Challenges in Learning from Behaviour

David Aspinall

Prof. Software Safety and Security, University of Edinburgh.

With thanks to **Henry Clausen, Robert Flood** and others

AI-CyberSec 2021

Outline

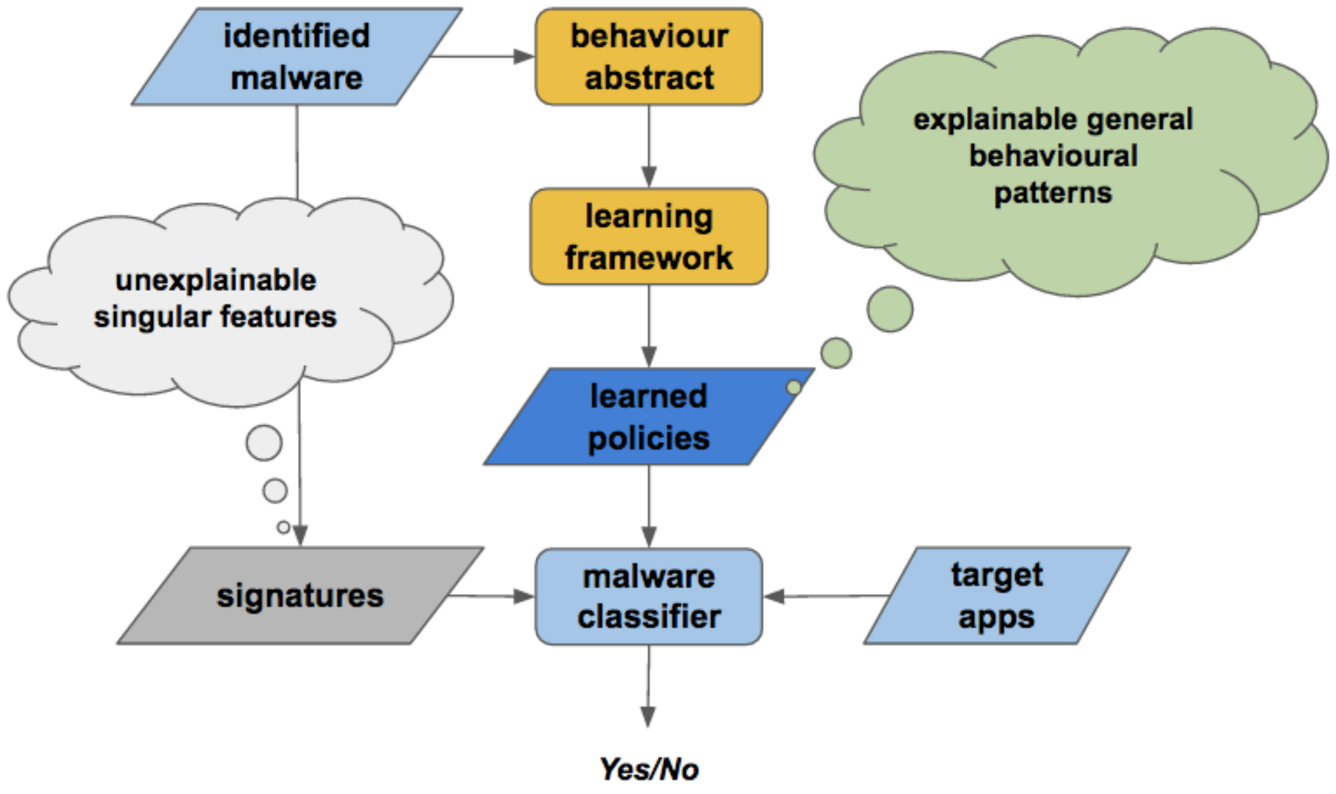
- From software behaviour to network behaviour
- Challenges
    1. collecting and sharing the data
    2. ensuring sufficient variety
    3. understanding model performance
- Outlook for future work

Learning Malware Behaviour

- Program analysis extracts *behaviour*
- Learn common bad patterns
- Train a classifier to recognise
- Robust against diversification, obfuscation.

Intuition:

If it behaves like a bad app, it is a bad app!



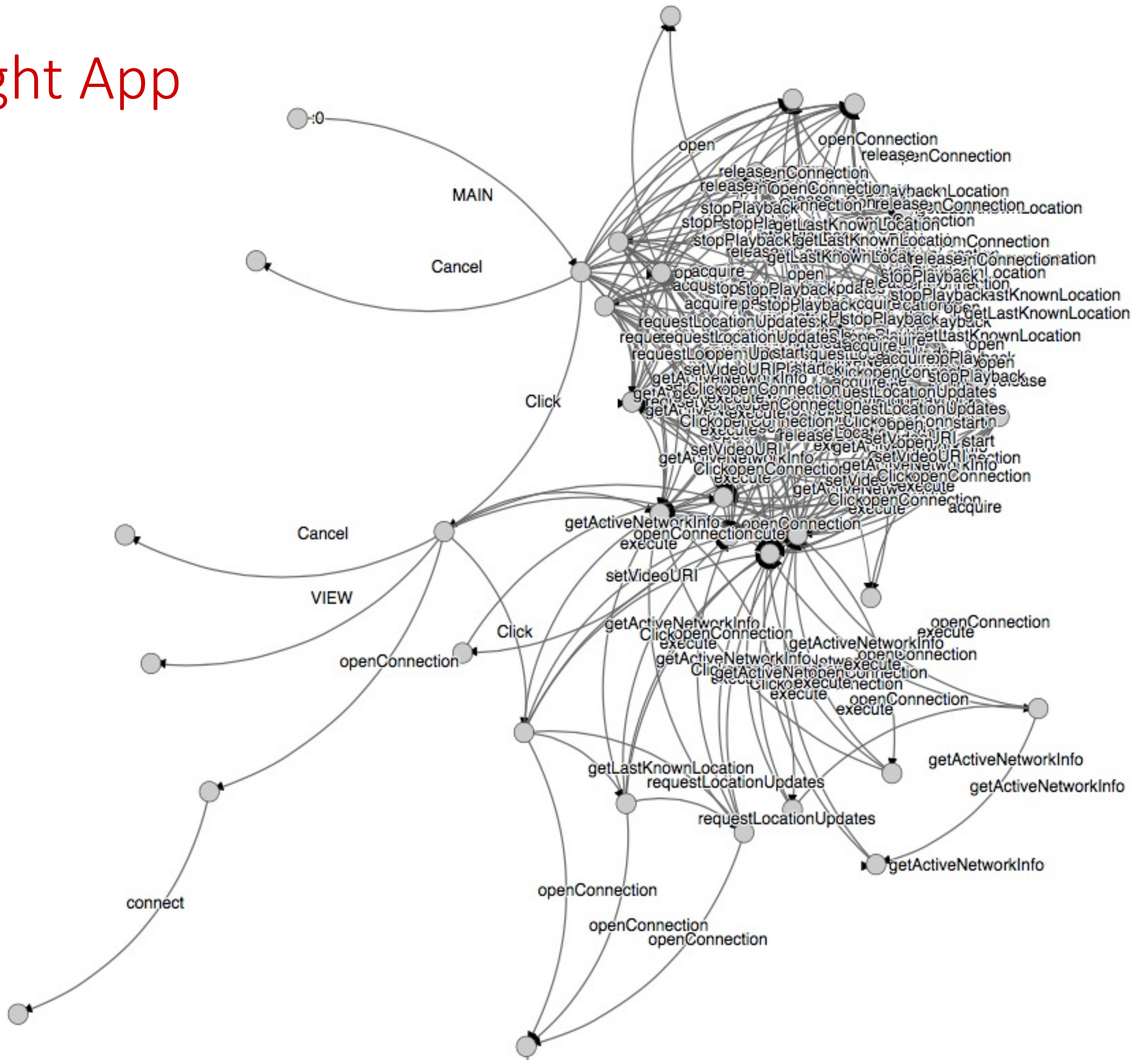
Predicting the Security Behaviour of Mobile Apps. UoE, 2017-20.
App Guarden: Secure App Stores. UoE, 2014-17.

Abstract Behaviours of a Flashlight App

Map out all things an app can do, approximately. Certain behaviour patterns considered bad by "policies" (rules).

Question: could we do this by external observations of activity, without looking at program code at all?

Perhaps even from **network traces**?!



Detection by Learning Software Models (2019-)

Our idea set out towards **Network Intrusion Detection** systems (NIDS):

1. gather interpretable external behaviours (network traces, log files)
2. learn interaction protocols (language grammars)
3. build approximate behaviours to construct software models
4. generate policies and rules to detect rogue behaviours

We’re still on the first step of this roadmap...

Data-driven Intrusion Detection

On the face of it, **anomaly detection** or **traffic classification** seem straightforward:

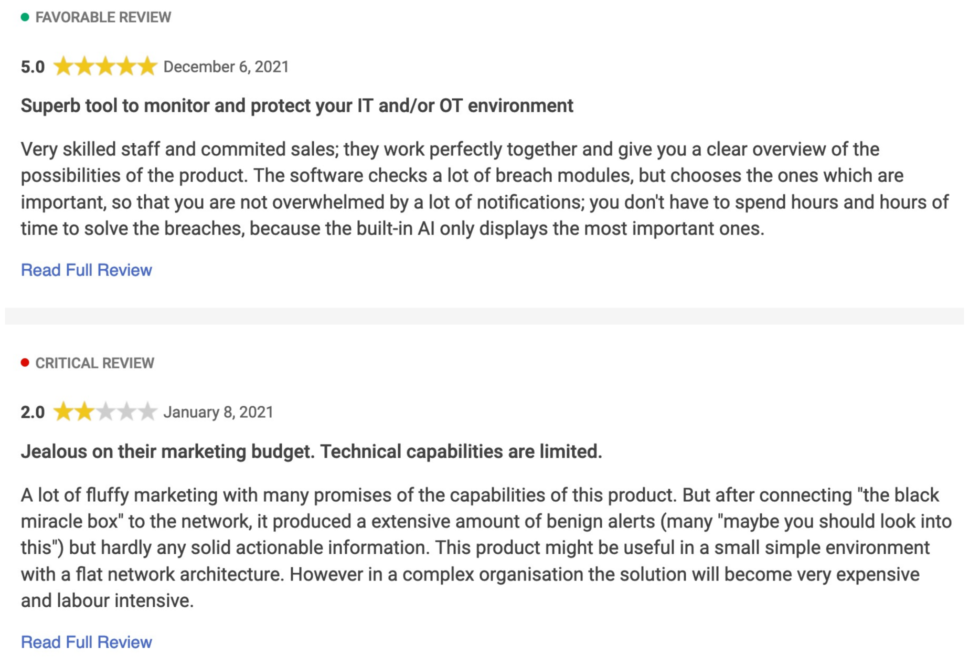
- train an Artificial Intelligence to recognise normal (and bad) behaviours
- raise alarms if suspicious (or known bad) behaviours appear

Despite almost three decades of research, fundamental challenges remain, e.g.:

- *volume and diversity of normal behaviours* + vanishingly small #s bad cases
- *rapid change in data baselines* + and few up-to-date public datasets
- *lack of accurate ground truth +* unclear benchmarks, few longitudinal evaluations

See: *Outside the Closed World: On Using Machine Learning for Network Intrusion Detection.* Sommer and Paxson, IEEE S&P 2010.

Promise of AI in Security



● FAVORABLE REVIEW

5.0 ★★★★★ December 6, 2021

Superb tool to monitor and protect your IT and/or OT environment

Very skilled staff and commited sales; they work perfectly together and give you a clear overview of the possibilities of the product. The software checks a lot of breach modules, but chooses the ones which are important, so that you are not overwhelmed by a lot of notifications; you don't have to spend hours and hours of time to solve the breaches, because the built-in AI only displays the most important ones.

Read Full Review

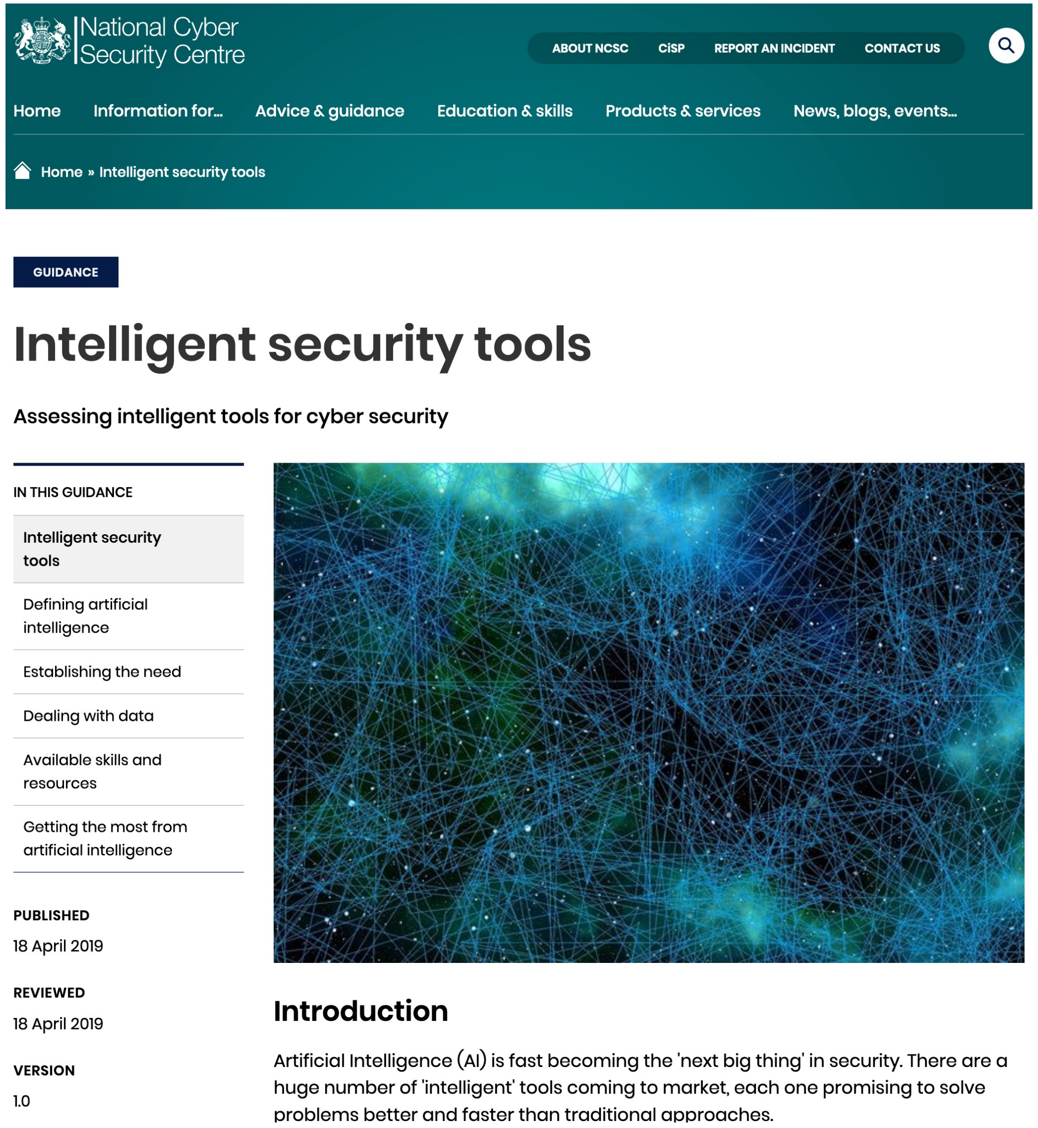
● CRITICAL REVIEW

2.0 ★★☆☆☆ January 8, 2021

Jealous on their marketing budget. Technical capabilities are limited.

A lot of fluffy marketing with many promises of the capabilities of this product. But after connecting "the black miracle box" to the network, it produced a extensive amount of benign alerts (many "maybe you should look into this") but hardly any solid actionable information. This product might be useful in a small simple environment with a flat network architecture. However in a complex organisation the solution will become very expensive and labour intensive.

Read Full Review



National Cyber Security Centre

ABOUT NCSC · CiSP · REPORT AN INCIDENT · CONTACT US

Home · Information for... · Advice & guidance · Education & skills · Products & services · News, blogs, events...

Home » Intelligent security tools

GUIDANCE

Intelligent security tools

Assessing intelligent tools for cyber security

IN THIS GUIDANCE

- Intelligent security tools
- Defining artificial intelligence
- Establishing the need
- Dealing with data
- Available skills and resources
- Getting the most from artificial intelligence

PUBLISHED
18 April 2019

REVIEWED
18 April 2019

VERSION
1.0

Introduction

Artificial Intelligence (AI) is fast becoming the 'next big thing' in security. There are a huge number of 'intelligent' tools coming to market, each one promising to solve problems better and faster than traditional approaches.

Challenge 1.

Collecting and Sharing Data

It’s the data, stupid!

Collecting and sharing data

We **must have high-quality open datasets to advance the field** as other domains have enjoyed. But:

- Tedious to collect and curate
- Real data is fraught with PII and security risk, needs anonymisation
- Synthetic data has issues: insufficient variety, accuracy other risks.

Consider the incentives behind data release...

DetGen: A Synthetic Data Tool

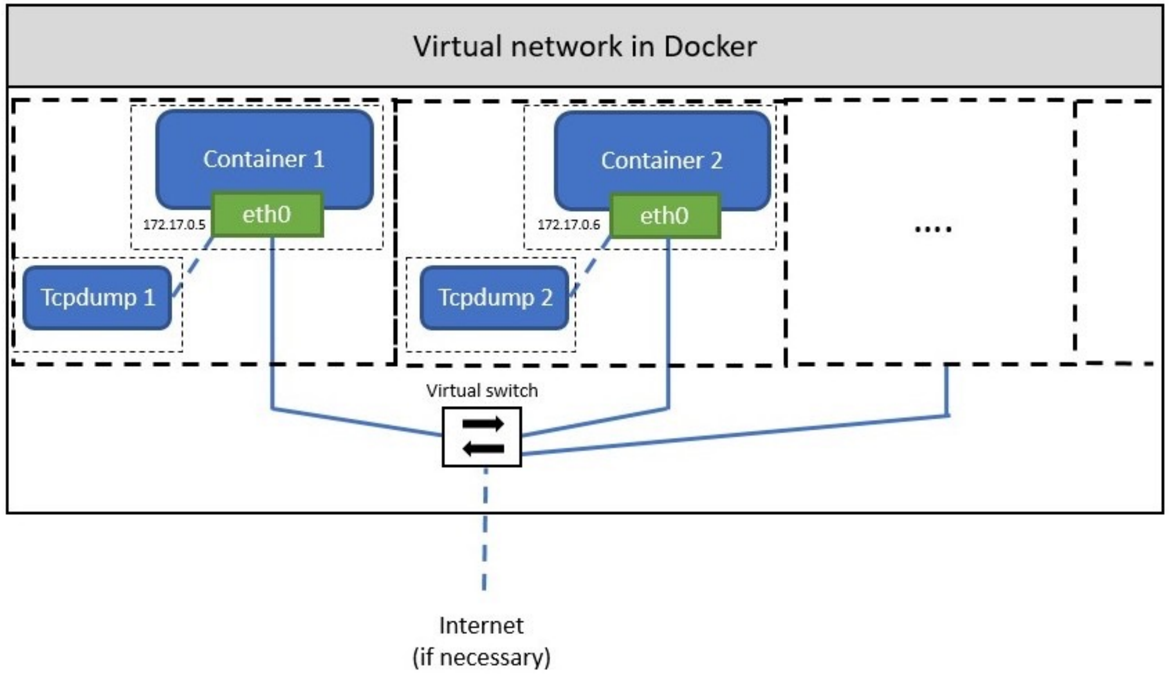
Idea: isolate single app behaviour.

Features:

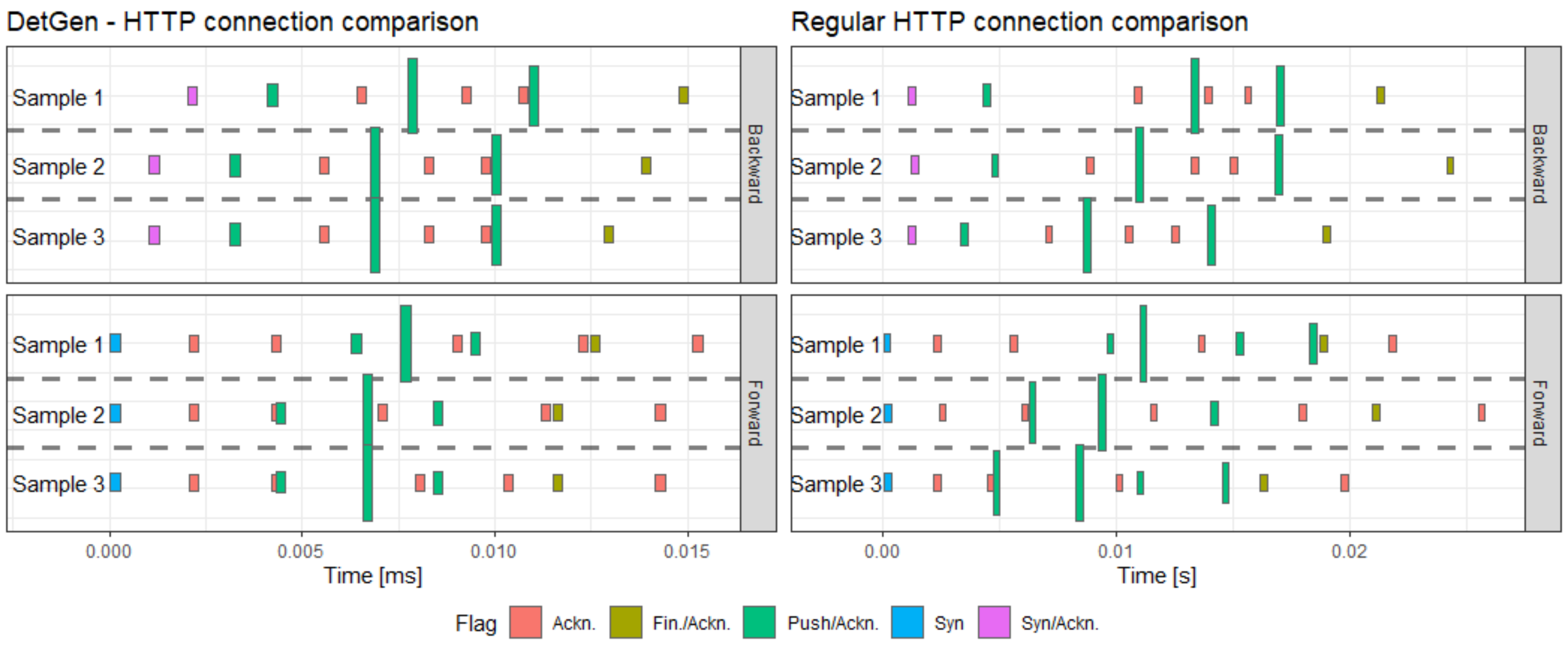
- scripted interaction scenarios
- provide built-in ground truth labels

Achieves, to an extent:

__Det__erministic data __Gen__eration.

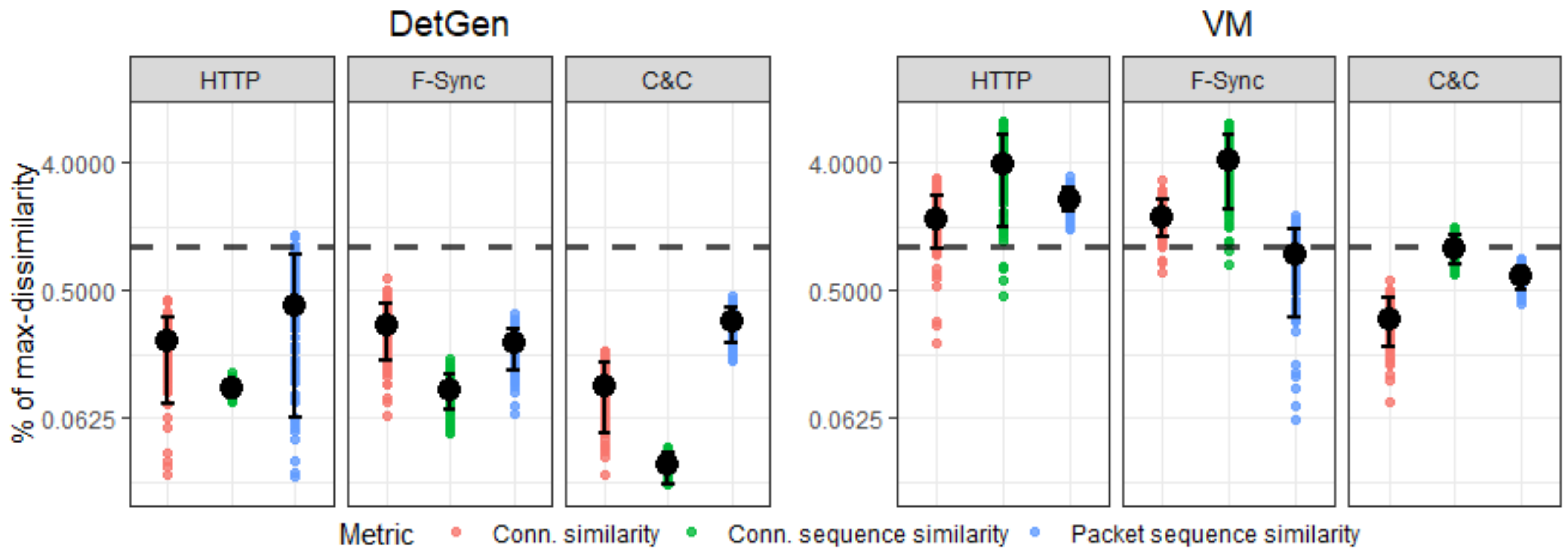


Evaluation: comparing against a Virtual Machine setting



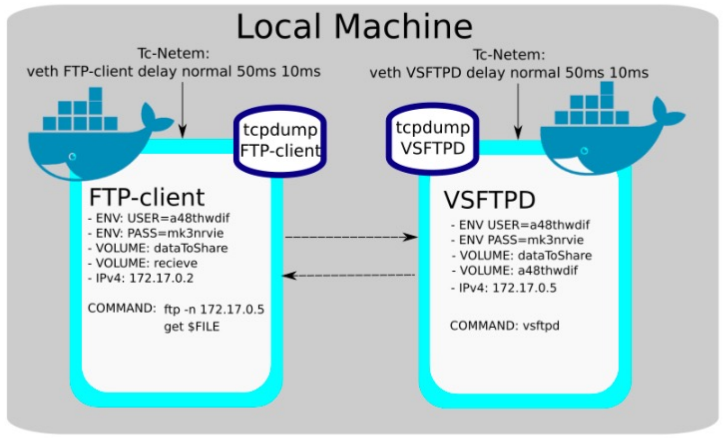
In VM we see greater spread of IATs. These **side-effects of simulation** are not inherent in HTTP software tested. So would like to minimise/remove them, try to ensure detection models are robust.

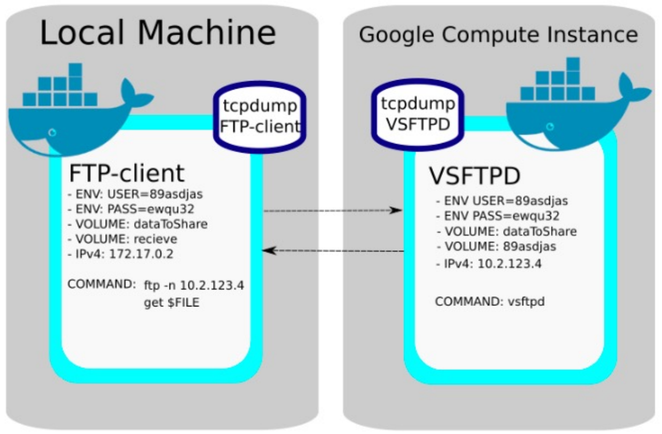
Evaluation: measuring determinism of DetGen

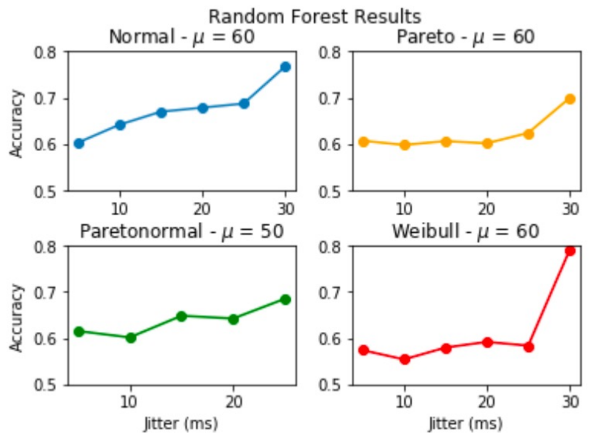


Exact determinism isn't possible but DetGen achieves greater reproducibility (less variation) compared to a simplified VM setting. Real networks or complex VMs would show much higher variation.

Evaluation: calibrating with real-world captures







A simple experiment shows that we should be able to re-calibrate or simulate real-world behaviours by manipulating the captures to insert delays.

Challenge 2:

Ensuring Variety of Data

Ensuring Variety of Data

In general, we want flexible ways to generate enough variety in data.

For example:

- to compare different software versions
- to mix data together (combining, injecting attacks)
- to generate data that wasn’t there

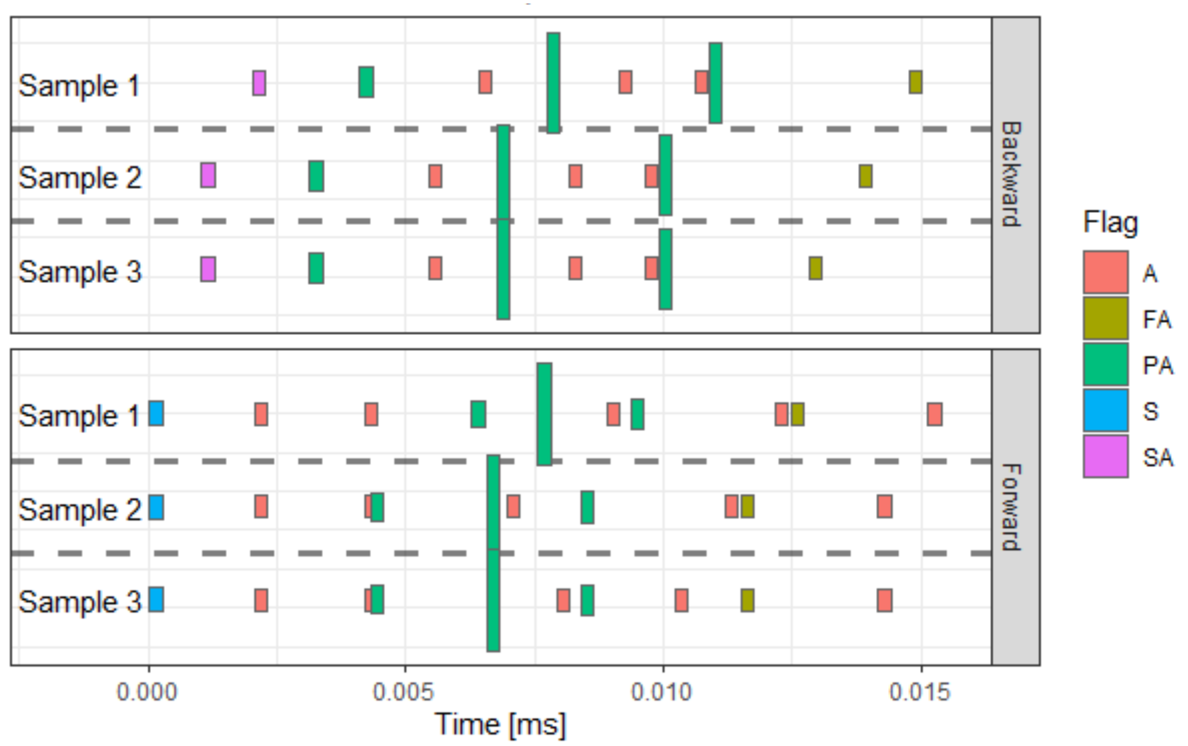
We want to understand various factors that need to be controlled for data generation and capture: “influence factors”.

Microstructures in traffic

A “**microstructure”** is a short-term structure corresponding to a particular activity, manifested by characteristic sequences at packet or connection level.

State-of-the-art IDS models use these to fingerprint or detect anomalous behaviour.

For robust models, we need robust microstructure notions.



Example microstructure for HTTP-get request, showing common characteristic structure

Factors influencing traffic and potential microstructures

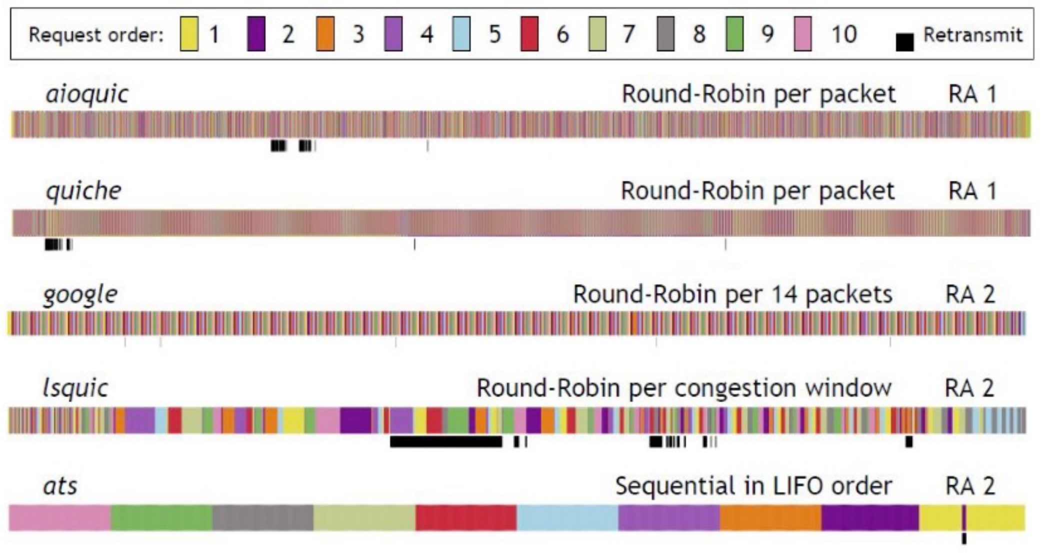
There are many!   For example:

- Application/task
- Implementation and version of
- Network congestion
- Host load
- Caching/repetition variation
- Background traffic

Factors influencing traffic and potential microstructures

There are many!   For example:

- Application/task
- **Implementation and version of**
- Network congestion
- Host load
- Caching/repetition variation
- Background traffic



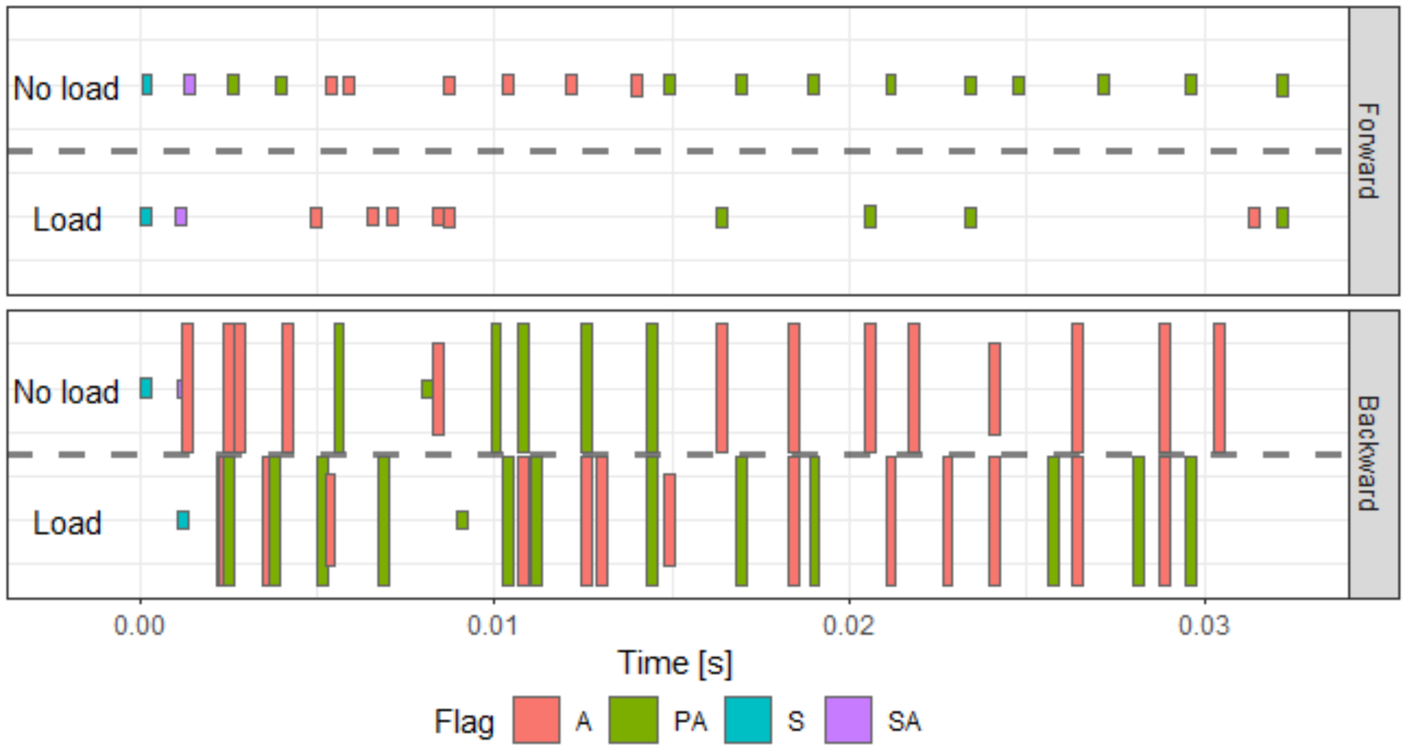
From Marx et al, ***Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity, EPIQ* 2020.**

Factors influencing traffic and potential microstructures

There are many! For example:

- Application/task
- Implementation and version of
- Network congestion
- **Host load**
- Caching/repetition variation
- Background traffic

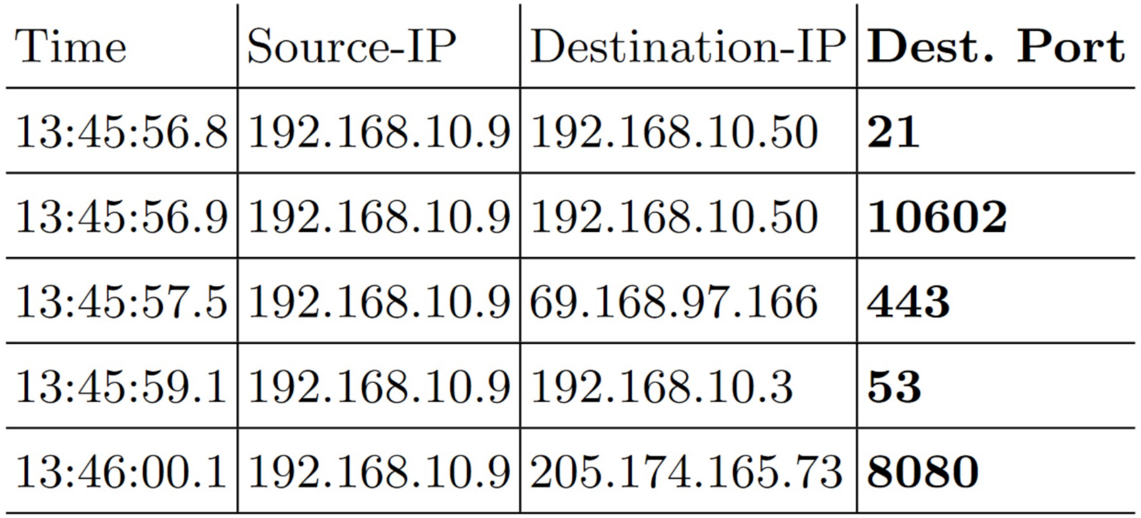
FTP-connection comparison under load



Factors influencing traffic and potential microstructures

There are many!   For example:

- Application/task
- Implementation and version of
- Network congestion
- Host load
- Caching/repetition variation
- **Background traffic**

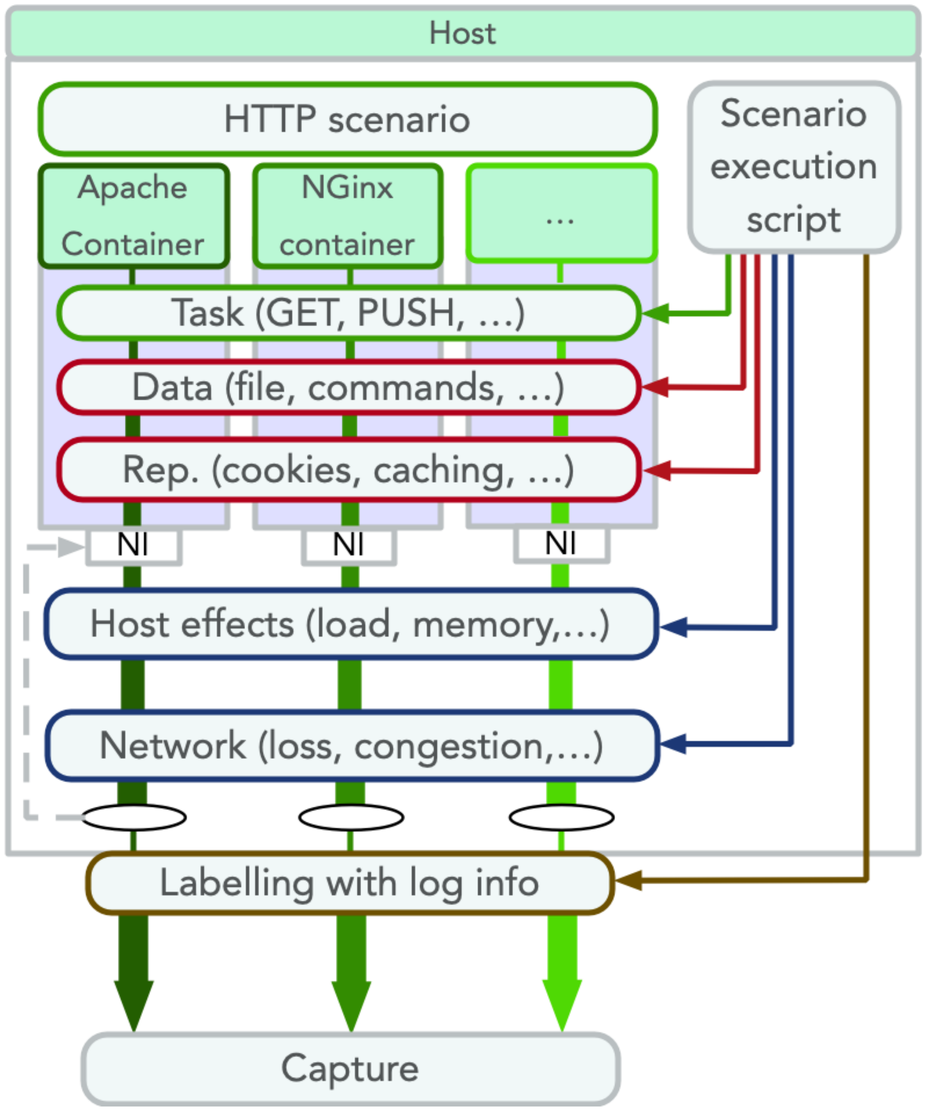


Time	Source-IP	Destination-IP	**Dest. Port**
13:45:56.8	192.168.10.9	192.168.10.50	**21**
13:45:56.9	192.168.10.9	192.168.10.50	**10602**
13:45:57.5	192.168.10.9	69.168.97.166	**443**
13:45:59.1	192.168.10.9	192.168.10.3	**53**
13:46:00.1	192.168.10.9	205.174.165.73	**8080**

DetGen: Controlling variations

To handle data variation, we extend DetGen:

- simulate external factors
- parameterise scripts
- randomise at every stage

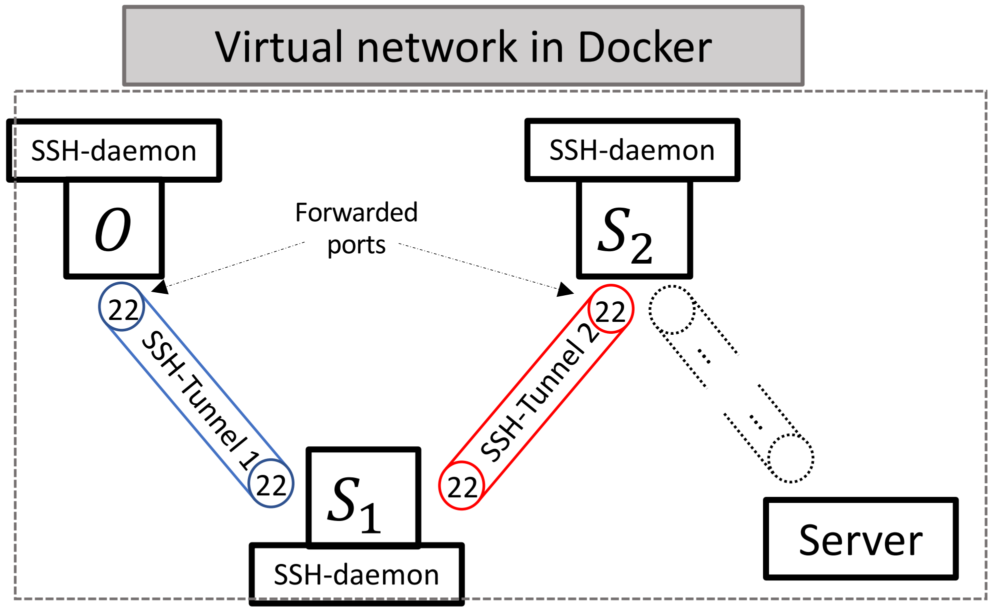

Host
HTTP scenario
Scenario execution script
Apache Container
NGinx container
...
Task (GET, PUSH, ...)
Data (file, commands, ...)
Rep. (cookies, caching, ...)
NI
NI
NI
Host effects (load, memory,...)
Network (loss, congestion,...)
Labelling with log info
Capture

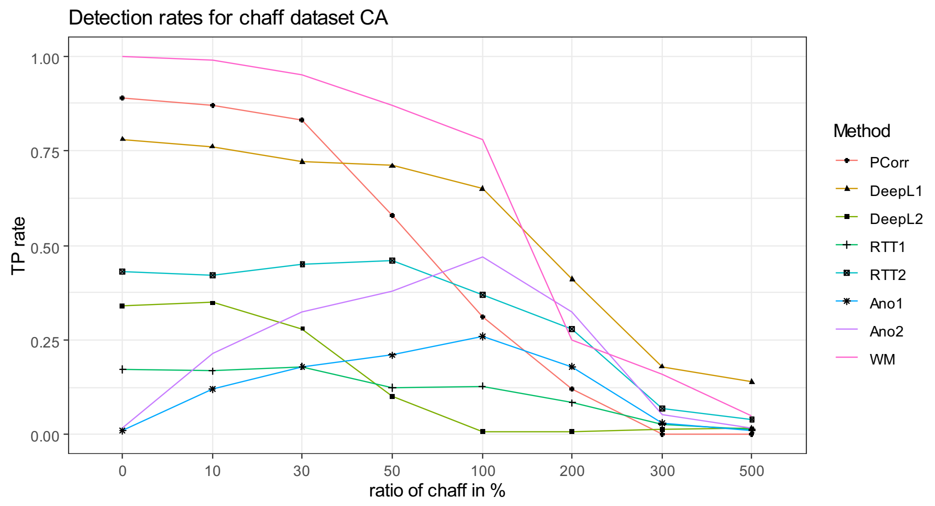
Application: Evaluating stepping-stone detection methods

Problem: no public datasets, existing work used own datasets, incomparable and overly simple.

Solution: generate independent dataset with Detgen, injecting chaff and jitter (Netcat, NetEm).

Compare competing methods on same basis.

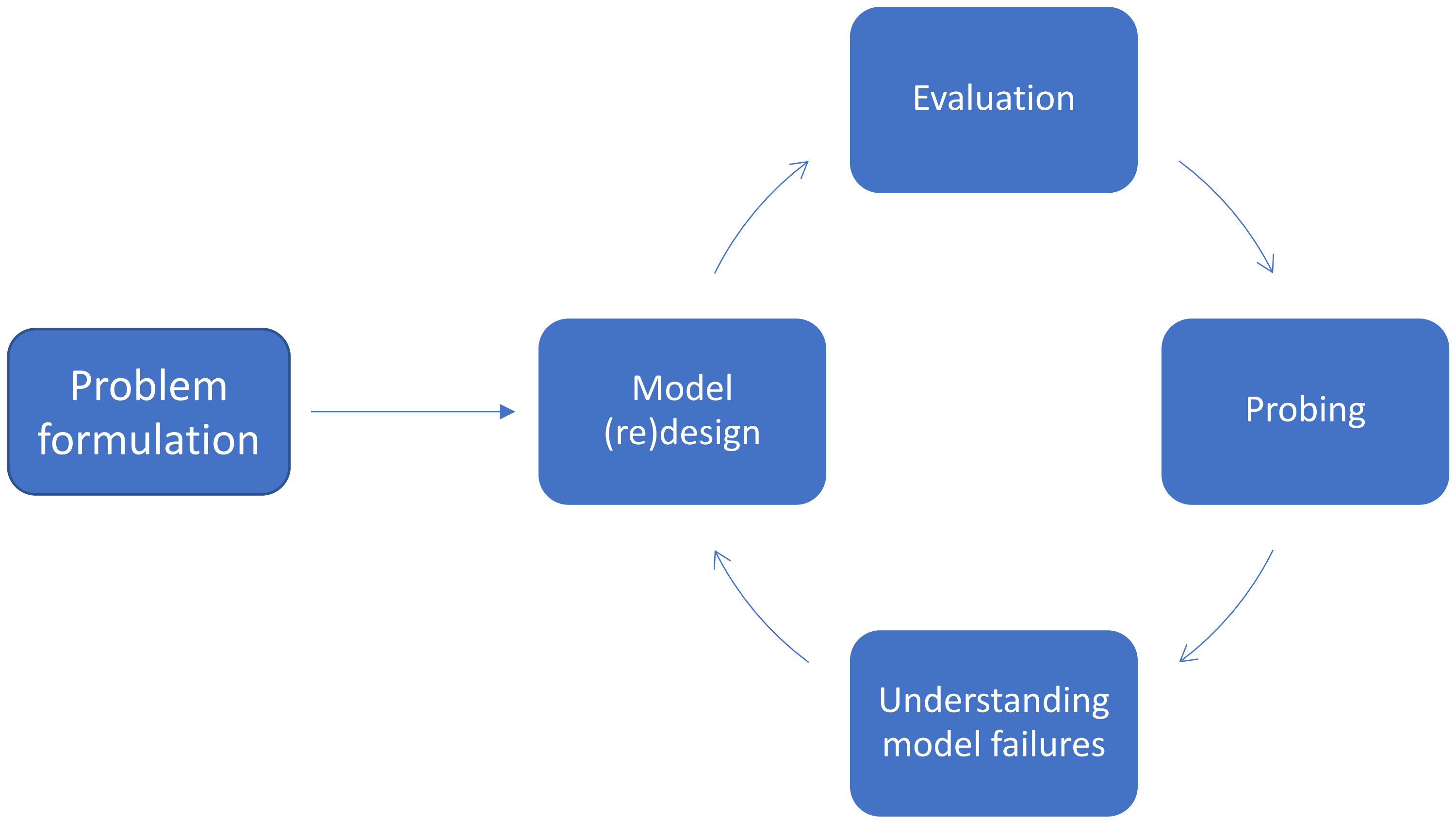




Challenge 3:

Understanding Model Performance

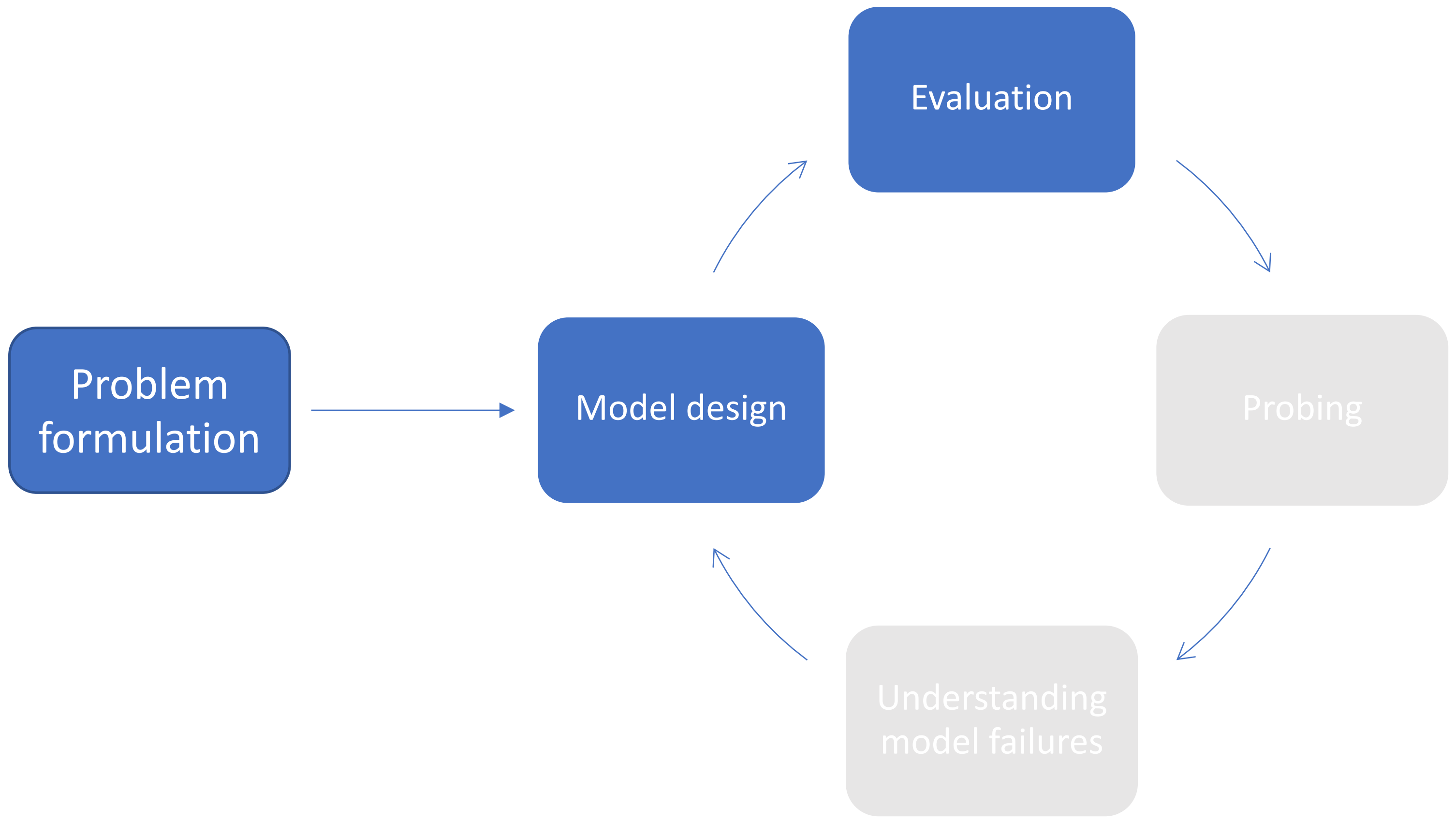
Machine Learning model development process



Famous advances

- Ambiguity in translation
  - Attention layer
- CNNs biased to texture
  - Image stylization
- Video enhancement
  - Multi-scale encoders

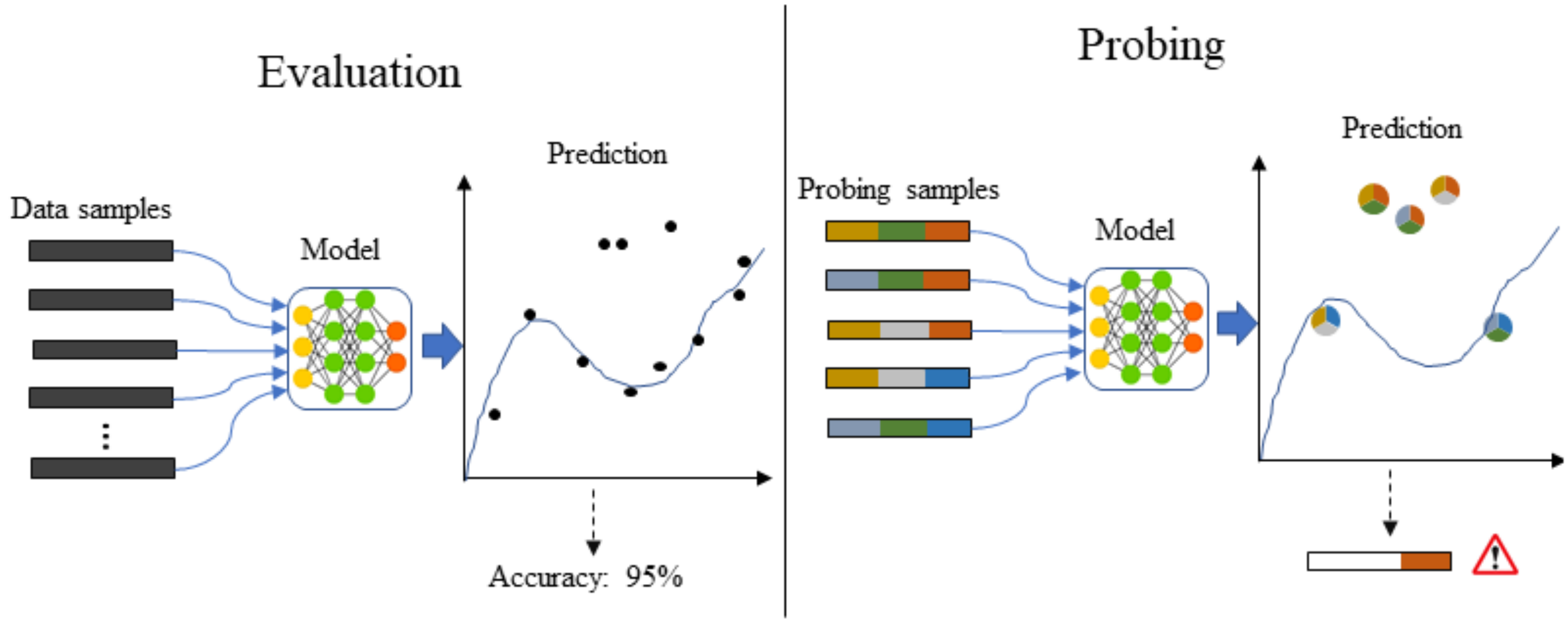
Machine Learning model development process in NID



Existing NID datasets are:

- Difficult to read/interpret
- Sparsely/poorly labelled
- Non-malleable

Model evaluation vs probing

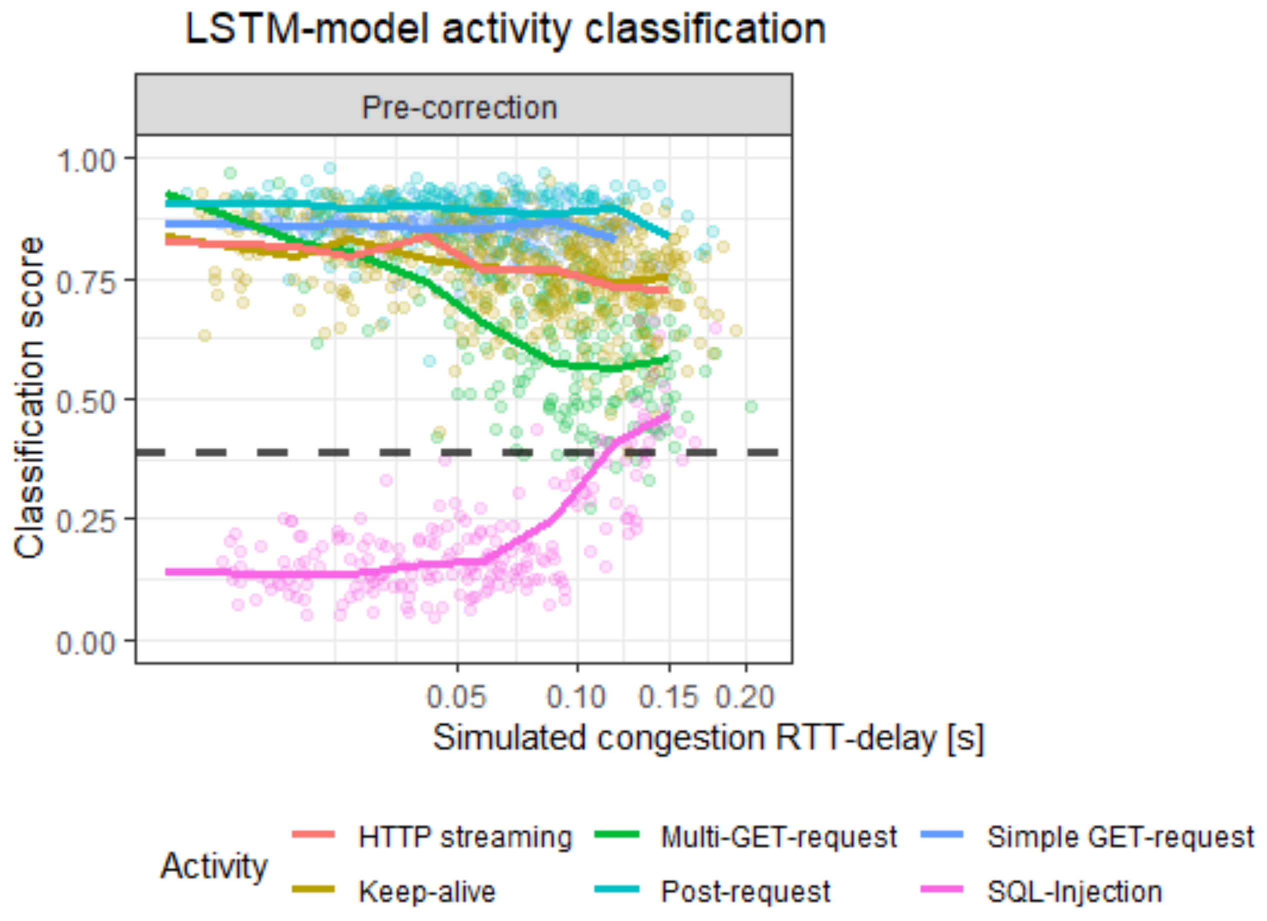


Probing example: SQLi attacks hidden in congestion

Take a state-of-the-art LSTM packet stream classifier (Hwang et al, 2019).

Train on original dataset and similar Detgen labelled traffic.

Probing the model with randomized labelled traffic shows that it makes mistakes under higher congestion!

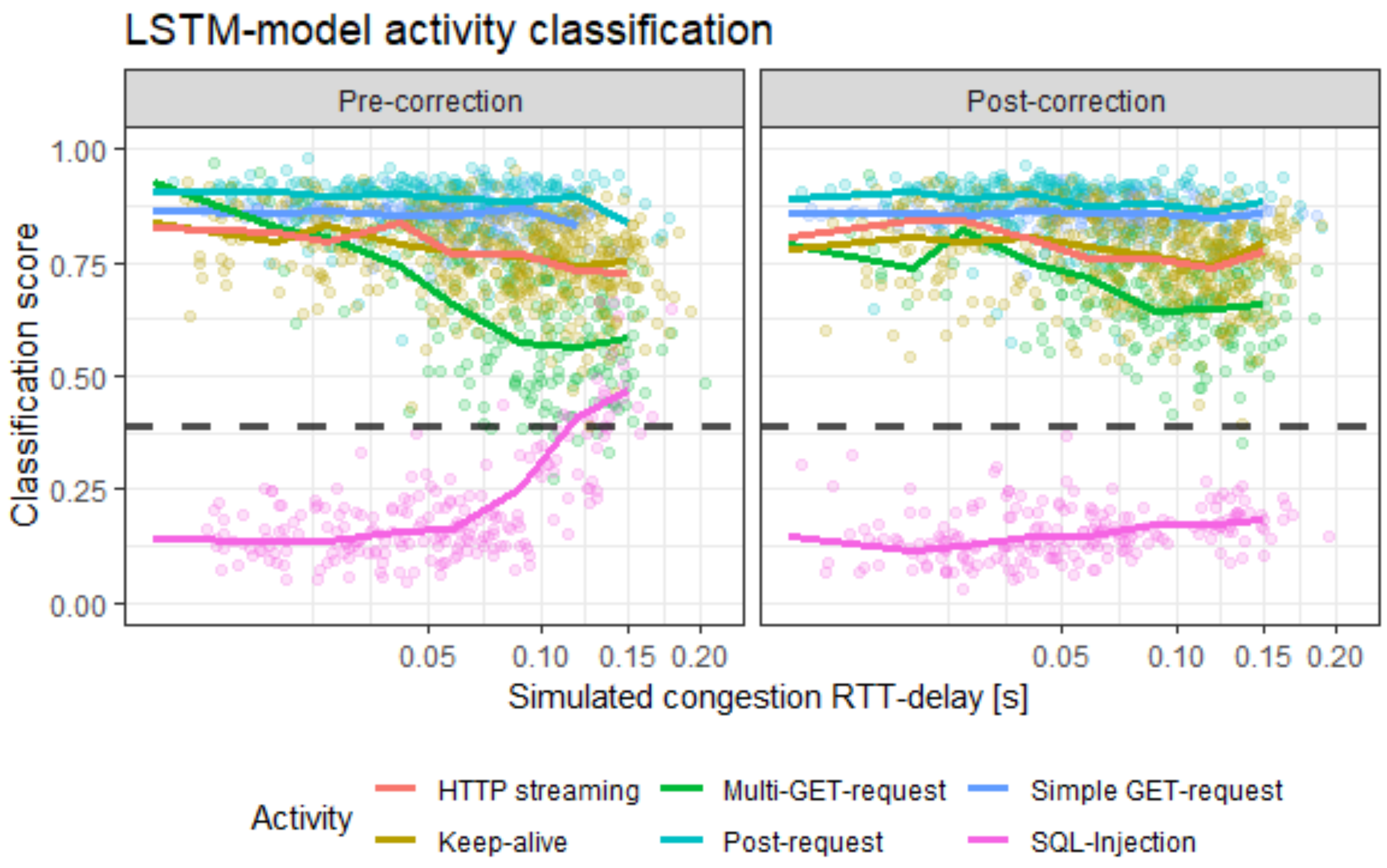


Understanding and repairing the error

Test hypothesis: generate two identical SQLi connections, one with high latency.

Second causes retransmissions and model misclassification!

Pre-processing the data improves model.



Outlook and Future Work

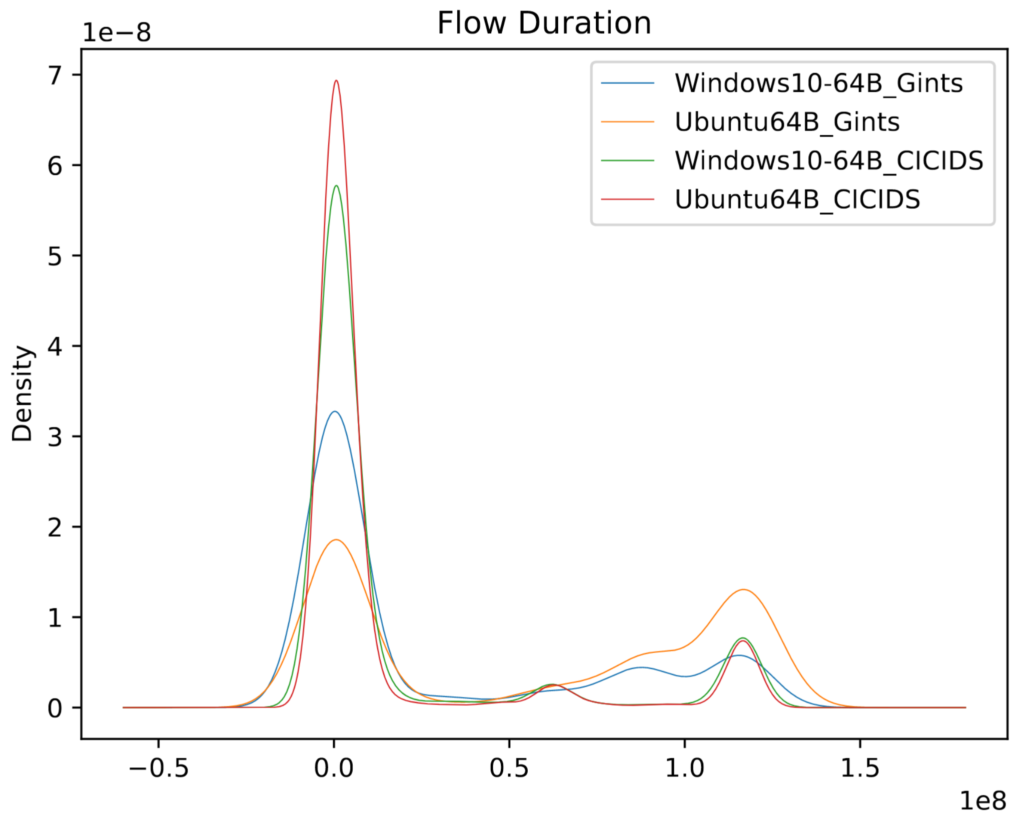
Revisiting NIDS datasets: data science forensics

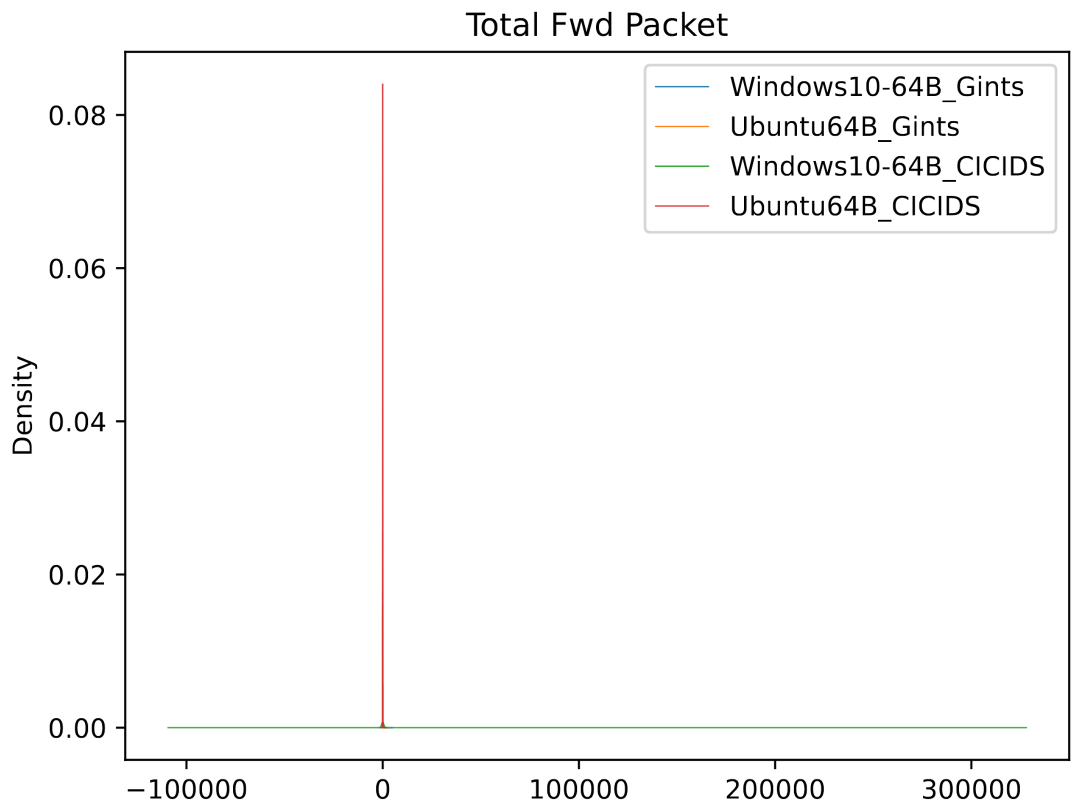
Examining some popular existing datasets in detail we find:

- Simulation artifacts cause short-cut learning e.g., OS-specific TTL values
- Whole classes of failed attacks, simplify classification massively
- Low overlap between malicious and benign traffic flow statistics
- Heavy reliance on pen-testing tools results in narrowly distributed features – unrealistic baselines for real-world attacks

These datasets have been used for dozens (sometimes hundreds) of research papers.

Example: comparing synthetic with real traffic





Need for anonymisation

Synthetic data is invaluable, but **real-world open datasets** are essential

In recent work with the Alan Turing Institute we have begun investigating the design and effectiveness of anonymisation mechanisms:

- **organisational privacy and PII**: use of anonymisation functions or PETs
- **utility for data analysis**: retaining ability to train classifiers or trigger rules

Data-driven vulnerabilities: adversarial examples and more

Intentional failures		Unintentional failures
Perturbation attack	Physical adversarial examples	Reward hacking
Poisoning attack	Training data recovery	RL environment side effects
Model inversion	Model supply chain attack	Concept drift/shift
Membership inference	Trojaned model	Natural adversarial examples
Model stealing	Software exploit confusion	Common corruption
Reprogramming		Incomplete testing

See Microsoft's methods for ***Threat Modeling AI/ML Systems and Dependencies*** at https://docs.microsoft.com/en-us/security/failure-modes-in-machine-learning

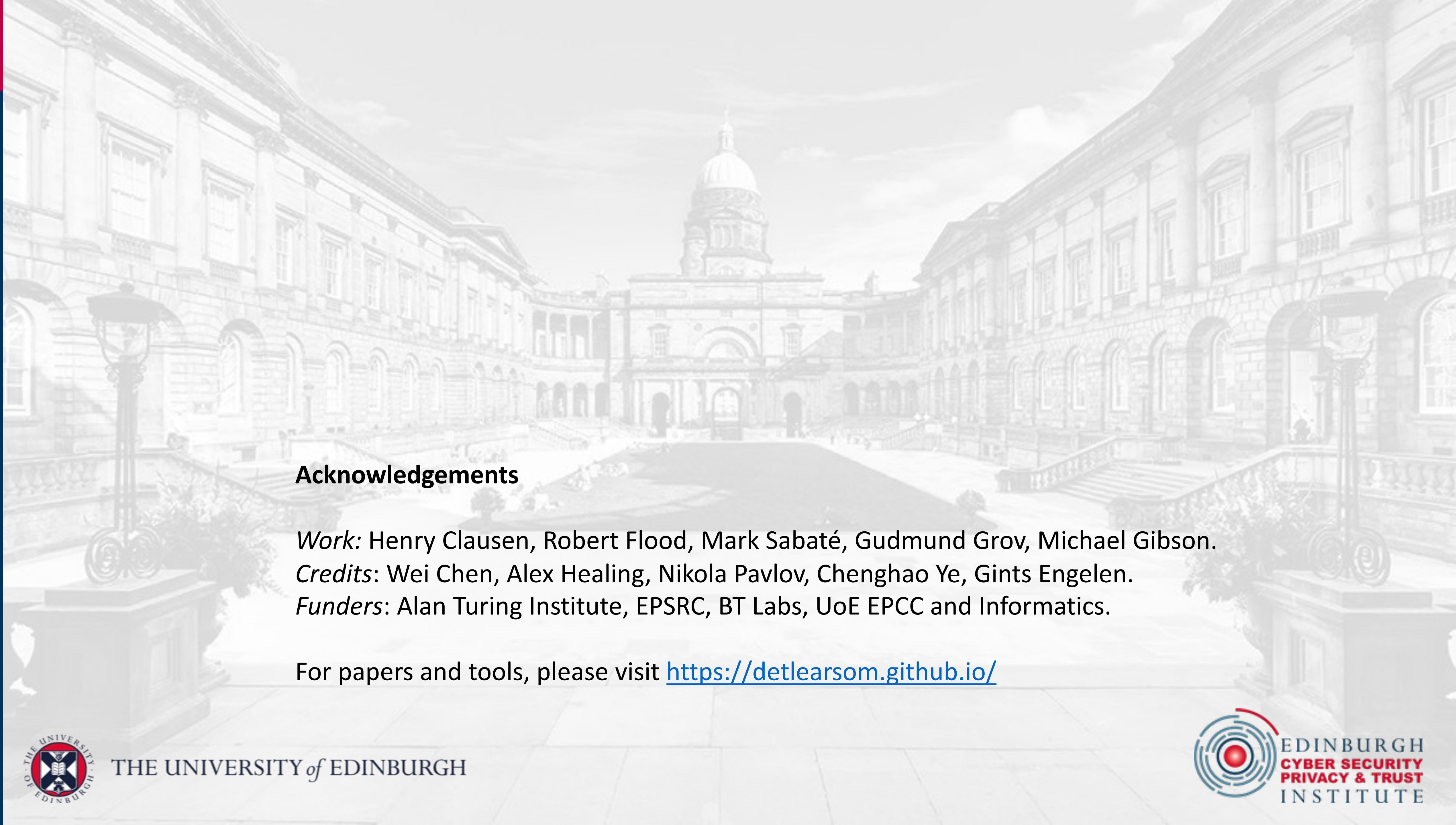
Summary

We set out to construct “software models” from outside the software, using data from network captures. Existing datasets were not precise enough, inspiring **DetGen.**

Having a precise synthetic data tool supported new NIDS contributions:

- better models for detecting small-signal **access attacks**
- new datasets with ground truth to **defeat stepping stone detection**
- ways to probe state-of-the-art model failures to **improve attack detection**

For papers and tools, please visit https://detlearsom.github.io/



Acknowledgements

Work: Henry Clausen, Robert Flood, Mark Sabaté, Gudmund Grov, Michael Gibson.
Credits: Wei Chen, Alex Healing, Nikola Pavlov, Chenghao Ye, Gints Engelen.
Funders: Alan Turing Institute, EPSRC, BT Labs, UoE EPCC and Informatics.

For papers and tools, please visit https://detlearsom.github.io/

THE UNIVERSITY of EDINBURGH

EDINBURGH CYBER SECURITY PRIVACY & TRUST INSTITUTE