User behaviour analysis for Malware detection

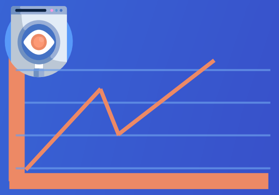
Valentina Dumitrasc & **René Serral**

Start

Part I
Introduction

Part II
Architecture

Part III
Model &
Scenario



Part IV
Results

Part IV
Next Steps/
Conclusions

Introduction

Part I

Problem identification

Increase in cyberattacks

Malware attacks increased 358% compared to 2019

68% of the companies experienced a targeted attack on their networks

Current techniques

Current techniques for malware detection are limited

Current techniques

Dynamic analysis

- Detected by malware
- Change behaviour
- Computer resources needed
- Not immediate
- Not detect until executed
- No specific solution for servers
- Malware-centric

Static analysis

- Malware signatures change
- Packed/Obfuscated
- Can’t detect zero days
- No specific solution for servers
- Malware-centric

Justification

User-centric

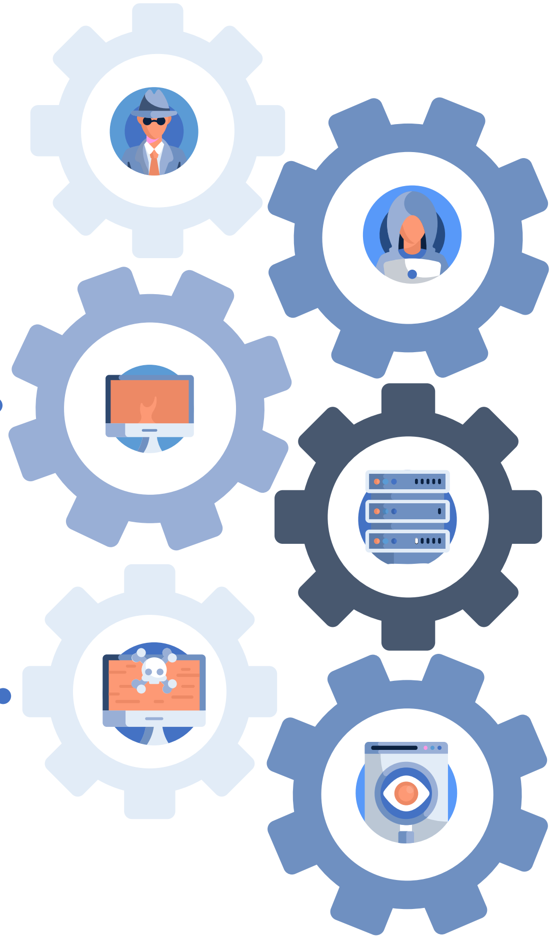
Detect deviation from normal user behaviour

Efficient

Does not involve a dedicated infrastructure for malware analysis

Quick detection

Metrics extracted every few seconds



Zero-days

Detect any deviation

Servers

Adapt to any host

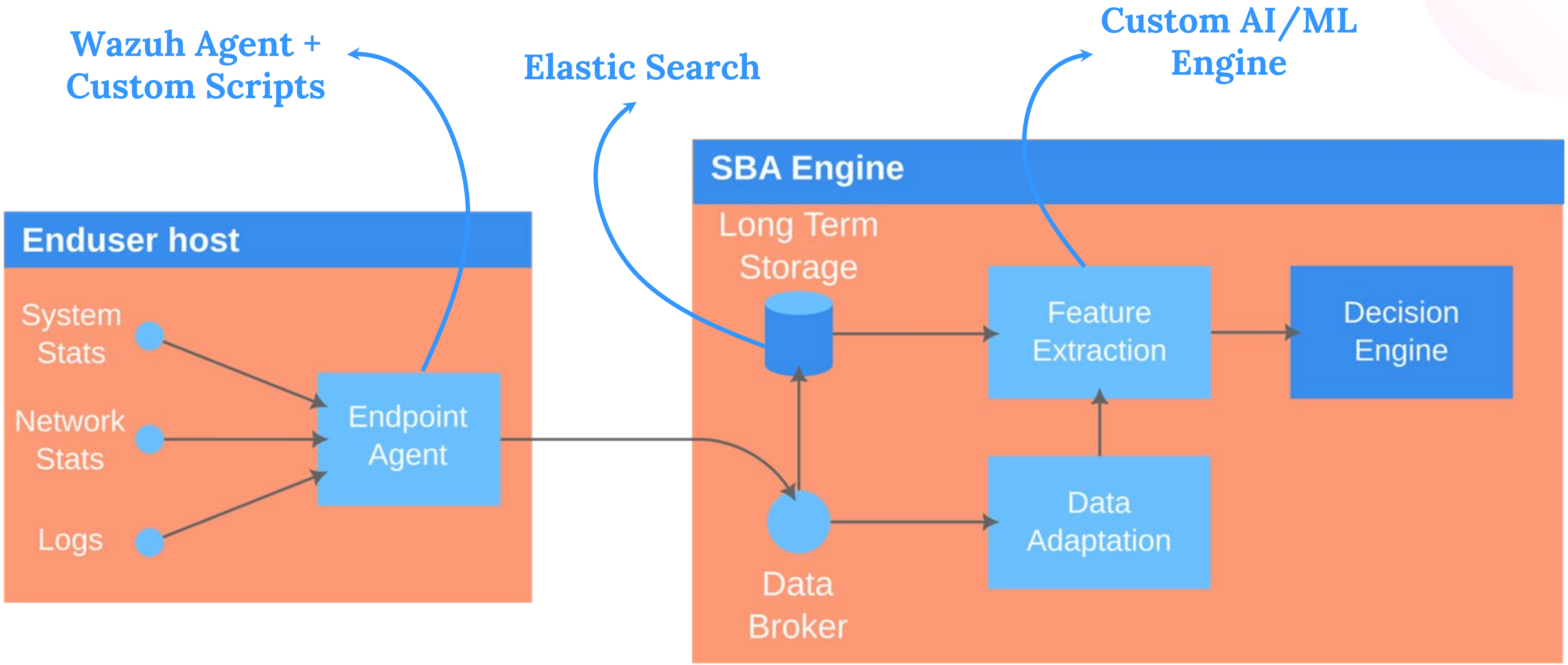
Lightweight

Models are quickly trained + fast prediction

Architecture

Part II

Architecture



Decisions

Machine learning

Relationships among features

Can handle noise

Adapt to changes

No supervised training

We propose a system able to adapt without previous supervised learning process

Model & Scenario

Part III

Metrics

Network

- Geographical connection origin
- Data Traffic Ratio
- Data Transfer Rate
- Packet Ratio
- Packet Transfer Rate
- Connection Status/Count

26

Processes

- Process count

3

System audit

- File Operations
- Command ratio

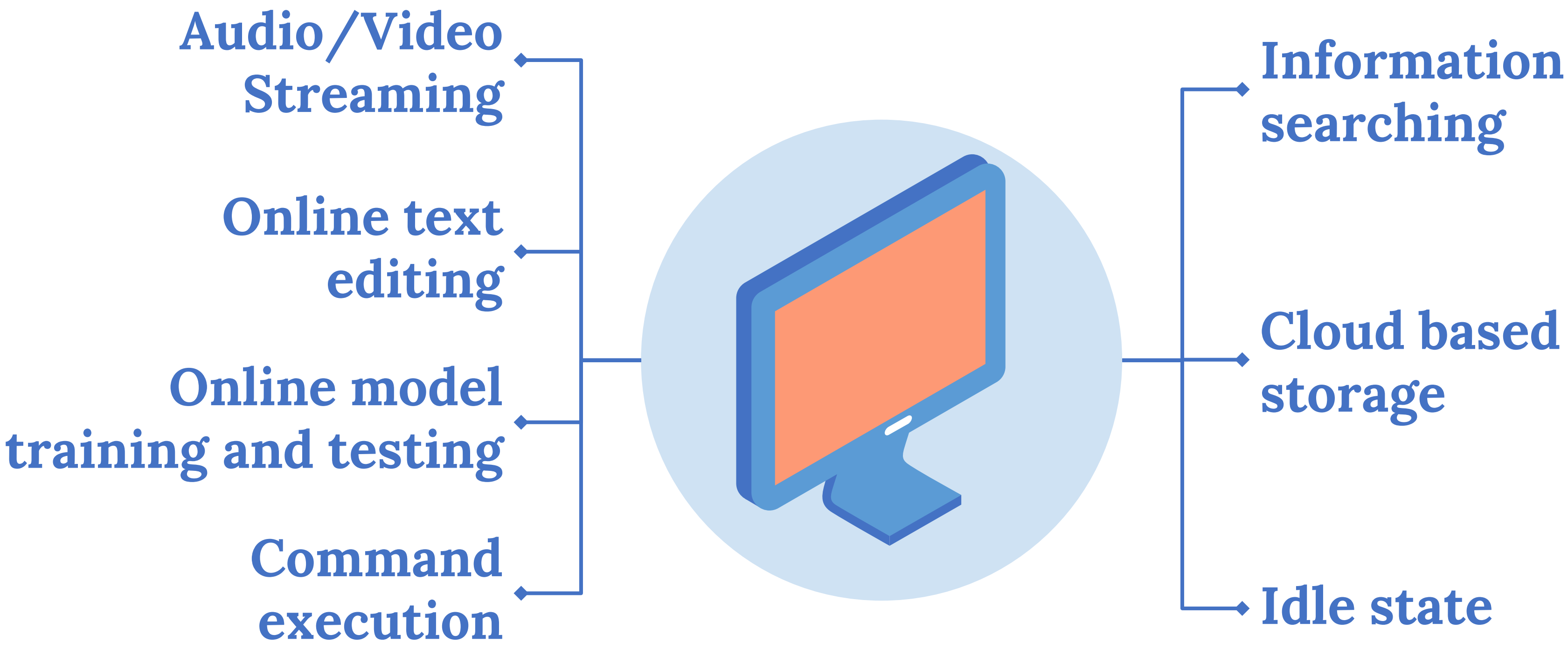
5

CPU/Memory

- Cpu usage
- Memory usage

15

Normal behaviour data



Test data

Ransomware

Denial of Service

Botnet

Backup

Software compilation

Normal behaviour

Algorithms

~~One-Class SVM~~

~~Local Outlier Factor~~

Kernel Density Estimation

Autoencoder

Kernel Density Estimation

Parameters

kernel →Gaussian
bandwidth →Scott
threshold →7th prc.

Issues

Wrong feature importance

Issues detecting botnet and ransomware

Autoencoders

Layers

input→Shape of the training data
encoding→49 n, ReLU activation, L1 regularization
decoding→49 n ReLU
output→sigmoid function

Issues

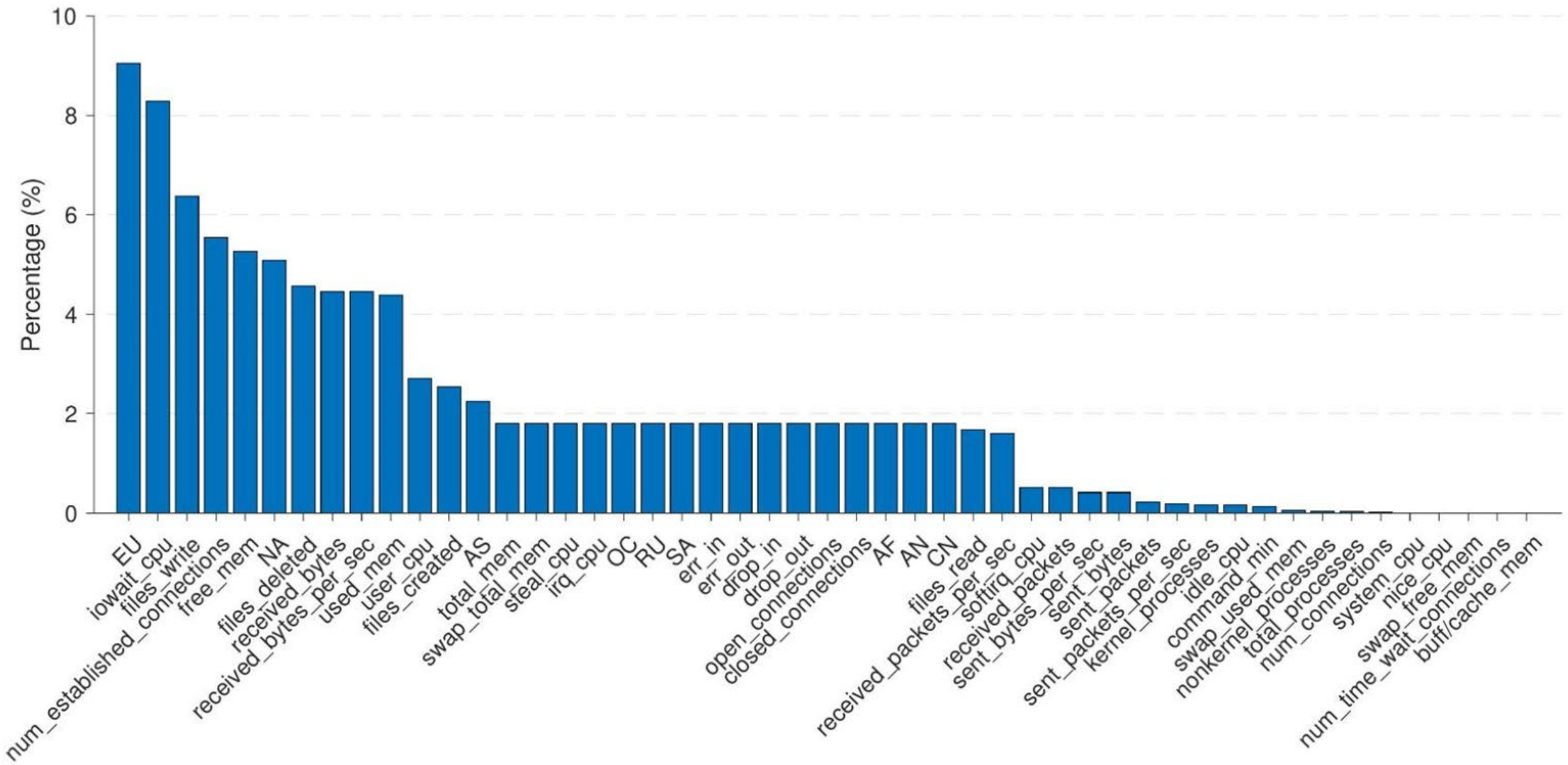
Poor feature importance

Issues detecting ransomware

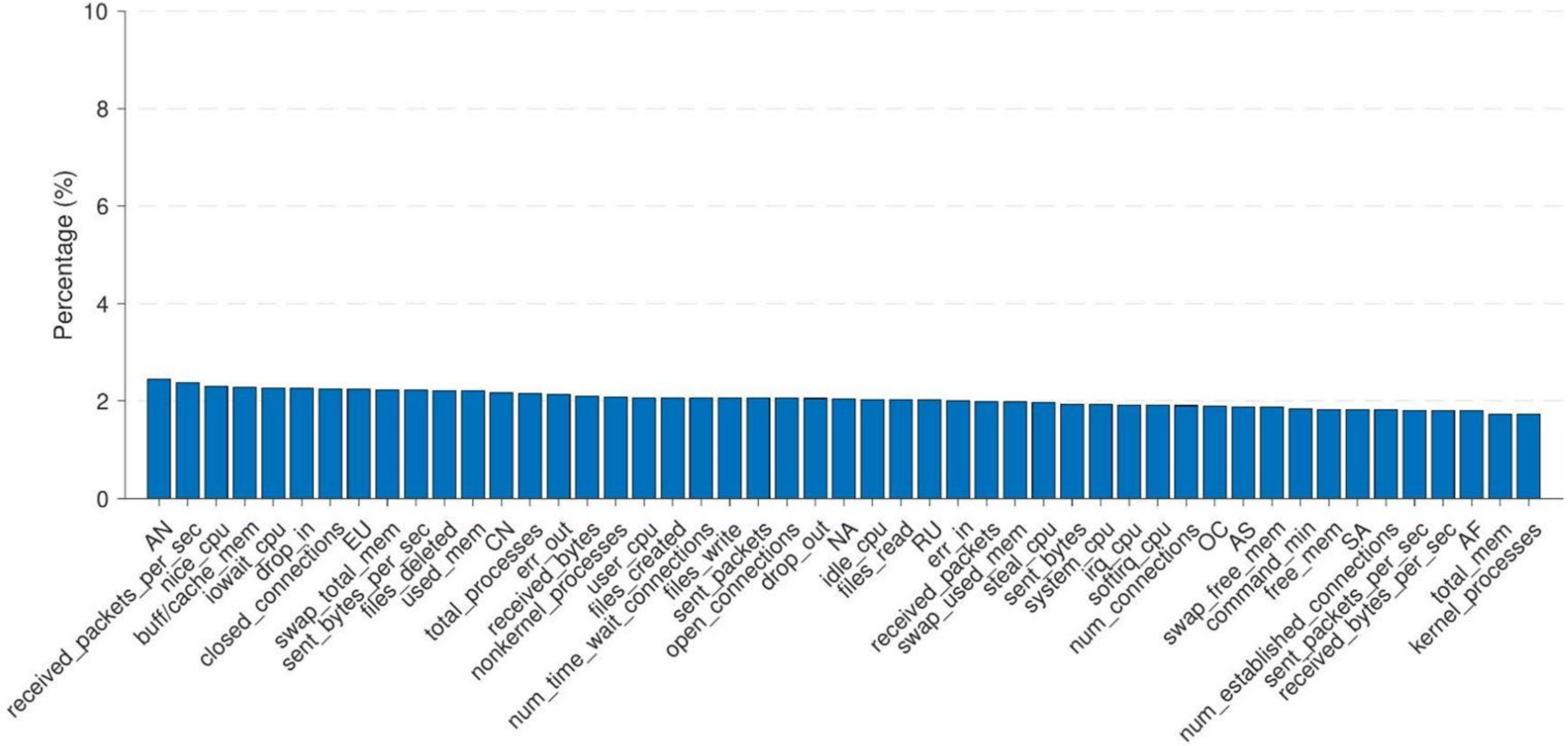
Results

Part IV

Kernel Density Estimation



Autoencoder



Comparison

	One-Class SVM	LOF	Autoencoder	KDE
Normal behaviour	0.9	0.92	1	1
Botnet	1	1	1	0.9
DoS	1	0.96	1	1
Ransomware	1	0.5	0.71	0.42

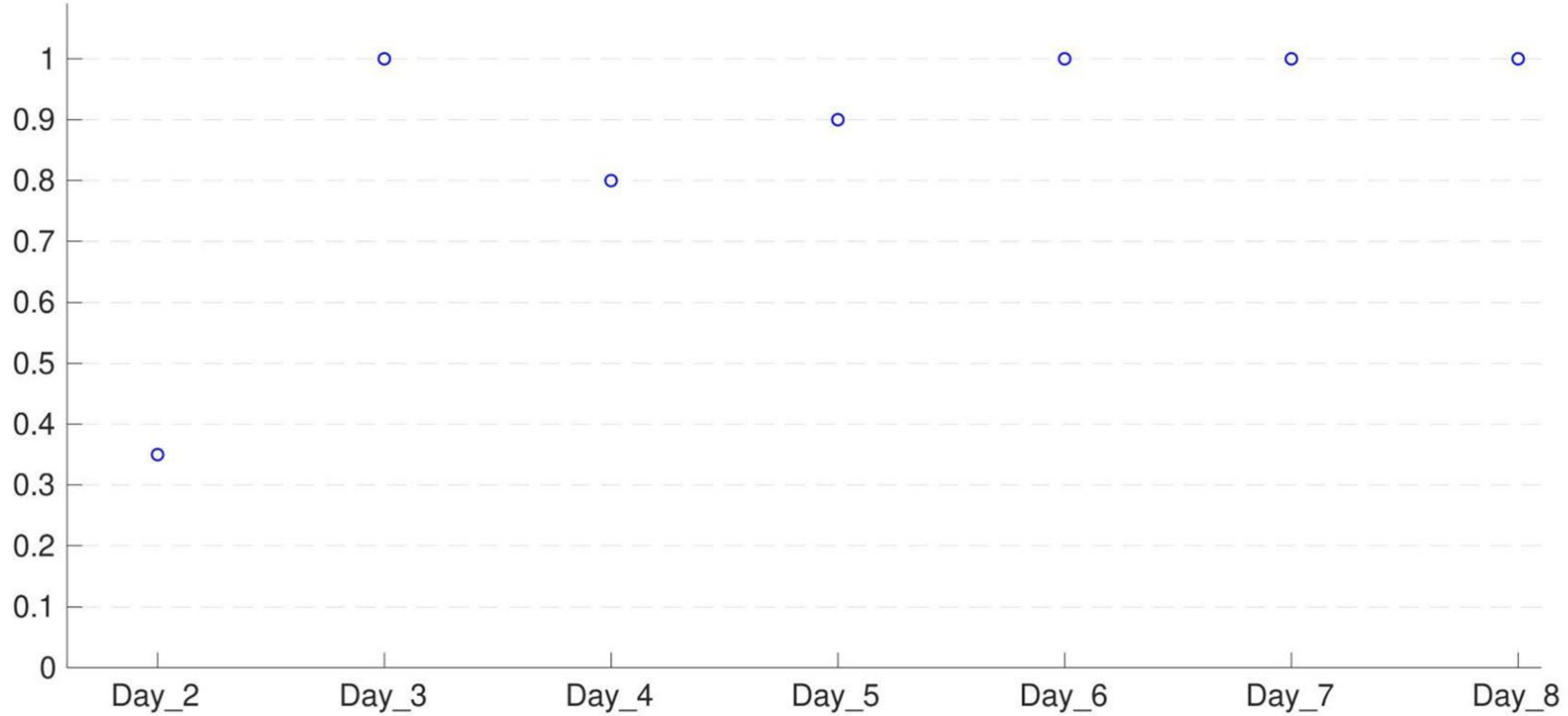
Comparison

Further training the system:

Introducing software compilation and backup data

	One-Class SVM	LOF	Autoencoder	KDE
Normal behaviour	0.81	0.95	1	1
Botnet	1	1	1	1
DoS	1	0.96	1	1
Ransomware	0.93	0.28	0.78	0.35

False positives



Next Steps / Conclusions

Part V

Next Steps

- Cloud
- Real life detection/ mitigation
- Preprocessing
- Test/Incorporate into a real system
- Different operating systems

THANKS

QUESTIONS?