Inference from trees: a cybersecurity example

Tom, GCHQ

The problem

We have data representing the events that occur on a computer.

(known as *host* or *endpoint* data)

For example:

- Reading/writing/modifying files.
- Making network connections.
- Setting off processes.

Within the data, we would like to detect malicious activity.

In this talk:

- How can we use path signatures to detect malicious behaviour?
- What are the practical considerations when using such a technique?

Background: Logging Made Easy (LME)

An initiative by GCHQ's National Cyber Security Centre (NCSC)

- **Aim:** to provide a baseline level of cybersecurity for small enterprises.
- **Contents:** Recommendations for logging activity on systems and networks, based on the Windows *sysmon* tool.
- **Output:** event logs, with a timeline of events associated with processes on the system.

Example data

Timestamp	Process ID	Event type	Process created	Details
2020-02-02 02:02:00.000	1	FILE_MODIFY		Filename=…
2020-02-02 02:02:00.222	1	PROCESS_CREATE	2	
2020-02-02 02:02:00.222	2	PROCESS_INIT		
2020-02-02 02:02:02.000	2	NETWORK_MESSAGE		Port=…
2020-02-02 02:02:02.020	1	PROCESS_TERMINATE		

Why machine learning approaches?

- Traditional methods are based on known malicious patterns, eg:
  - Modifying files in specific locations.
  - Types of network connections.
- By applying machine learning, we hope to detect more general suspicious patterns of behaviour.

Challenges applying ML to cybersecurity

- Lack of widely-available data:
  - Many of the available data sets are **not real-world data**, but simulated.
  - There is only a **small amount of labelled activity** that we know is malicious.
- Malicious activity takes many forms.
- Data can be messy: what features do we extract?

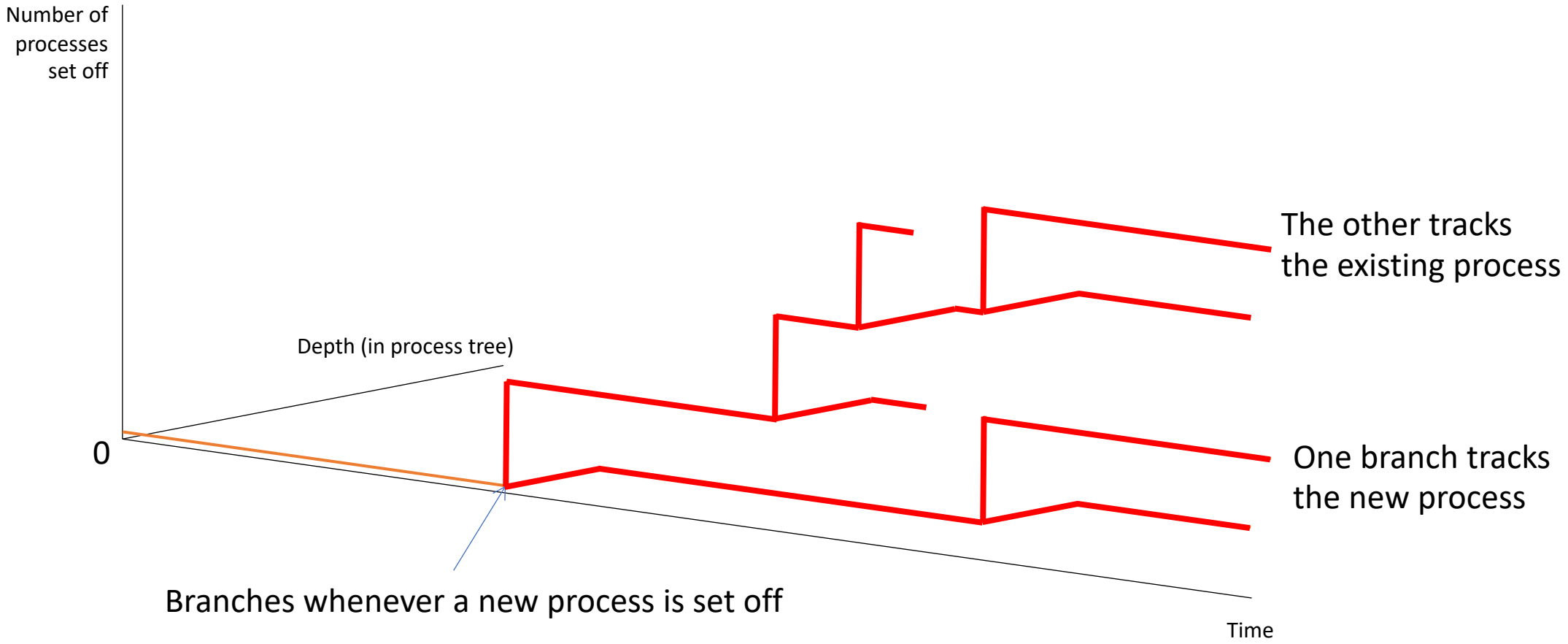
The challenge we attempt to address here

Modelling the data

Modelling the data as an evolving tree

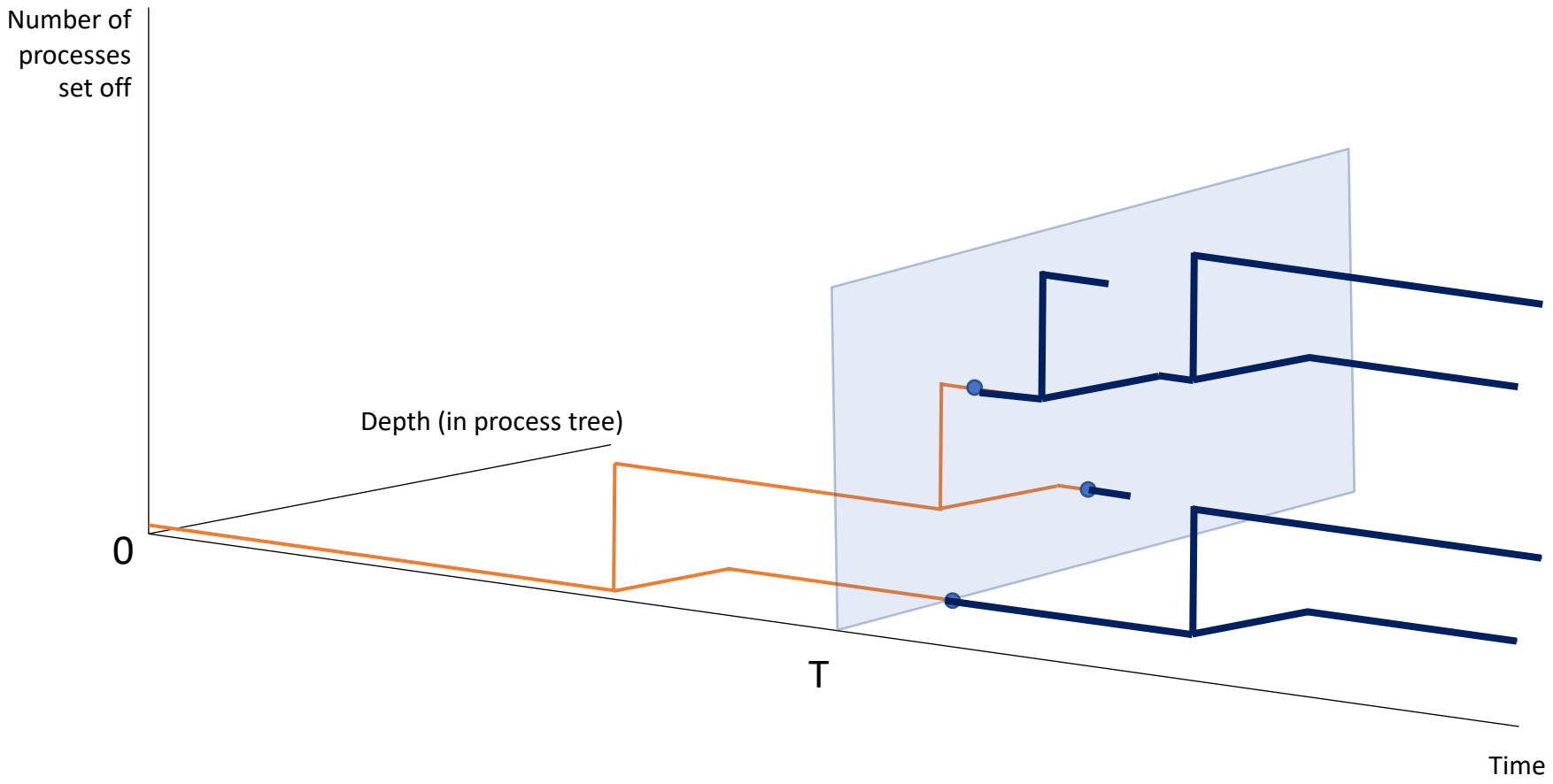
- Take the events associated with a process.
- We want to represent the data in such a way that we can apply (expected) path signatures, and extract features.
- To do this, we first consider **process creation** events:
  - At each **timestamp** when a new process is created, we see
  - the new **child process** created
  - and the **parent process** responsible for its creation

From process creation events, we create a tree as follows:

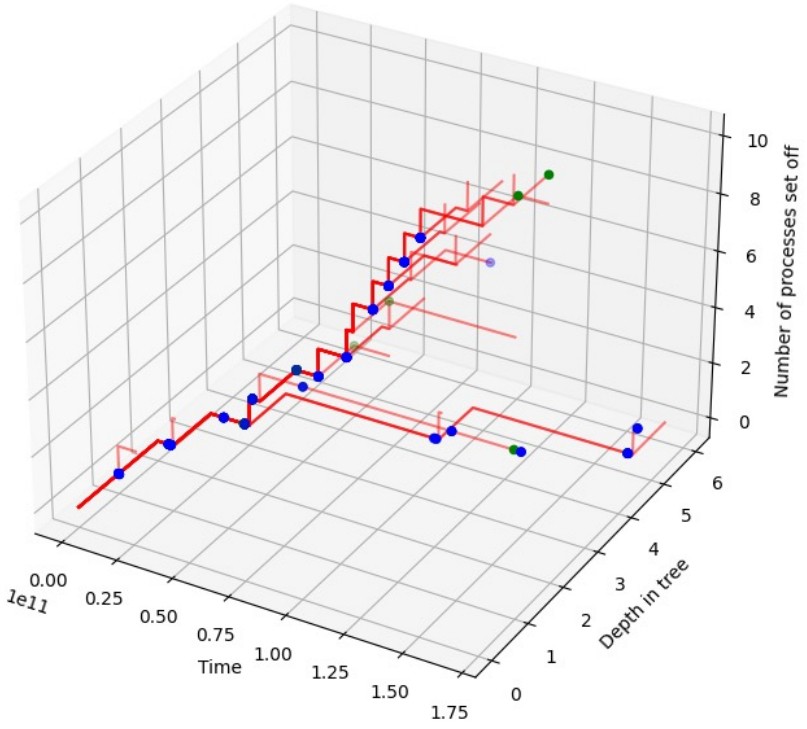


Time windows

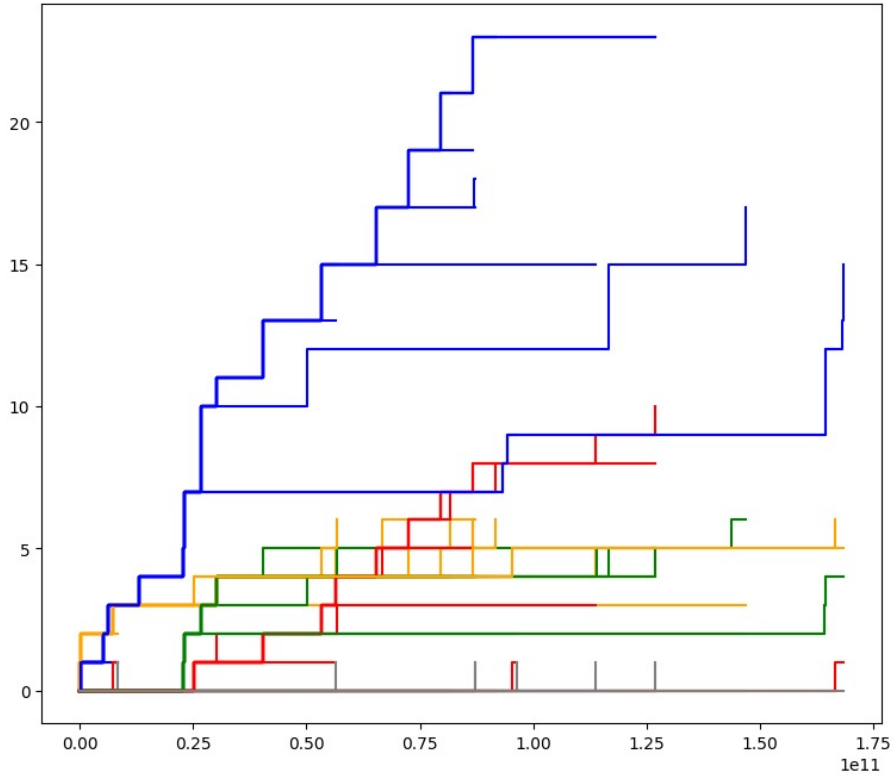
In practice, trees can become large so we split them into time windows:



Now consider the various other types of events set off by each process:



The dots represent **file** or **network** events occurring along a branch

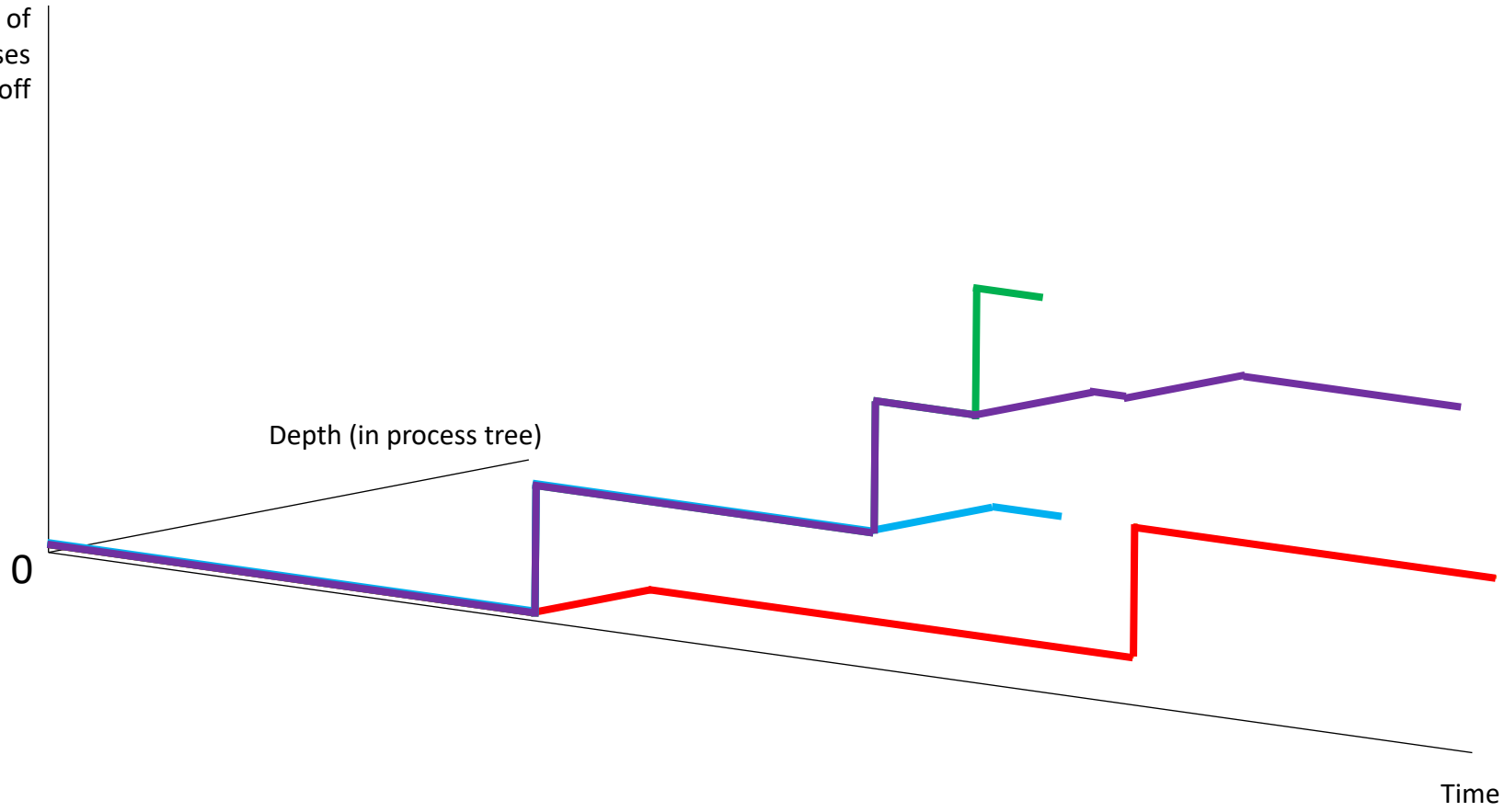


When we see a **file** or **network** event, we increment its count along that branch

Represent each type of event with its own dimension.

Result: a tree in n+3 dimensions, where n is the number of event types)

A tree can be considered as a *collection of paths*: they just happen to share part of their trajectory.



This gives us a natural feature vector: the **expected signature** of the collection of paths.

An example: classification problem

Setup: We have two sets of processes:

- ‘Cleanware’ processes, known to be benign.
- Malware processes, known to be malicious.

Procedure:

- Create a tree from each process’s events.
- Compute the expected signature, to give a feature vector for each process.
- Train a classifier on this labelled data, to distinguish benign vs malicious.

Results:

In this test scenario, incorporating signature features gives improved accuracy vs simple counts of events.

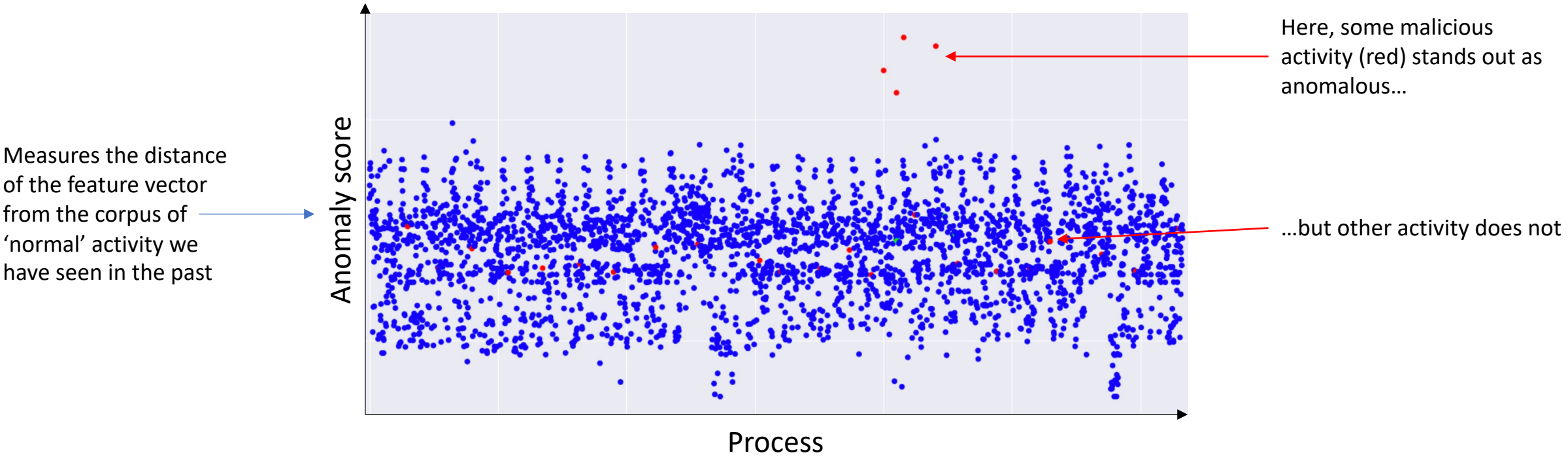
Using the technique in practice

In practice…

- Real-world scenarios are messier.
- This would be used as part of a suite of tools: we wouldn’t expect it to detect all types of malicious activity by itself.
- Focus on the blind spots of traditional ‘fingerprint-based’ methods:
  - For example, unknown types of malicious activity.

Anomaly detection problem

- Here the practical problem is closer to anomaly detection.
  - We don't necessarily know what malicious activity will look like.
  - But we hope it looks *anomalous*, compared to what we've seen before.



Anomaly detection problem

- The practical problem is closer to anomaly detection.
  - We don’t necessarily know what malicious activity will look like.
  - But we hope it looks *anomalous*, compared to what we’ve seen before.
- Ultimate goal: flag anomalous activity for further (manual) analysis.
  - Care is needed: there are legitimate reasons for activity to look anomalous.
  - Need a low false positive rate, to avoid overwhelming analysts!

Conclusion

- We can extract features from host event data using **path signatures on trees.**
- These show promise at being able to distinguish different types of behaviour.
- Incorporating them into a practically useful technique for anomaly detection is still an open problem.