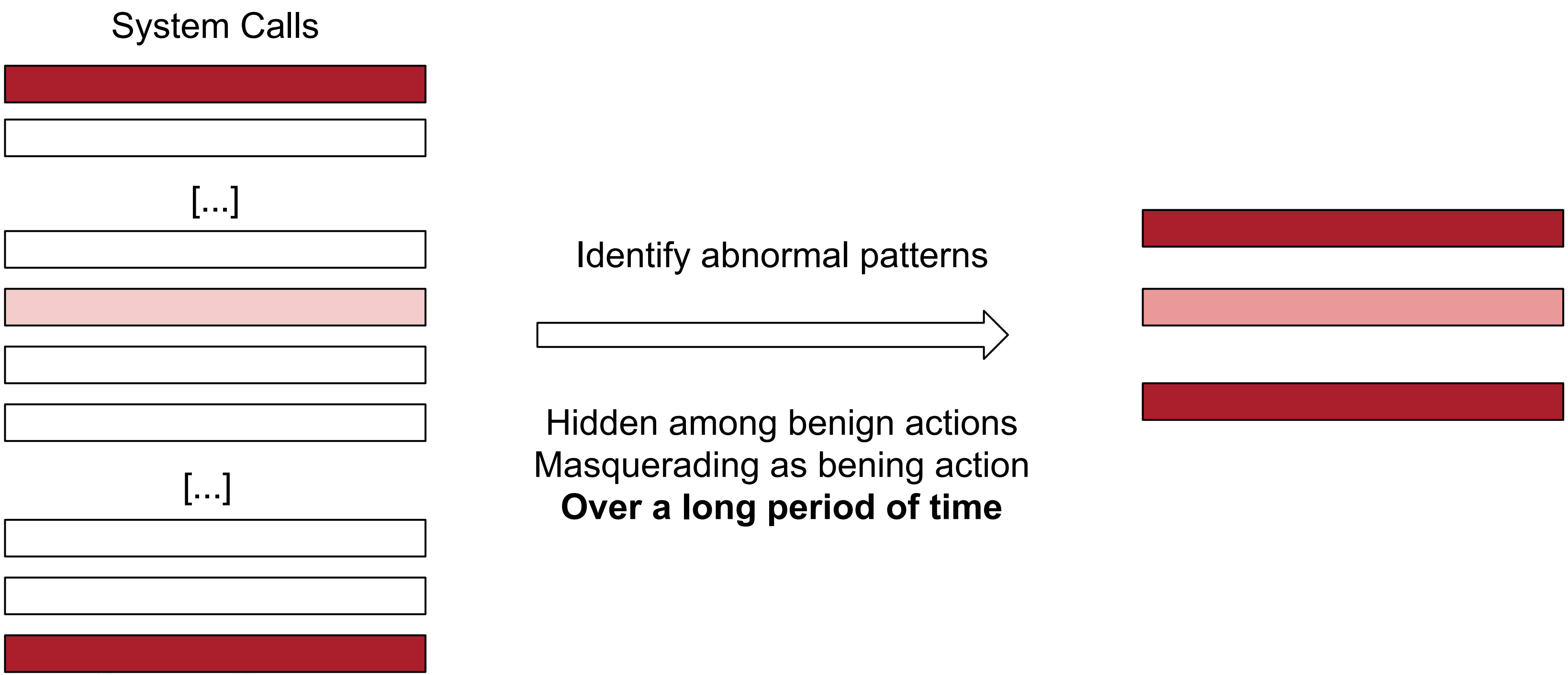
System call based intrusion detection



System call based intrusion detection



System call based intrusion detection



How to solve this with provenance?

- What is provenance?
- Why use provenance?
- How to capture provenance?
- How to perform detection?
- How to evaluate?
- Insights

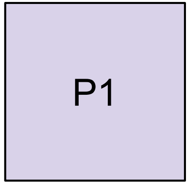
University of BRISTOL

What is provenance?

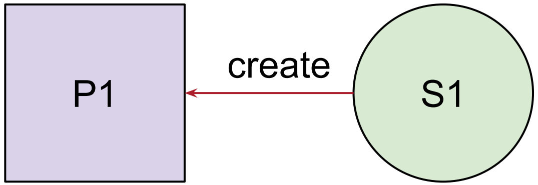
System-level provenance graph

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

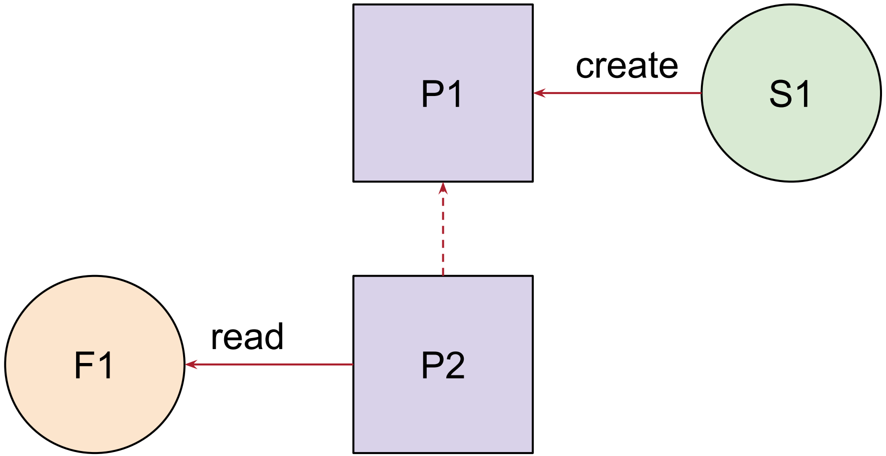
Example provenance (simplified)



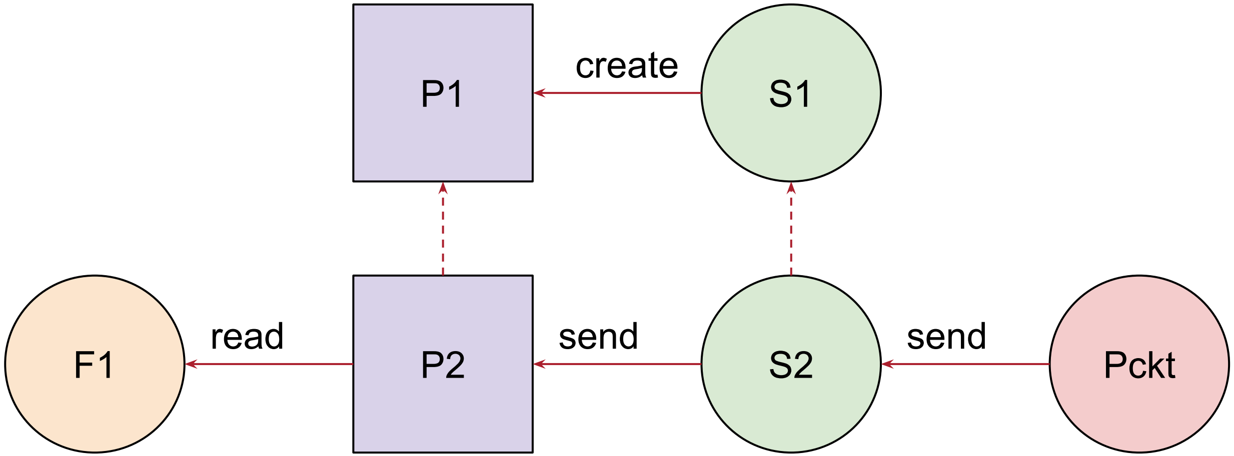
Example provenance (simplified)



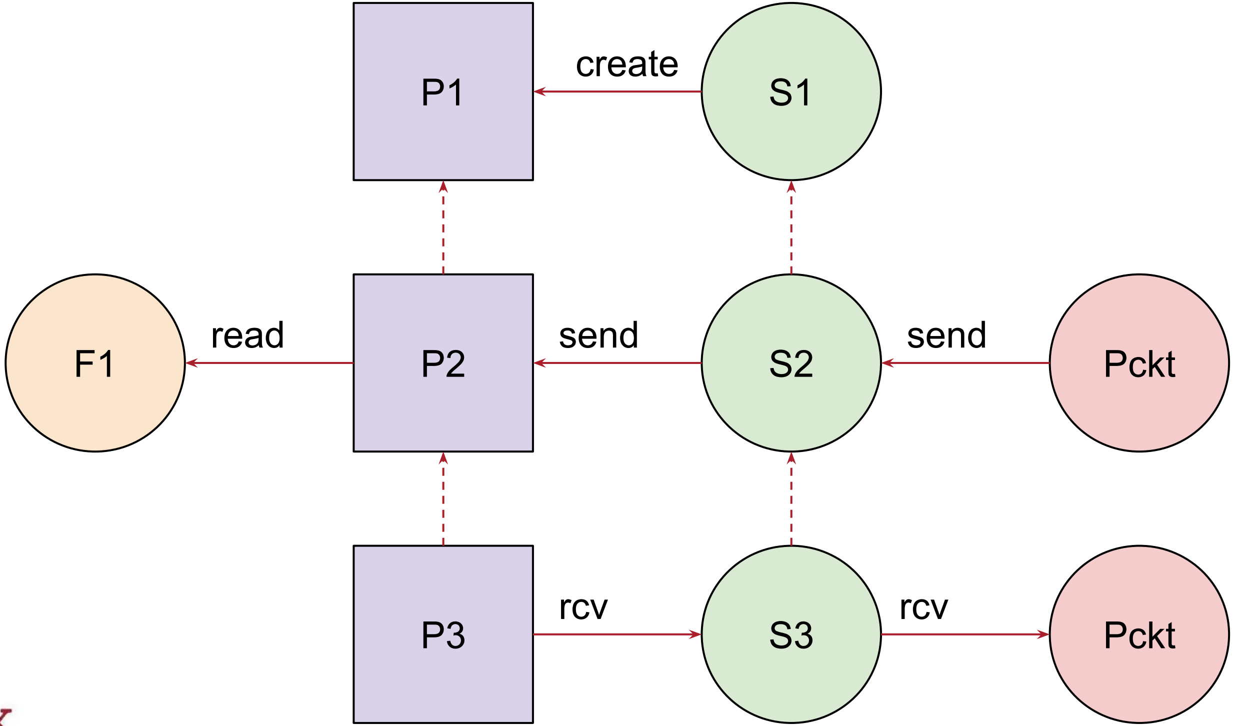
Example provenance (simplified)



Example provenance (simplified)

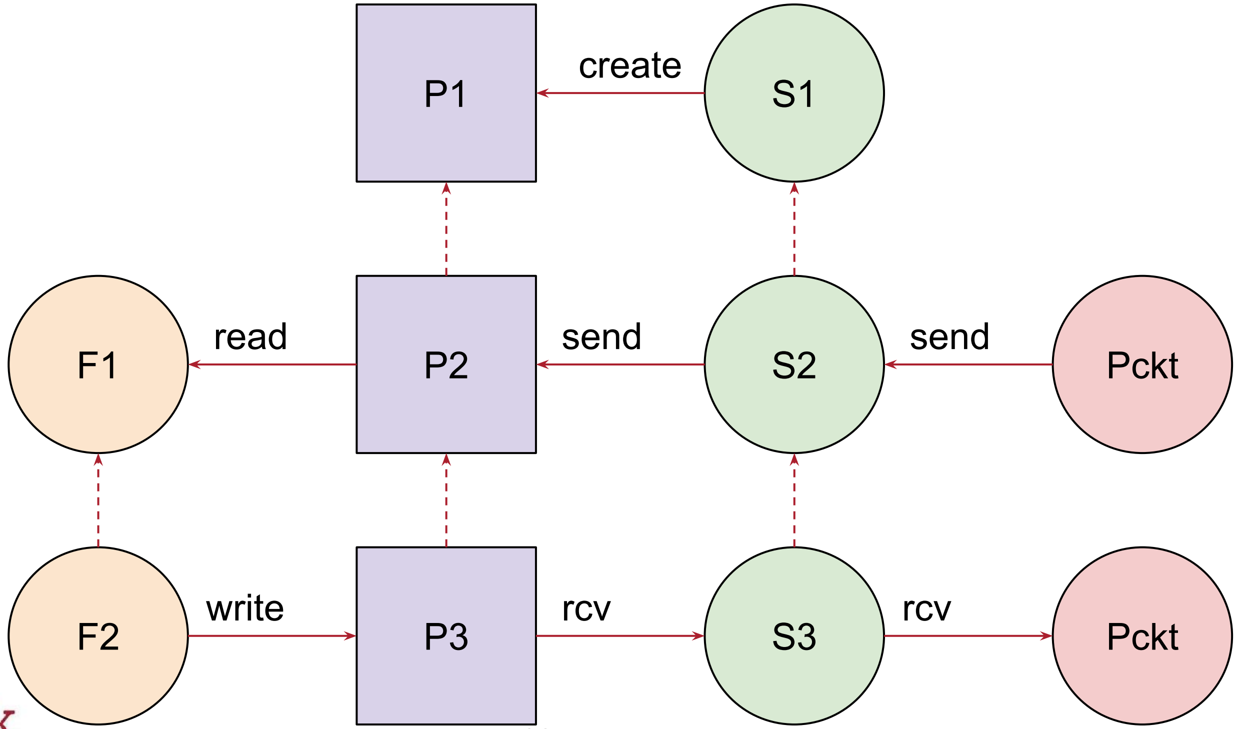


Example provenance (simplified)



bristol.ac.uk

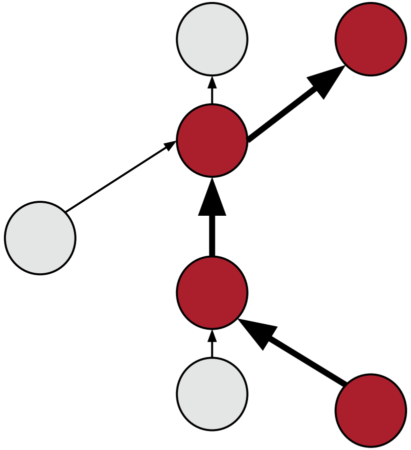
Example provenance (simplified)



Why use provenance?

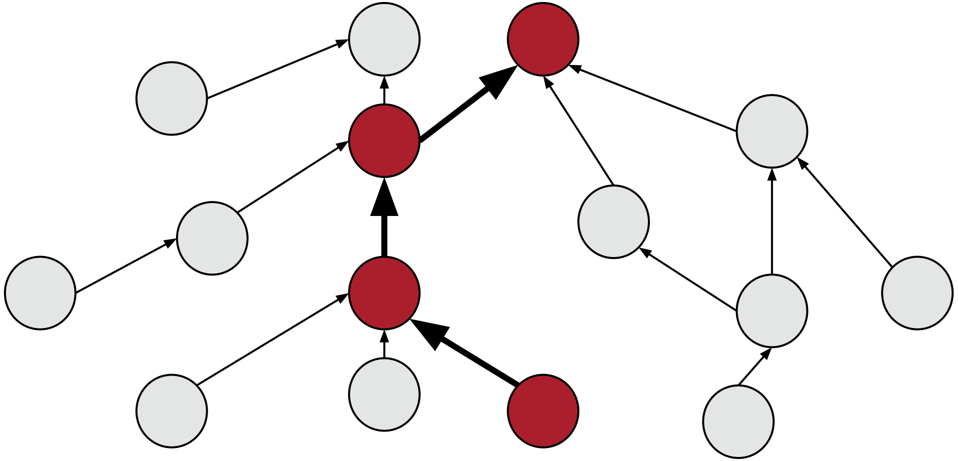
Provenance-based intrusion detection

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



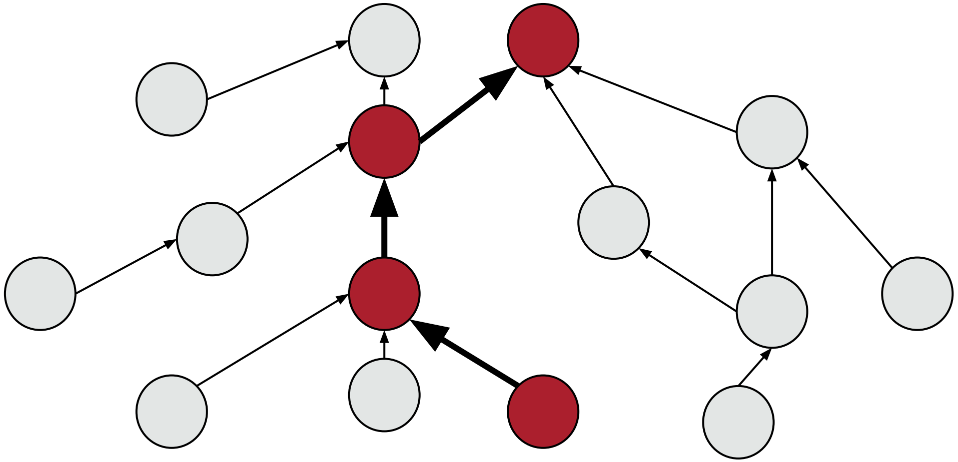
Provenance-based intrusion detection

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



Provenance-based intrusion detection

- Related events are connected even across long period of time

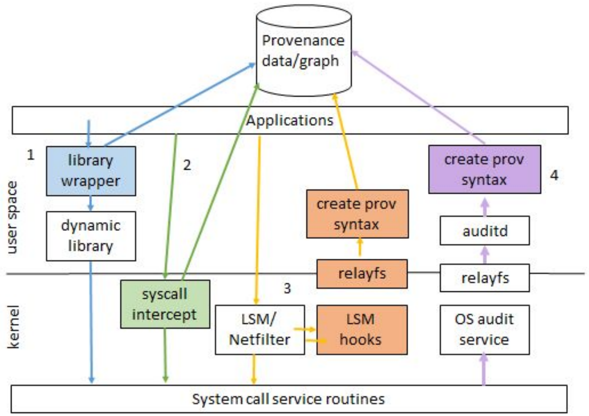


bristol.ac.uk

University of BRISTOL

How to capture provenance?

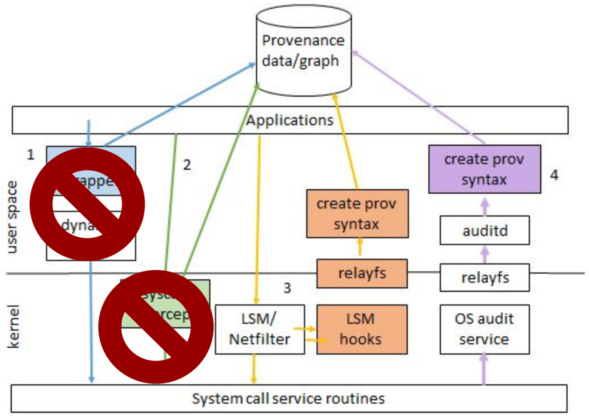
Capture methods



Examples

1. Balakrishnan et al. "**OPUS: A Lightweight System for Observational Provenance in User Space**" *Workshop on the Theory and Practice of Provenance*. 2013
2. Muniswamy-Reddy et al. "**Provenance-aware storage systems**" *USENIX ATC.* 2006.
3. Pasquier et al. "**Practical whole-system provenance capture**" *SoCC.* 2017
4. Gehani et al. "**SPADE: support for provenance auditing in distributed environments**" Middleware Conference. 2012

Capture methods



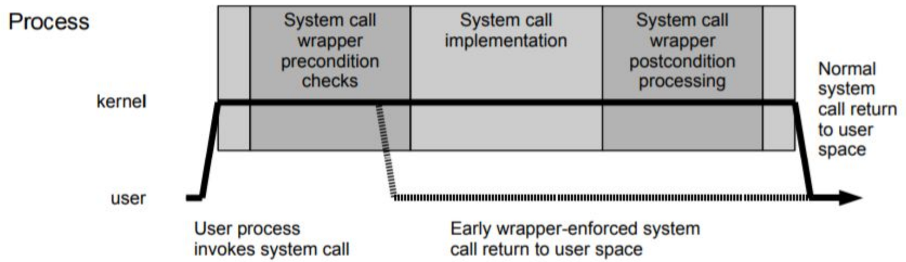
Examples

1. Balakrishnan et al. "**OPUS: A Lightweight System for Observational Provenance in User Space**" *Workshop on the Theory and Practice of Provenance*. 2013
2. Muniswamy-Reddy et al. "**Provenance-aware storage systems**" *USENIX ATC*. 2006.
3. Pasquier et al. "**Practical whole-system provenance capture**" *SoCC*. 2017
4. Gehani et al. "**SPADE: support for provenance auditing in distributed environments**" Middleware Conference. 2012

University of BRISTOL

Why are they not appropriate?

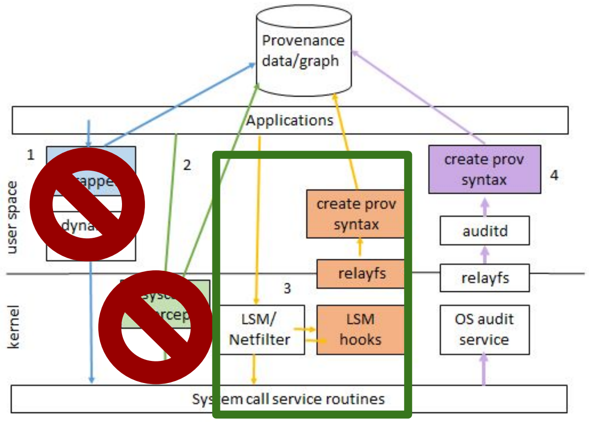
Interposition is unsafe



- Watson "**Exploiting Concurrency Vulnerabilities in System Call Wrappers**" WOOT. 2007

- **Time-of-audit-to-time-of-use attack**
  - Race condition
- Syntactic Race
  - **different copy of parameters**
- Semantic Race
  - **Kernel state may change**

Capture methods



Examples

1. Based on Linux reference monitor
2. Best accuracy
3. Stronger formal guarantees
4. Formally specified semantic
5. Best performance

Pasquier et al. **“Runtime Analysis of Whole-System Provenance”**, CCS 2018

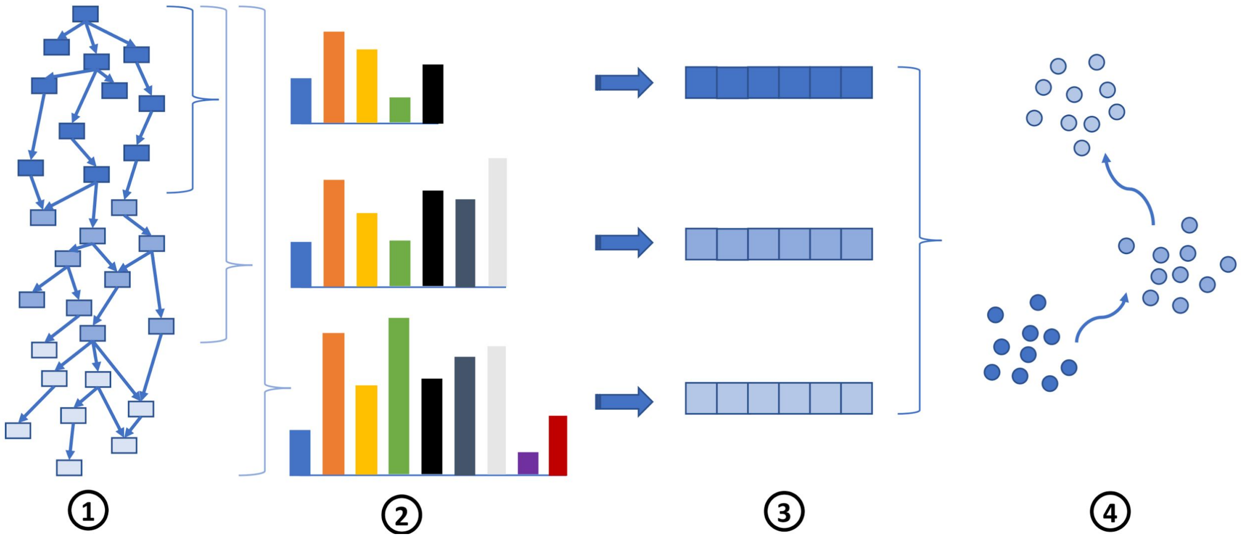
University of BRISTOL

How to perform detection?

Assumptions (and limitations)

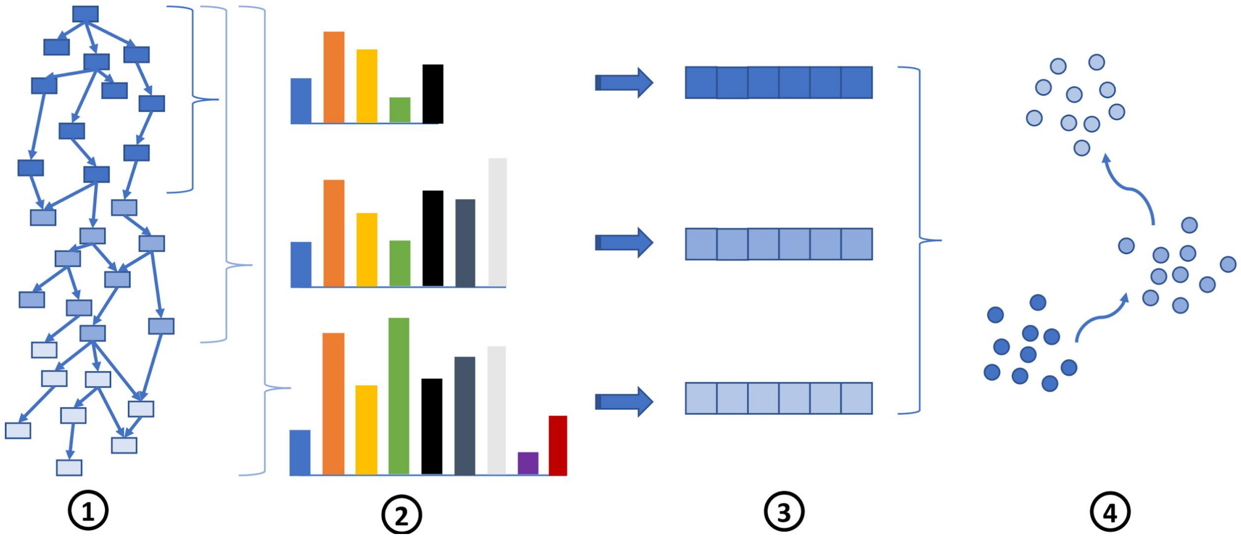
- **Runtime detection**
- We target environment with **minimal human intervention**
  - relatively consistent behaviour
  - e.g. web servers, CI pipelines etc...
- Build a **model of system behaviour** (unsupervised training)
  - in a controlled environment
  - from a representative workload (this is hard!)
- **Detect deviation** from the model
- Several approaches being explored…

Example



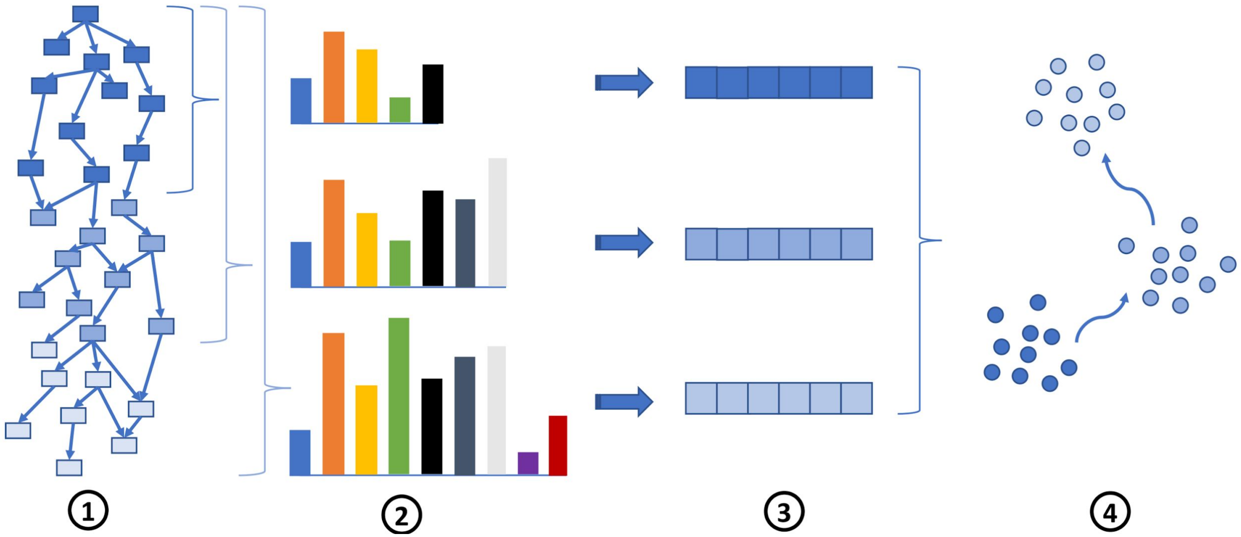
- Han et al. **"UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats"**, NDSS 2020

Example



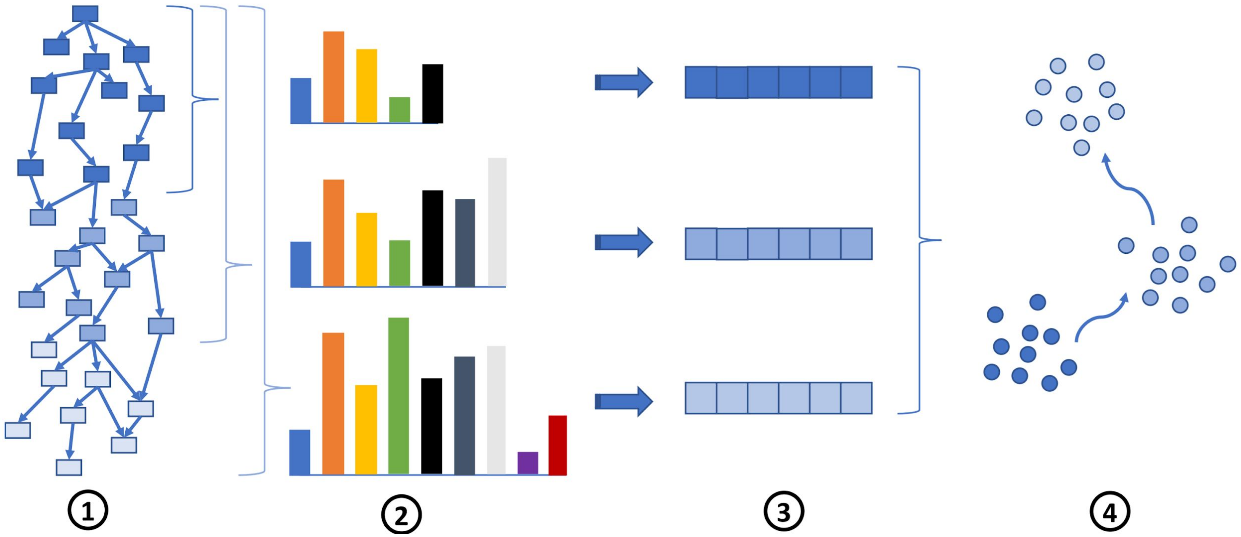
1) Graph streamed in, converted to histogram, labelled using (modified) **struct2vec**

Example



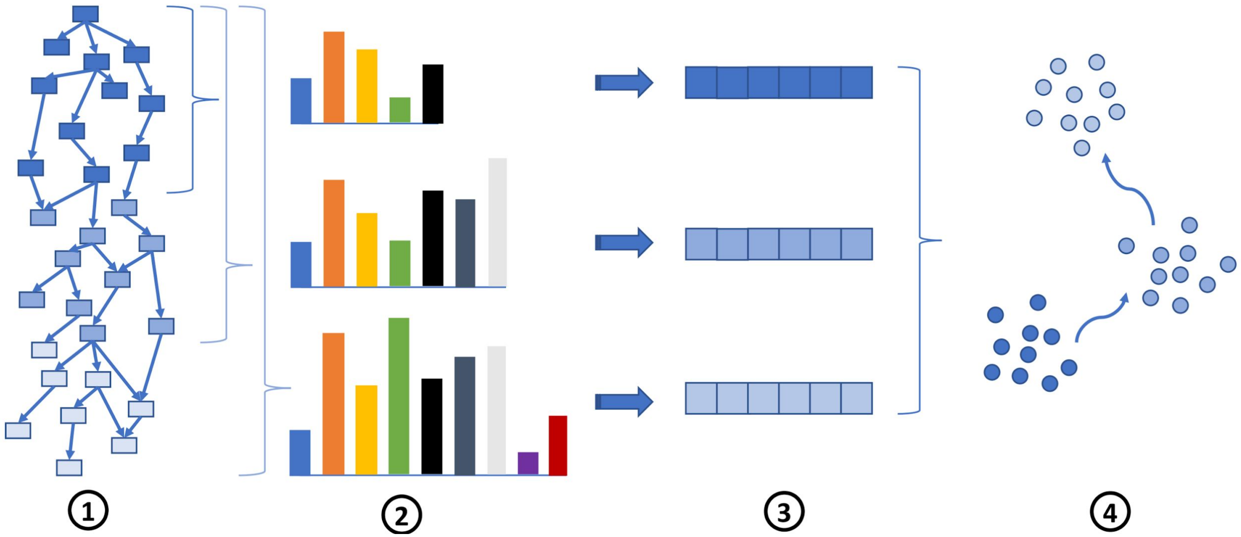
2) At regular interval, histogram converted to a fixed size vector using **similarity preserving hashing**

Example



3) Feature vectors are **clustered**

Example



4) Cluster forms “**meta-state**”, transitions are modelled

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

Relatively simple

- Nothing overly fancy here
- Labelled directed acyclic graph
  - node/edge types
  - security context (when available)
- Modification and combination of existing algorithms
  - struct2vec
  - similarity preserving hashing
  - clustering
- Right combination + domain knowledge

How to evaluate?

Comparison state of the art

Experiment	Dataset	# of Graphs	Avg. $|V|$	Avg. $|E|$	Preprocessed 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

TABLE I: Characteristics of the StreamSpot dataset. The dataset is publicly available only in a preprocessed format.

Experiment	Precision	Recall	Accuracy	F-Score
StreamSpot (baseline)	0.74	N/A	0.66	N/A
$R = 1$	0.51	1.0	0.60	0.68
$R = 3$	0.98	0.93	0.96	0.94

TABLE II: Comparison to StreamSpot on the StreamSpot dataset. We estimate StreamSpot's average accuracy and precision from the figure included in the paper [83], which does not report exact values. They did not report recall or F-score.

Manzoor et al. "**Fast memory-efficient anomaly detection in streaming heterogeneous graphs**" ACM KDD, 2016.

R -> neighborhood size for struct2vec algorithm

Evaluation with DARPA datasets

Experiment	Dataset	# of Graphs	Avg. |V|	Avg. |E|	Raw Data Size (GiB)
DARPA	Benign	66	59,983	4,811,836	271
CADETS	Attack	8	386,548	5,160,963	38
DARPA	Benign	43	2,309	4,199,309	441
ClearScope	Attack	51	11,769	4,273,003	432
DARPA	Benign	2	19,461	1,913,202	4
THEIA	Attack	25	275,822	4,073,621	85

TABLE IV: Characteristics of graph datasets used in the DARPA experiments.

Experiment	Precision	Recall	Accuracy	F-Score
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

TABLE V: Experimental results of the DARPA datasets.

Evaluation with DARPA datasets

Experiment	Dataset	# of Graphs	Avg. |V|	Avg. |E|	Raw Data Size (GiB)
DARPA	Benign	66	59,983	4,811,836	271
CADETS	Attack	8	386,548	5,160,963	38
DARPA	Benign	43	2,309	4,199,309	441
ClearScope	Attack	51	11,769	4,273,003	432
DARPA	Benign	2	19,461	1,913,202	4
THEIA	Attack	25	275,822	4,073,621	85

TABLE IV: Characteristics of graph datasets used in the DARPA experiments.

Experiment	Precision	Recall	Accuracy	F-Score
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

TABLE V: Experimental results of the DARPA datasets.

SUCH GOOD RESULTS ARE NOT NORMAL

Building our own dataset

Experiment	Dataset	# of Graphs	Avg. |V|	Avg. |E|	Raw Data Size (GiB)
SC-1	Benign	125	265,424	975,226	64
	Attack	25	257,156	957,968	12
SC-2	Benign	125	238,338	911,153	59
	Attack	25	243,658	949,887	12

TABLE VI: Characteristics of the datasets used in the supply-chain APT attack experiments.

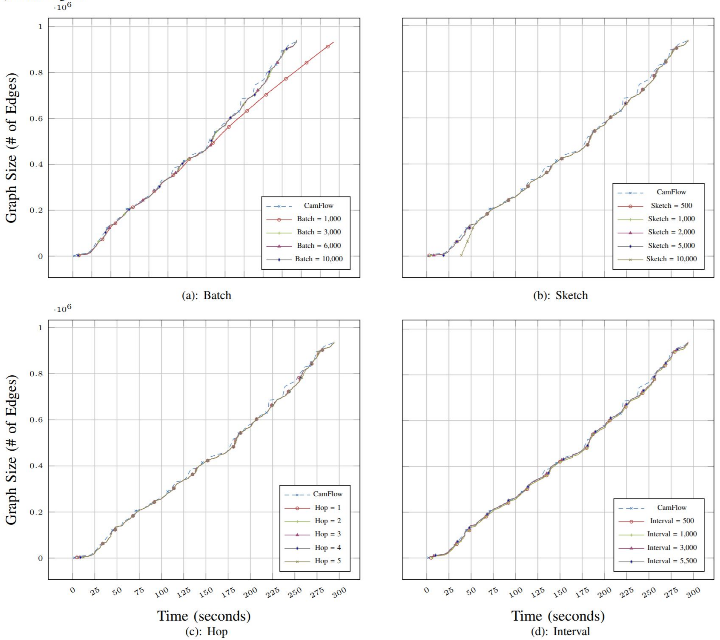
Experiment	Precision	Recall	Accuracy	F-Score
SC-1	0.85	0.96	0.90	0.90
SC-2	0.75	0.80	0.77	0.78

TABLE VIII: Experimental results of the supply-chain APT attack scenarios.

- Attack designed to look similar to background activity

Runtime performance

Fig. 4: Total number of processed edges over time (in seconds) in the SC-1 experimental workload with varying batch sizes (Fig. 4(a)), sketch sizes (Fig. 4(b)), hop counts (Fig. 4(c)), and intervals of sketch generation (Fig. 4(d)). Dashed blue line represents the speed of graph edges streamed into UNICORN for analysis. Triangle maroon baseline has the same configurations as those used in our experiments and indicates the values of the controlled parameters (that remain constant) in each figure.



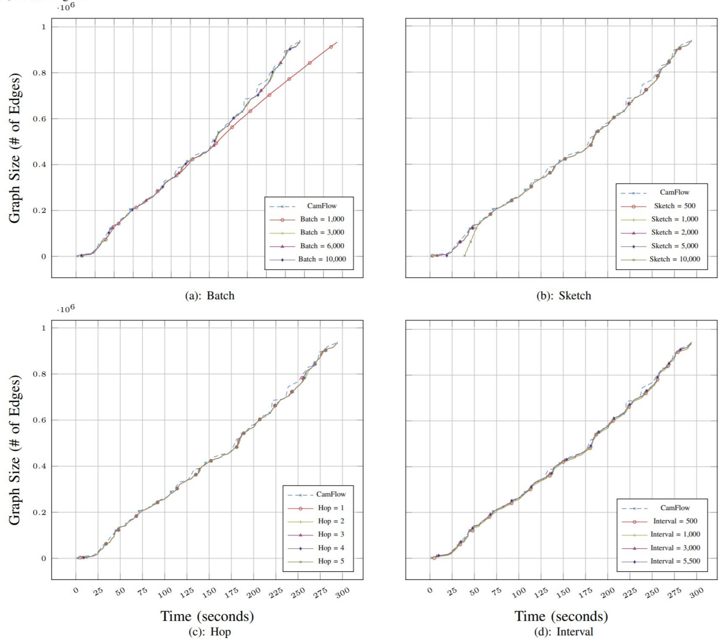
F. CPU & Memory Utilization

Configuration Parameter	Parameter Value	Max Memory Usage (MB)
	R = 1	562

Runtime performance

Memory usage: ~500MB
CPU usage 15% on 1 core

Fig. 4: Total number of processed edges over time (in seconds) in the SC-1 experimental workload with varying batch sizes (Fig. 4(a)), sketch sizes (Fig. 4(b)), hop counts (Fig. 4(c)), and intervals of sketch generation (Fig. 4(d)). Dashed blue line represents the speed of graph edges streamed into UNICORN for analysis. Triangle maroon baseline has the same configurations as those used in our experiments and indicates the values of the controlled parameters (that remain constant) in each figure.



F. CPU & Memory Utilization

Configuration Parameter	Parameter Value	Max Memory Usage (MB)
	R = 1	562

bristol.ac.uk

Insights

Provenance-based IDS work!

- 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!

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 (i.e. ingest different data type)

Explainability is a problem

- There is a problem within the last batch of X graph elements
  - 2,000 in previous figures
- Good luck finding out what went wrong
- Provenance forensic is an active field of research
  - Promising work out of the DARPA programme
- … but could we do better during detection?
  - Promising work with colleagues at NEC Labs America
  - Report vertex within 3 nodes of anomaly in 75% of cases!
  - Deep graph learning techniques

University of BRISTOL

What about unpredictable workload?

Everything has been open-sourced

- Capture: http://camflow.org
  - Linux package(s) available!
- Data management: https://github.com/ashish-gehani/SPADE/wiki
- IDS: https://github.com/crimson-unicorn
- Streamspot data: https://github.com/sbustreamspot/sbustreamspot-data
- DARPA data: https://github.com/darpa-i2o/Transparent-Computing

For those really interested (20+ papers on the topic):

- https://github.com/tfjmp/provenance-papers

University of BRISTOL

Thank you, questions?

tfjmp.org
thomas.pasquier@bristol.ac.uk