

IMPERIAL

IMPERIAL COLLEGE LONDON
DEPARTMENT OF COMPUTING

Anomaly Detection for IoT Environments

Eman Maali

December 18, 2024

Submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in
Computing of Imperial College London and the Diploma of Imperial College London.

Declaration of Originality

I, Eman Maali, certify that all material in this thesis, which is not my work, has been appropriately acknowledged.

By the conventions of my native language, Arabic, where "we" is a formal address. I have used the pronoun 'we' throughout the thesis instead of 'I'. However, I led all the work presented here and am the primary contributor.

Copyright

The copyright of this Thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-NonCommercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for use of this work that is not included in this licence or permitted under UK Copyright Law.

Abstract

The dramatic increase in globally deployed insecure IoT devices has exposed the internet to various cyberattacks. Botnets such as Mirai exploit vulnerable IoT devices for distributed denial-of-service (DDoS) attacks on critical infrastructure. Network operators seek to limit the impact of these attacks by using anomaly detection solutions. However, these anomaly detection solutions are not tailored for IoT environment characteristics (*i.e.*, diverse IoT behaviours), thus leading to an increase in false alarms. In this Thesis, we address the urgent need to develop effective methods for detecting malicious IoT traffic. Through systematic exploration, we have demonstrated that accurate and robust device identification is fundamental to enhancing anomaly detection in IoT environments. We explore three key objectives. First, we comprehensively evaluate existing machine learning approaches for IoT device identification, highlighting their limitations and areas for improvement. Second, we develop a model that combines three identification models: device behavioural profiles, fingerprints, and functional blocklists. This combination enhances robustness and reduces false alarm rates in traffic analysis. Third, we design an unsupervised model using Symbolic Aggregate Approximation (SAX) and compression-based dissimilarity to enable real-time detection of malicious IoT traffic, ensuring the identification of evolving threats at the network's edge. Through these objectives, this Thesis advances IoT security by enhancing device identification accuracy and enables more reliable detection of anomalous and malicious activity.

Acknowledgments

I want to express my sincere gratitude to the Schlumberger Foundation for their generous sponsorship, which has made this scholarship possible. Their unwavering support has been invaluable in advancing my academic and research pursuits. This opportunity has provided the financial means to further my studies and motivated me to continue pushing the boundaries of knowledge in my field. Thank you for your confidence and commitment to fostering the growth of future leaders.

I also want to express my heartfelt gratitude to my supervisor, Prof. Julie McCann, for her invaluable expertise, constructive feedback, and unwavering support, which have profoundly shaped this journey.

Dedication

To my mom, for your countless sacrifices, unwavering love, and the constant belief you have always had in my dreams, even when the path seemed unclear. You have been my rock, giving me the strength to keep going, even during the toughest moments. Your wisdom, kindness, and unshakeable faith in me have shaped who I am today. No words can fully express my gratitude for the endless support you've given me throughout this journey. You have been my guiding light, and I carry your love with me in every step I take.

To my grandfather, whose unwavering faith in me has strengthened me throughout this journey. Even though you are no longer with us, your belief in my potential continues to guide and inspire me. Your wisdom, patience, and shared lessons have shaped who I am today. I will forever cherish the love and encouragement you gave me, and I carry your memory with me every step of the way. Thank you for being a constant source of support, even in spirit.

To my mentor, Omar Alrawi, a second supervisor, thank you for your invaluable patience, constructive feedback, and the countless long discussions that helped shape this research. Your ability to bounce ideas back and forth, challenge my thinking, and guide me through complex moments has been a key part of this journey. I am deeply grateful for your mentorship and the insight you've provided every step of the way.

To Lorraine, thank you for your unwavering support, boundless patience, and the peaceful walks through Hyde Park, which have always helped me reset. I am genuinely speechless when it comes to your kindness and support. The day you stepped into this journey was a game changer. Every time I look back, I am profoundly grateful for having you by my side, especially when I was about to quit. Your belief in me and gentle reminders of my capabilities kept me going. You were my saving angel, and I will always be thankful.

To my friends Maryam, Zainb, Weam, and Rana, thank you for your incredible patience and understanding through limited communication. Thank you for always being welcoming and supportive, offering encouragement when I needed it most, and being my sanctuary during the toughest moments of this journey. Your friendship has been a steadfast anchor throughout.

To Claire, for the countless moments of joy and unwavering support and for reminding me to celebrate even the smallest victories. Thank you for our food journeys and the warmth of your friendship, which made this journey even more meaningful.

To Ali and Myles for standing by me when I needed it the most, especially when no one else was there. Your unwavering support and friendship gave me the strength to keep going. Thank you for being my rock during the toughest times—I am forever grateful.

To Michael, for your wisdom, invaluable advice, and the laughter that brightened my journey. Thank you for being a father figure when I needed it most and for your unwavering kindness and support. Your presence in the last few years has made an immeasurable difference in my journey.

To my lab colleagues and friends at Imperial, thank you for being there every step of the way. Your support and shared dedication made this journey so much more meaningful. Whether through collaborative discussions, shared challenges, or moments of laughter, your presence has been invaluable. I am deeply grateful for the friendships and experiences we have built together.

To those who were there at the beginning of this journey, thank you for your presence, guidance, and support. Though you may not be here to witness the end, your impact remains a part of this achievement.

Thank you to those who gave me a tough time during this journey. With your challenges, I discovered my strength, my stars, and the true extent of what I can achieve.

To all women in STEM who are working under hostility and facing countless obstacles yet continue to push forward with resilience and determination. Your strength, perseverance, and commitment to your passions inspire us all. This journey is yours as much as it is mine.

And finally, to my country, Palestine, where I hope one day we will achieve liberty and justice. May our resilience and hope lead us to a future of peace and freedom.

Quote

“The more you know, the more you know you don’t know.”

— Aristotle

Contents

1	Introduction	8
1.1	Motivation, Research Question and Objectives	8
1.2	Contributions	10
1.3	Thesis Organisation	12
1.4	List of Publications	13
2	Background and Preliminaries	14
2.1	Introduction	14
2.2	Preliminary Analysis	14
2.2.1	Consumer IoT Devices	15
2.2.2	Types of Network Attacks	16
2.2.3	Anomaly Definition	18
2.2.4	Types of Traffic Anomalies	18
2.2.5	Mapping of Network Attacks with Anomalies	19
2.2.6	Network Data Types	20
2.3	Anomaly Detection Solutions in IoT Environments	21
2.3.1	Anomaly Detection Literature Review	21
2.3.2	Challenges in Anomaly Detection in IoT Environments	24
2.4	Conclusion	26
3	Evaluating Machine Learning-Based IoT Device Identification Models for Security Applications	28
3.1	Introduction	28
3.2	Background	29
3.2.1	Literature Review	29
3.2.2	Evaluating ML-based IoT Identification	31
3.3	Evaluation Method	32
3.3.1	Practicality Attributes	33
3.3.2	Explainability Analysis	34
3.3.3	Statistical Analysis	35
3.4	Results	35
3.4.1	Experimental Setup	35
3.4.2	Dataset Collection and Labeling	37

3.4.3	Evaluation Results	40
3.4.4	Explainability Results	46
3.4.5	Feature Selection Improvement Results	52
3.5	Discussion	54
3.5.1	Takeaways and Lessons	54
3.5.2	Limitations	55
3.6	Conclusion	56
4	A Hybrid Model for IoT Device Traffic Identification on the Home Gateway	57
4.1	Introduction	57
4.2	Background	58
4.2.1	IoT Fingerprinting	59
4.2.2	IoT Behavioural Profiles	61
4.2.3	Blocklisting Profiles	62
4.3	Hybrid Model Architecture	62
4.3.1	Hybrid Model Overview	63
4.3.2	Baseline Phase	64
4.3.3	Pre-processing Phase	64
4.3.4	Identification Phase	65
4.4	Results	69
4.4.1	Experimental Setup	69
4.4.2	Dataset Collection and Labelling	70
4.4.3	Fingerprinting Model Results	70
4.4.4	Traffic Identification and Validation Results	72
4.4.5	Ablation Study Results	73
4.5	Discussion	74
4.5.1	Takeaways and Lessons	74
4.5.2	Limitations	75
4.6	Conclusion	75
5	iDetector an Anomaly Detection for IoT Environments on Edge Devices	76
5.1	Introduction	76
5.2	Background	77
5.3	Threat Model	80
5.4	iDetector Design	80
5.4.1	Design Goals	80
5.4.2	Design Overview	81
5.4.3	Design Details	82
5.4.4	Theoretical Analysis	87
5.5	Results	92
5.5.1	Experimental Setup	93
5.5.2	Data Collection and Labelling	94
5.5.3	Detection Results	96

5.6	Discussion	102
5.6.1	Takeaways and Lessons	102
5.6.2	Limitations	102
5.7	Conclusion	103
6	Conclusion	104
6.1	Summary of Thesis Achievements	104
6.2	Security Applications	105
6.3	Limitations	106
6.4	Future Work	108
6.5	Final Thoughts on Anomaly Detection in IoT Environments	109
	Bibliography	111

List of Figures

3.1	The UK and USA testbed devices' number of flows per day between 5 th – 27 th December, and the devices' mode of operation was idle.	38
3.2	t-SNE analysis of IoT devices from the UK testbed, from 6 th to the 27 th of December 2022. Devices' modes of operation were mixed.	47
3.3	SHAP of IoT devices from the UK testbed, from 6 th to the 27 th of December 2022. Devices' modes of operation were mixed.	48
3.4	Correlation matrix for uncorrelated features used in IoT device identification solutions.	53
4.1	High-Level architecture diagram for iHybrid traffic identification.	64
4.2	Percentage of traffic under detection comparing different setup: MUD, IoTrim, MUD + IoTrim, and iHybrid.	74
5.1	High-level design of iDetector full cycle as implemented for a single device.	81
5.2	Empirical entropy difference ΔH for different SAX parameters in TP-Link Smart Plug.	93
5.3	Experimental setup to examine malicious events in our testbed.	95

List of Tables

2.1	Anomaly categorisation overview with examples [99, 26, 44, 80].	19
2.2	Comparison of network traffic analysis tools and protocols [99, 26, 44, 80]. . .	20
3.1	Qualitative comparison for selected IoT device identification current literature. Each column corresponds to a discussion of the comparison aspect of related work.	30
3.2	List of features used in each experimental setup under evaluation.	36
3.3	A sample activities of IoT devices performed during active experiments in the UK testbed.	39
3.4	IoT devices under test.	40
3.5	Reproducibility and mode of operation evaluation for learning-based IoT identification solutions.	41
3.6	Temporal transferability detailed AUCPR evaluation for learning-based IoT identification solutions.	42
3.7	Spatial transferability detailed AUCPR evaluation for learning-based IoT identification solutions.	44
3.8	Observability detailed AUCPR evaluation for learning-based IoT identification solutions.	45
3.9	Improvement evaluation results AUCPR for IoT identifications tailored features, under device mode of operation and transferability metrics.	54
4.1	Detection performance of device model recognition phase, binary classification represents IoT vs non-IoT classification, while multiclassification represents the device mode recognition in IoT devices.	71
4.2	Performance comparison between our fingerprinting and the IoT identification state of the art.	72
4.3	Ablation study comparing different setup: MUD, IoTrim, MUD + IoTrim, and iHybrid.	73
5.1	The qualitative comparison for the most recent malicious IoT traffic detection. Each column corresponds to a discussion of the comparison aspect of related work.	79
5.2	List of features extracted from flows, and minimum list of features used in detection.	83

5.3	Testbed and dataset configuration and statistics used to evaluate iDetector. We use "/" when the dataset is benign only traffic.	93
5.4	Detection rate of iDetector for different malicious activities conducted by our experiments and public datasets.	97
5.5	Comparison analysis of iDetector and HorusEye across different datasets detection rate performance.	97
5.6	Overhead Analysis of proposed approach across different datasets.	100

Chapter 1

Introduction

Internet of Things (IoT) devices are hardware devices, such as sensors, gadgets, appliances and other machines that collect and exchange data over the Internet. IoT devices need to be connected to the Internet to function properly. IoT devices tend to be very specific to a particular set of functions related to a specific desired outcome [226, 193, 87, 138, 21].

Most of these connected devices are consumer IoT devices [107]. Consumer IoT devices are for everyday use. *e.g.*, home appliances, voice assistance, and light fixtures. In our daily lives, people are increasingly relying on smart devices to monitor and manage the home or business. Meanwhile, vendors are now seen as responsible for servicing and updating the product over time, incurring more costs for the vendors.

Take smart thermostats and smart lighting systems as prime examples. These are ‘fit and forget’ items that we, as end users, do not expect to have to replace frequently. As end users, we are unlikely to manage these proactively, allowing updates to lapse if these are not installed automatically, and we may not even realise when support is withdrawn. Forecasts suggest that by 2030, an average home could host between 20 to 30 IoT devices, covering everything from smart speakers and lighting systems to connected appliances [197].

Unfortunately, IoT devices suffer from several security and privacy vulnerabilities. These devices are used most of the time with their default settings. The manufacturer of these devices prioritises functionality, so these devices are produced without sufficient protection, and to cut costs, vendors rarely update these devices. In addition, IoT devices have limited computational resources, which makes applying traditional security techniques inefficient [226, 193, 87, 138, 21].

1.1 Motivation, Research Question and Objectives

Over time, the sheer scale of deployed IoT will result in a bank of exposed devices in millions of homes. It will be an attacker’s playground, allowing devices to be hijacked and used to feed botnets dwarfing those created by the Mirai exploit [16, 115, 125]. We can also expect to see other attacks, such as malware and ransomware, with the user blackmailed into paying to get back control of their heating or lighting systems.

According to IoT-based attacks statistics for 2019, the average IoT device gets attacked just five minutes after it goes live [173, 63]. Once an attack works with one device, it can work with millions of devices. Additionally, the number of malware incidents involving IoT devices has grown from 813 million in 2018 to 2.9 billion the following year [20]. In 2020, IoT hardware comprised approximately a third of infected devices. In 2021, 1.8 million misconfigured IoT devices were exploited to perform large-scale attacks [223]. It took just half an hour to find passwords for most devices – including baby monitors, home security cameras, doorbells, and thermostats – often through a simple Google search of the brand name [223].

Thus, assuming that all devices in a network are well-behaved and trustworthy is no longer valid, especially in smart home office environments. Consequently, networks are more likely to incorporate vulnerable IoT devices that may be easily compromised to launch cyber attacks. The inclusion of IoT devices paves the way for a new category of anomalies to be introduced to networks.

IoT devices face significant constraints in memory, computing resources, and energy, which make deploying conventional security mechanisms insufficient. Unlike general computing devices, IoT devices often operate in constrained environments with limited processing power and storage, restricting their ability to run resource-intensive anomaly detection algorithms [126, 154, 12]. Additionally, the diversity in IoT device behaviours—driven by differences in manufacturers, communication protocols, and operational contexts—introduces further complexity. Most existing anomaly detection solutions rely on analysing single network packets or clustering large numbers of packets to detect anomalous and malicious activities [50, 249]. However, these approaches struggle to adapt to the dynamic and heterogeneous nature of IoT ecosystems. Popular detection systems (*e.g.*, Bro and SNORT) primarily use static, signature-based solutions and are designed to work on IP networks [202, 104]. Such approaches are ineffective in IoT environments, where devices communicate using a variety of protocols and exhibit non-standard traffic patterns. As a result, these infeasible solutions lead to increased false alarms and reduced detection accuracy [15, 174]. Given these challenges, a more adaptive tool is needed to differentiate between "normal" and "anomalous" behaviour in IoT devices.

An enormous effort has been made to propose anomaly detection solutions within IoT environments. Although the definition of anomaly looks straightforward, it still challenges the Era to find an accurate, robust identification model as a baseline for detecting malicious activities that decrease false alarms to a minimum while being highly scalable. Finding such a definition is challenging because of the evolving and diverse environment of consumer IoT devices, along with evolving user behaviour and potential device updates, reinstallations, and changes in physical environments.

Accurate device identification is crucial for anomaly detection as a foundation for distinguishing between normal and potentially malicious behaviours. As IoT ecosystems expand rapidly, with new devices introduced daily, securing these environments becomes increasingly complex. Evolving attack strategies further complicate detection as adversaries continuously adapt to bypass security measures. Without a reliable identification model, anomaly detection systems risk misclassifying benign deviations as threats or overlooking sophisticated attacks. Establishing a

strong identification baseline not only enhances detection precision and reduces false alarms but also eases the workload for administrators by enabling more efficient security management. Moreover, it serves as a proactive strategy, allowing for early threat mitigation and preventing anomalies before they escalate into security breaches rather than merely reacting after an attack.

Recognising these challenges, we derived our motivation for this Thesis from understanding the diverse behaviours observed from consumer IoT devices. We systematically tackle the critical issue of securing IoT environments against emerging threats by focusing on the following research question: *can a robust device identification model enhance IoT anomaly detection?* To answer this question, we focus on the following key objectives:

- **Critical Evaluation of Machine Learning Approaches for IoT Device Identification.**

Anomaly detection starts with setting up an identification model that will identify normal and malicious behaviours. Although current identification models can successfully identify IoT devices and their traffic with high accuracy, they are subject to randomness (*e.g.*, spoofing), which is known to cause high false alarms [154, 50]. We critically examine current IoT device identification solutions, evaluating their strengths and limitations. This evaluation identifies gaps and improvement areas in current methodologies using highly curated datasets and representative features. Additionally, this evaluation provides valuable insights into enhancing these identification solutions for better performance in operational deployments.

- **Development of a Hybrid Model for IoT Traffic Identification.** Our Critical evaluation confirmed the high number of false alarms from the machine learning IoT device identification. Using these identification solutions alone is unlikely the optimal solution to reduce false alarm rates. A possible solution is to introduce a model for traffic identification which combines three identification models: device behavioural profiles, fingerprints, and functional blocklists. If one of the identification fails, others will handle the identification process. We propose a hybrid model for IoT device traffic identification on home gateways, combining devices' fingerprints with behaviour profile (Manufacturer Usage Description) with destination blocklists to enhance robustness and efficiency in traffic analysis.

- **Development of a Model for IoT Malicious Traffic Identification.** The attacker honed their skills to mimic the malicious behaviour and make it identical to normal behaviour. The identification model can successfully identify incoming network traffic. However, a malicious event may or may not have caused the validated normal traffic. We present a malicious traffic detection solution utilising Symbolic Aggregate Approximation (SAX) and compression-based dissimilarity operating in unsupervised mode. The model aims to validate the rest of the traffic, improve real-time detection of anomalies, addressing evolving malicious events at the network's edge.

1.2 Contributions

The contributions of this Thesis in achieving the above objectives are:

- We analyse the current ML-based IoT identification solutions and thoroughly evaluate their performance degradation across different operational contexts, including varying modes of operation and network environments. Our evaluation identifies the key attributes affecting deployment performance and determine which attributes are critical for the effectiveness of ML-based IoT identification.
- We define these attributes as mode of operation, transferability, and observability. We organise the key attributes identified into a framework and present a methodology that provides researchers with a comprehensive method to evaluate the practicality of their solutions in real-world scenarios, aid in pinpointing performance failure causes in IoT identification solutions, and offer targeted recommendations for feature selection.
- We develop multi-model devices' fingerprints to cope with diverse IoT environments. The fingerprints consist of binary classification, where we distinguish between IoT and non-IoT devices, and multi-classification, where a majority voting of three distinct Random Forest models to classify the IoT devices, each trained on data corresponding to a specific mode of operation.
- We develop a way to combine individual, currently used IoT device identification techniques with the multi-model devices' fingerprints. We combine behaviour profiles (Manufacturer Usage Description-MUD) and destination blocklists to develop our Hybrid Traffic Identification Model (iHybrid) that enhances detection performance compared to the MUD or BlockLists alone and reduces detection latency.
- We combine Symbolic Aggregate Approximation and compression-based dissimilarity operating in unsupervised mode to detect malicious traffic. We show how SAX-encoding can represent different traffic patterns efficiently, and the compression-based dissimilarity enables compression lightweight, online traffic classification near the user, such as at home gateways or routers.
- We demonstrate that SAX-encoding and Compression-based dissimilarity (iDetector) effectively facilitate the transition from offline to online detection by representing high-dimensional time-series data as simple patterns. This combination reduces processing overhead and addresses the challenge of predefined anomalies and evolving attack patterns, enabling the detection of both seen and unseen malicious traffic.
- We develop a theoretical analysis framework to prove that SAX-encoding accurately extracts data patterns by demonstrating that the information loss in SAX-encoding over collected traffic is approximately zero, showing its low false-positive rates.

Overall, this Thesis contributes to advancing the field of IoT security by highlighting that a robust and accurate device identification model is key to successful anomaly detection. This Thesis seeks to enhance the protection of consumer IoT devices and environments against emerging security threats.

1.3 Thesis Organisation

This Thesis is organised, mirroring the particular objectives we mentioned above. Specifically, each chapter contains the following content:

- **Chapter 2.** In this chapter, we review prior work on IoT anomaly detection solutions and highlight key challenges. Additionally, we provide a preliminary analysis that includes an overview of the specific IoT devices our research focuses on (*i.e.*, consumer IoT devices), a discussion of network attacks commonly targeting or exploiting these devices, a definition of anomalies and an explanation of their different types, how anomalies are mapped to network attacks to emphasise their relevance, and finally, an examination of the network data types involved.
- **Chapter 3.** In this chapter, we evaluate current IoT device identification solutions, identify potential limitations that affect robustness, derive insights from existing solutions, and offer recommendations for developing more robust and accurate IoT identification solutions that can effectively underpin anomaly detection.
- **Chapter 4.** In this chapter, we develop a fingerprinting model that focuses on features related to the mode of operation. We also investigate how our fingerprinting can be a foundation for malicious traffic identification. In particular, we investigate how combining the fingerprinting model with behavioural profile and blocklisting ensures robust identification of what normal behaviour of IoT devices should follow, including its design and performance results.
- **Chapter 5.** In this chapter, we present a malicious traffic detection solution utilising Symbolic Aggregate Approximation and compression-based dissimilarity operating in unsupervised mode. We present iDetector, a novel approach for anomaly detection on edge devices, including its design and performance results.
- **Chapter 6.** In this chapter, we summarise the Thesis achievements, discuss the security applications of our contributions, highlight the limitations encountered, and outline potential directions for future work.

1.4 List of Publications

Posters

- **Eman Maali**, David Boyle, Hamed Haddai, “**Towards identifying IoT traffic anomalies on the home gateway: poster abstract**”, *Conference on Embedded Networked Sensor Systems (SenSys)*, Virtual Event Japan, November 2020.

Peer-reviewed Conferences

- **Eman Maali**, Omar Alrawi, Julie McCann, “**Evaluating IoT Device Identification Machine Learning Models for Network Deployment**”, *Network and Distributed System Security Symposium (NDSS)*, San Diego (USA), February 2025.

Submitted (Under Review)

- **Eman Maali**, Julie McCann, “**Privacy-Preserving IoT Malicious Traffic Classification: SAX Encoding and Dissimilarity Compression on Flow-Based Features**”, *Privacy Enhancing Technologies Symposium (PoPETS)*, Washington (USA), July 2025.

Chapter 2

Background and Preliminaries

2.1 Introduction

To build upon our motivation and research question discussed in Chapter 1, we now review prior work in anomaly detection within IoT environments. In our review, we will highlight that, while significant efforts have been made to address anomaly detection in IoT environments, many approaches underestimate the foundational importance of developing robust device identification models. Such models are critical for establishing a robust basis for detecting malicious activities, as they address the complexities of accurately identifying devices amidst diverse IoT environments.

Before reviewing the prior work anomaly detection approaches, it is essential to establish foundational knowledge about anomaly detection in IoT environments. To this end, we have structured this chapter into two sections. Section 2.2 provides a preliminary analysis, starting with an outline of the specific IoT devices our research focuses on (*i.e.*, consumer IoT devices), followed by an explanation of network attacks that commonly exploit or target consumer IoT devices. We then define an anomaly and detail the different types. Additionally, we illustrate how anomalies are mapped to network attacks to highlight their relevance. Finally, we examine the network data types prevalent in IoT environments. In Section 2.3, we review the prior work approaches for anomaly detection and conclude with a discussion of the challenges associated with these approaches.

2.2 Preliminary Analysis

In this section, we provide a preliminary analysis, starting with an outline of the specific IoT devices our research focuses on, followed by an explanation of network attacks that commonly exploit or target consumer IoT devices. We then define an anomaly and detail the different types. We illustrate how anomalies are mapped to network attacks to highlight their relevance. Finally, we examine the network data types prevalent in IoT environments.

2.2.1 Consumer IoT Devices

IoT devices are physical hardware, such as sensors, gadgets, appliances, and other machines, that collect and exchange data over the Internet [226, 193, 87, 138, 21]. These devices are programmed to serve specific applications. They are often embedded into other IoT systems, forming an interconnected ecosystem. Devices in this ecosystem communicate with each other to automate tasks, whether in-home or industrial settings. For example, an IoT device in a vehicle can detect traffic conditions and automatically notify contacts of any delays [226, 193, 87, 138, 21].

IoT applications are generally tailored to specific functions to achieve a desired outcome, such as monitoring environmental factors (*e.g.*, , temperature, pressure, or vibration) or tracking health metrics (*e.g.*, , heart rate). These devices can incorporate standard sensors (*e.g.*, pressure, flow, temperature) or custom sensors (*e.g.*, skin or heart rate). The scope of IoT devices ranges from simple household appliances to complex industrial equipment [226, 193, 87, 138, 21].

IoT devices are typically categorised into five main groups based on their usage and environment [226, 193]:

- **Consumer IoT.** Devices designed for everyday use, such as home appliances, voice assistants, and light fixtures.
- **Commercial IoT.** Devices used primarily in sectors such as healthcare and transportation, such as smart pacemakers and monitoring systems.
- **Military IoT (IoMT).** Devices used in military applications, including surveillance robots and wearable biometrics for combat.
- **Industrial IoT (IIoT).** Devices designed for industrial applications, including digital control systems, smart agriculture, and big data in manufacturing.
- **Infrastructure IoT.** Devices used to support smart city initiatives, including infrastructure sensors and management systems.

Among these categories, consumer IoT devices make up most connected devices today [178]. These include everyday household items such as smart speakers, thermostats, and connected light bulbs, which offer convenience and automation. These devices are typically connected to the internet or local networks, allowing integration with other devices to provide enhanced functionality and user experiences [57, 54, 1].

The rapid proliferation of consumer IoT devices has introduced a range of security challenges that need urgent attention [57, 54, 1]. These challenges stem from various device design, installation, and usage vulnerabilities. Some of the key security issues faced by consumer IoT devices include [18, 154, 126]:

- Weak or hard-coded passwords that are easily exploited by attackers.
- Failure by users to change default passwords, leaving devices vulnerable.

- Lack of secure update mechanisms from manufacturers, resulting in unpatched vulnerabilities.
- Abandoned devices that are no longer supported by manufacturers midway through their expected lifecycle.
- Unpatched operating software, leaving devices open to exploitation.
- Insecure data storage and transfer, despite improvements in encryption, many devices still transmit sensitive data insecurely.
- Absence of multi-factor authentication for cloud services connected to these devices.
- Open ports that expose devices to external attacks.

These security weaknesses highlight the urgent need for more secure IoT device designs and stronger management practices, especially in the consumer sector, where the adoption of such devices is expanding rapidly [18, 154, 126].

2.2.2 Types of Network Attacks

IoT devices are increasingly becoming targets for various network attacks due to their often limited security features and widespread deployment. Here are the types of network attacks that leverage and target IoT devices [223, 65, 12, 18, 17, 34, 62, 230, 231, 32]:

- **Distributed Denial of Service (DDoS) Attacks.** DDoS is a type of misuse of the rights to the resources of a network or host which is targeted at disrupting the normal computing environment and rendering the service unavailable [124, 16, 223, 65]. This consists of two types:
 - **Botnets.** IoT devices are frequently compromised and used in botnets to launch DDoS attacks. High-profile incidents, such as the Mirai botnet attack, have demonstrated the widespread use of IoT devices in such attacks. These attacks are common due to the large number of poorly secured IoT devices [124, 16, 223, 65].
 - **Amplification.** Exploiting IoT devices to amplify the scale of a DDoS attack by sending queries that generate large responses [124, 16, 223, 65].
- **Credential Theft and Brute Force Attacks.** Credential theft involves stealing user authentication credentials to gain unauthorised access. Brute force attacks use automated tools to guess passwords or other credentials through trial and error [223, 65]. This consists of two types:
 - **Default Password Exploits.** Many IoT devices come with default passwords that users often do not change, making them easy targets for brute force attacks [223, 65].
 - **Shodan Scans.** Attackers use Shodan, a search engine for internet-connected devices, to identify IoT devices with weak or default credentials [223, 65].

- **Firmware and Software Exploits.** This attack exploits vulnerabilities in the firmware (software embedded in hardware) or software of IoT devices, often by injecting malicious code or commands [223, 65]. This consists of two types:
 - **Unpatched Vulnerabilities.** Exploiting known vulnerabilities in outdated firmware or software on IoT devices [223, 65].
 - **Malicious Updates.** Delivering malicious firmware updates to compromise devices [223, 65].
- **Man-in-the-Middle (MitM) Attacks.** MitM attacks involve an attacker intercepting and potentially altering communications between two parties without their knowledge [255, 223, 65]. This consists of two types:
 - **Eavesdropping on Unencrypted Communications.** IoT devices that transmit data over unencrypted channels are susceptible to MitM attacks where attackers can intercept and alter the data [223, 65].
 - **Wi-Fi Interception.** Attackers intercept and manipulate traffic between IoT devices and their networks when devices connect to unsecured Wi-Fi networks [223, 65].
- **Network Spoofing and Phishing.** Network spoofing involves creating fake network devices or nodes to deceive legitimate devices. Phishing involves tricking users into divulging sensitive information by impersonating legitimate entities [223, 65]. This consists of two types:
 - **Fake IoT Devices.** Creating fake devices or network nodes to trick legitimate devices into connecting and compromising their communications [223, 65].
 - **Phishing Attacks.** Using fake interfaces or communications to deceive users or systems into revealing sensitive information [223, 65].
- **Brute Force Attacks.** Brute force attacks use automated tools to try many password or credential combinations to gain unauthorised access [223, 65]. This consists of two types:
 - **IoT Device Login.** Automated scripts attempt to guess login credentials for IoT devices with weak or default passwords [223, 65].
 - **SSH and Telnet Exploits.** Many IoT devices use SSH or Telnet with default credentials, making them vulnerable to brute force attacks [223, 65].

While IoT devices are susceptible to a variety of cyberattacks, we focus on the most common types, including Distributed Denial of Service (DDoS) attacks, scanning, and botnets [34, 62, 230, 231].

Each attack exploits common vulnerabilities and weaknesses in IoT devices, such as inadequate security practices, outdated software, and weak authentication methods. Addressing these issues requires a multi-layered security approach, including regular updates, strong passwords, encryption, and monitoring [126, 255, 12, 17, 32].

2.2.3 Anomaly Definition

There have been several attempts to define the nature of anomaly. Hawkins [99] defines an outlier as: “an observation which deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism [99, 132, 212, 208, 32]. An alternate definition was offered by Barnett and Lewis where [26]: “An outlier is an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data [26, 132, 212, 208, 32]”.

A general definition of an anomaly within the context of the IoT is the measurable consequences of an unexpected change in a system’s state outside of its local or global norm. This definition comprises several vital observations about the nature of the IoT data [99, 26, 44, 80, 223, 132, 212, 208, 32]:

- The majority of data captured by an IoT system can be considered “normal” because it represents the usual operating characteristics for that specific system.
- The concept of normal operation of a system can change over time for various reasons.
- The data generated by an IoT deployment represents only a view of the processes governing the system being monitored.

Anomalies in data can arise for various reasons, such as malicious activities (*e.g.*, credit card fraud, cyberattacks), unexpected events (*e.g.*, system breakdowns), or even legitimate but rare occurrences. Despite their diverse origins, all anomalies share a key characteristic: they represent significant deviations from normal patterns, which makes them essential to identify. The relevance of anomalies lies in their connection to real-world implications [99, 26, 44, 80, 223, 132, 212, 208, 32]. For example:

- Detecting fraudulent transactions prevents financial losses.
- Identifying cyber-intrusions enhances system security and data protection.
- Recognising system failures enables prompt intervention to avoid operational disruptions.

This real-life importance of anomalies is critical to anomaly detection, highlighting its value in addressing problems with practical and impactful consequences.

2.2.4 Types of Traffic Anomalies

In the literature, various categories for anomalies can be found [99, 26, 44, 80, 223, 132, 212, 208]. Each categorisation is made based on a range of factors, such as the observed duration of the anomaly, the source of the anomaly, and its type. Table 2.1 below provides an overview of the different ways to categorise anomalies, including short-lived and long-lived anomalies, network operation and abuse, and point, contextual, and collective anomalies [99, 26, 44, 80, 223, 132, 212, 208].

Grouping anomalies into categories helps define the characteristics of the desired anomaly, enabling a more targeted selection of algorithms and detection approaches. By understanding the nature and properties of each anomaly category, as researchers, we can align detection techniques

with the specific characteristics of the anomalies, such as their frequency, duration, or impact. Such categorisation ensures that the chosen detection techniques suit the problem and enhance efficiency and accuracy in identifying anomalous patterns [99, 26, 44, 80, 223, 132, 212, 208].

Category	Subcategory	Description	Example
Duration	Short-lived	Network failures, attacks, and events that resemble normal burst behaviour; exposed via mid/high-frequency signal representations.	A sudden network failure or a brief DDoS attack.
	Long-lived	Events such as flash crowds lasting up to a week; detected using low-frequency signal representations.	A week-long surge in traffic due to a major software release.
Traffic Nature	Network Operation	Includes device outages and configuration changes; characterised by steep, stable changes in bit rate.	Significant traffic change due to adding new network equipment.
	Flash Crowd	Sudden rise in traffic due to external interest; characterised by gradual drop-off.	Traffic spike from national publicity of a website.
	Network Abuse	Volumetric attacks, port scans, not always apparent in bit/packet rate measurements.	A series of port scans leading to increased but unnoticed network activity.
	Measurement	Related to data collection infrastructure issues, <i>e.g.</i> , data loss due to router overload.	Loss of flow data due to router overload, causing incomplete monitoring.
Causal Aspect	Point Anomalies	Short-lived anomalies, <i>e.g.</i> , unusual packet counts; simplest type, often due to noise or significant short-period events.	Unusually high packet transmission from a smart speaker.
	Contextual Anomalies	Anomalies relevant in specific contexts, <i>e.g.</i> , unusual traffic at certain times; determined by contextual and behavioral attributes.	High packet transmission during an unusual time for a smart speaker.
	Collective Anomalies	Group of observations forming an anomaly, <i>e.g.</i> , sequence of actions indicating an attack; not necessarily anomalous individually.	Sequence of actions in a computer network indicating a potential attack, such as [buffer-overflow, ssh, FTP].

Table 2.1: Anomaly categorisation overview with examples [99, 26, 44, 80].

2.2.5 Mapping of Network Attacks with Anomalies

Building on the discussion of various anomalies and attacks presented earlier. This section delves into the relationship between these cyberattacks and the anomalies they induce in IoT devices [44, 32]. We will address collective anomalies in IoT devices, as these anomalies often reflect the underlying cyberattacks more clearly [44, 32]. For example, a Denial of Service (DoS) attack on an IoT device typically results in a collective anomaly: a surge in connection requests that overwhelms the device, despite each request appearing benign. This collective pattern of high-volume requests creates a disruption that signals an attack, even if the requests individually do not seem malicious. In contrast, probe attacks involving information gathering and reconnaissance lead to contextual anomalies in IoT systems [44, 32]. These attacks disrupt the expected oper-

ational context by probing for weaknesses, creating anomalies in the context of normal device behaviour. User-to-root (U2R) and Remote-to-Local (R2L) attacks, which are more sophisticated and condition-specific, tend to be point anomalies. Due to their complexity, these attacks represent isolated deviations from normal behaviour and are more challenging to execute in IoT environments. Understanding these distinctions helps effectively detect and mitigate attacks targeting IoT devices [44, 32].

2.2.6 Network Data Types

A vital aspect of any anomaly detection technique is the input data. There are two central aspects of data type: the source and the nature of data type [99, 26, 44, 80]. In the following sections, we will explain each aspect in detail:

Network Data Source

An essential step in building an anomaly detection solution is selecting the appropriate network data source. The nature of the chosen data set significantly influences the types of anomalies the solution can detect and the methodologies that can be applied [99, 26, 44, 80]. Therefore, selecting a data source should be guided by the specific anomalies and approaches intended for the research. Accurate data characterisation is crucial for optimising the performance of the anomaly detection solutions [99, 26, 44, 80]. Table 2.2 below summarises the most popular data sources used in the field of anomaly detection [99, 26, 44, 80].

Selecting the appropriate network data source is essential for building an anomaly detection solution, with recent research focusing on network flow relevance to address challenges such as traffic encryption and privacy preservation.

Aspect	TCPdump	Simple Network Management Protocol (SNMP)	Internet Protocol address (IP address) Flow
Description	Packet analyser, shows Transmission Control Protocol (TCP) and Internet Protocol (IP) headers, limited due to encryption [44].	Manages network elements using Management Information Base (MIB), collects and aggregates traffic data [52].	Analyzes flows of IP packets, used in protocols like NetFlow, sFlow, IP Flow Information Export (IPFIX) [51].
Tools/Protocols	tshark, tcpdump	SNMP, MIB.	NetFlow, sFlow, IPFIX, Flexible NetFlow, NetFlow-lite.
Advantages	Provides detailed packet-level data for analysis.	Widely deployed, collects fine-grained performance metrics, remains relevant.	Lower processing cost, handles large traffic volumes efficiently, minimal privacy concerns.
Disadvantages	Ineffective with encrypted data, limited modern applicability.	May lack detailed traffic granularity, requires ongoing updates.	Issues with User Datagram Protocol (UDP) reliability, difficulty in end-to-end traffic visibility, sampling techniques may distort the network view.

Table 2.2: Comparison of network traffic analysis tools and protocols [99, 26, 44, 80].

Nature of Network Data

A vital aspect of any anomaly detection technique is the nature of the input data. Input is generally a collection of data instances (also referred to as object, record, point, vector, pattern, event, case, sample, observation, or entity). Each data instance can be described using a set of attributes

(also referred to as variable, characteristic, feature, field, or dimension) [99, 26, 44, 80]. The attributes can be of different types, such as binary, categorical, or continuous. Each data instance might consist of only one attribute (univariate) or multiple attributes (multivariate). In the case of multivariate data instances, all attributes might be the same type or a mixture of different data types. The nature of attributes determines the applicability of anomaly detection techniques [99, 26, 44, 80]. For example, for statistical techniques, different statistical models have to be used for continuous and categorical data. Input data can also be categorised based on the relationship among data instances. Most existing anomaly detection techniques deal with record or point data in which no relationship is assumed among the data instances. In general, data instances can be related to each other. Some examples are sequence data, spatial data, and graph data [99, 26, 44, 80]. In sequence data, the instances are linearly ordered, such as time-series, genome, and protein sequences. In spatial data, each data instance is related to its neighbouring instances, such as vehicular traffic and ecological data. When spatial data has a temporal (sequential) component, it is referred to as spatio-temporal data, such as climate data. In graph data, data instances are represented as vertices and are connected to other vertices with edges [99, 26, 44, 80].

The nature of input data plays a crucial role in anomaly detection, as it determines the applicability of various techniques, with attributes ranging from univariate to multivariate and relationships spanning point, sequence, spatial, and graph data. In the context of IoT environments, traffic data is typically time-series and multidimensional, requiring methods capable of handling temporal dependencies and multiple attributes.

2.3 Anomaly Detection Solutions in IoT Environments

In this section, we will review the state-of-the-art methods for anomaly detection in general and in IoT Environment in particular and discuss the challenges associated with these approaches.

2.3.1 Anomaly Detection Literature Review

Anomaly detection in network traffic is not an emerging topic. It has a long history [72, 238, 252, 83, 267, 217, 98, 254, 133, 201, 105, 24, 83, 172, 164, 172, 136, 98, 129, 92, 67, 32, 25, 210, 121, 161, 28, 102, 148, 43, 222, 188, 45, 251, 190, 119, 233, 177, 213, 33, 241, 162, 202, 187, 263, 139, 83, 214, 86, 112, 27, 75, 142, 46, 260, 218, 85, 50, 249, 124, 10]. Most existing anomaly detection approaches rely on analysing single network packets or clustering large numbers of packets to detect anomalous and malicious events [50, 249]. Traditional detection solutions, such as Zeek (Bro), SNORT, and Suricata, primarily use signature-based techniques and are designed for traditional IP networks [104, 202]. However, IoT devices present unique challenges, including limited memory, computational resources, and energy constraints, in addition to the significant diversity in their behaviours. These limitations make traditional signature-based approaches infeasible for IoT environments, increasing false positives and detection failures [15, 174]. Consequently, anomaly detection techniques, which can adapt to the diverse nature of IoT networks, are becoming an essential alternative for identifying abnormal activities and securing IoT environments.

Anomaly detection within IoT has attracted extensive research, with various approaches emerg-

ing to address the distinct challenges posed by IoT environments [164]. These methods range from rule-based to machine learning models, offering unique insights and solutions to securing IoT environments. This section explores the most recent work, highlighting their methodologies, applications, and limitations.

IoT security often starts with rule-based systems that rely on predefined behaviours and device interaction patterns to detect anomalies. Wang *et al.* [242] introduced IoT-praetor, a behaviour security detection system that extracts Device Usage Rules (DURs) from SmartApps and IFTTT applets, along with communication behaviour patterns, through traffic analysis. This rule-based method helps identify anomalous behaviours based on predefined device interactions. Likewise, Dong *et al.* [70] presented a rule generation algorithm for iForest, incorporating flow-based feature extraction for anomaly detection. Dong *et al.* aimed to improve feature tracking in IoT networks, although limitations such as the inability to use floating-point arithmetic in programmable switches pose challenges. Both approaches leverage rules to model device behaviour and detect security threats within the IoT environments.

While rule-based systems effectively detect known threats, unsupervised clustering can uncover unknown threats without requiring labelled data. Yen *et al.* [257] developed Beehive, a framework that clusters unsupervised data to detect potential security threats by analysing data-specific features from logs and identifying outliers. This method, which relies on manual labelling of these outliers, provides a promising way to uncover anomalous events in IoT networks without prior knowledge of potential attacks. Similarly, Mudgerikar *et al.* [168] proposed a model to detect fileless attacks using autoencoders. Mudgerikar *et al.* system uses unsupervised learning, but it requires further empirical validation, as its evaluation was conducted on a small sample of IoT devices. These works demonstrate the value of unsupervised methods in identifying new and unknown threats, offering flexibility in threat detection across various IoT scenarios.

Software-defined networking (SDN) has recently gained attention for its flexibility and control in IoT networks. Several studies have proposed SDN-based frameworks to improve the detection of network anomalies. For example, Mohammadi *et al.* [167] introduced SLICOTS, a lightweight countermeasure against TCP SYN (Synchronise) flooding attacks, leveraging the control plane to monitor ongoing TCP connections. Similarly, Hamza *et al.* [96] focused on applying SDN to monitor IoT devices based on MUD rules, with promising results in detecting volumetric attacks. However, both approaches highlight the challenge of scalability, which limits their practical application in large-scale deployments. The scalability issue observed in SDN-based methods points to a more general concern in anomaly detection techniques for IoT—the need for efficient and scalable systems. Moving beyond SDN, researchers have turned to policy-based systems as a potential solution to this challenge.

Policy-based approaches, which enforce rules and restrictions on IoT devices, have been explored as an alternative to SDN. For example, the CIRA Lab’s Protocol Data Analysis and Processing (PDAP) based software [49] uses predefined MUD profiles to manage IoT device behaviour, automatically monitoring devices for behavioural anomalies. While effective in controlled environments, this system’s reliance on predefined profiles limits its flexibility and applicability to

devices without such profiles. Similarly, IoTGuard [42] relies on user-defined policies to control app behaviour, though it also faces challenges due to its dependence on manual configurations. This reliance on pre-configured policies introduces limitations that lead to the consideration of more adaptive anomaly detection models, which can autonomously learn and adjust to new or unknown device behaviours. A promising direction in this regard has been integrating deep learning models, which can learn from data more dynamically.

Deep learning has revolutionised the detection of complex and subtle anomalies by leveraging large datasets for training models. Nguyen *et al.* [176] employed Long Short-Term Memory (LSTM) networks to detect collective anomalies in time-series data, while Deeplog[72] used LSTM networks to model system logs, automatically flagging deviations as anomalies. Brun *et al.* [39] designed a deep learning-based approach using dense random neural networks to detect network attacks. Doshi *et al.* [71] used deep learning to detect DDoS attacks. Mirsky *et al.* [163] uses LSTM with statistical features to measure the behaviour of the network, such as the mean or variance of the packet size or the correlation coefficient. These approaches have shown strong performance, especially when dealing with large, unstructured data such as logs or network traffic. However, challenges persist regarding the data requirements for training deep models and the time needed for model training and updates. As deep learning techniques mature, there is an increasing focus on specialised anomaly detection for particular attack types, such as botnets, which have become a critical concern in IoT security.

Botnet attacks remain among the most pressing security threats in IoT networks. Meidan *et al.* [157] and McDermott *et al.* [155] employed deep learning techniques for botnet detection, achieving promising results in detecting botnet-related anomalies. Gu *et al.* [91] proposed BotMine to detect botnet Command and Control protocol (C&C) protocol and structure and requires no a priori knowledge of botnets (such as captured bot binaries and hence the botnet signatures and C&C server names/addresses). Despite their success, these methods have yet to be validated in real-world scenarios beyond controlled experiments, limiting their applicability in more diverse environments. The challenges highlighted by these works underline the need for robust models for a wide range of attacks and perform well across different device types and network conditions.

Moving from botnet detection, it is crucial to consider the broader field of hybrid approaches that combine multiple techniques to address the evolving nature of IoT threats. Hybrid systems have shown promise by combining the strengths of different methods. Hafeez *et al.* [94] proposed IoT-KEEPER, which integrates fuzzy K-means clustering clustering with fuzzy interpolation for on-line anomaly detection, while Fu *et al.* [84] introduced Whisper, which utilises frequency-domain features for real-time anomaly detection with high throughput. These hybrid models have demonstrated significant potential but face limitations such as difficulty handling encrypted traffic or deploying in highly constrained environments. As these novel approaches illustrate, the IoT anomaly detection field is rapidly evolving, with continuous improvements in real-time detection and robustness to various attack types. However, empirical validation remains a critical issue that must be addressed, as many methods have been evaluated only in limited test environments or on a few devices.

Applying feature extraction and machine learning techniques is crucial for advancing IoT security, particularly in identifying specific attack vectors. Potluri *et al.* [194] introduced a feature-learning approach to detect false data injection attacks in Cyber-Physical Systems (CPS). Their system extracts relevant features from network packets to identify malicious activities, enhancing the accuracy of security systems in real-world environments. Similarly, Schneider *et al.* [209] proposed a framework for anomaly detection across fieldbus protocols using stacked denoising autoencoders. This combined approach of feature learning and packet classification enables the detection of anomalies with higher precision, even in complex network settings. These feature-based methods are pivotal in detecting a variety of IoT security threats, demonstrating the importance of advanced learning techniques in improving IoT security system.

Broader IoT security frameworks also focus on mitigation, leveraging machine learning to detect, prevent, or mitigate attacks. Restuccia *et al.* [198] reviewed security threats in IoT networks and proposed a machine learning-driven solution to detect and reduce polymorphic attacks. However, their work needs empirical evaluation, limiting its practical applicability. IOT SENTINEL [159] is a system designed to identify devices introduced to a network and enforce mitigation measures for those with potential security vulnerabilities. By controlling the traffic flows of vulnerable devices, IOT SENTINEL aims to protect other devices in the network from threats and prevent data leakage. However, it has not been evaluated in scenarios where IoT devices have already been compromised or targeted by cyberattacks. In a related area, PFirewall [47] provides a customisable, semantics-aware system to protect the privacy of smart homeowners. It filters data based on data-minimisation policies that are either automatically derived from automation applications or specified by users. This system offers advanced privacy control, ensuring that sensitive information is filtered before being shared across the network. While this type of research provides valuable conceptual insights, such systems' real-world implementation and testing remain crucial for validating their effectiveness in diverse IoT environments. This highlights the ongoing need for both detection and mitigation mechanisms to address IoT security threats holistically.

2.3.2 Challenges in Anomaly Detection in IoT Environments

After reviewing the current state-of-the-art IoT anomaly detection, it is evident that while these approaches have significantly advanced anomaly detection, certain challenges remain pervasive. In this section, we shall review these challenges, mainly focusing on scalability, real-time detection, the generalisability of detection results, and encrypted traffic [6, 89, 55]:

- **Scalability.** Many approaches need help scaling large IoT deployments due to their computational complexity or architectural limitations. For example, IoT-praetor [242], which extracts device interaction rules and communication behaviour rules, has been implemented only for Samsung SmartThings, limiting its ability to scale across diverse IoT networks. Similarly, Beehive [257] uses unsupervised clustering for security threat identification, but its ability to handle large, dynamic datasets has yet to be thoroughly evaluated in more extensive settings. Moreover, PFirewall [47] provides a flexible, customisable system for data flow control, but scaling this privacy-focused approach across multiple IoT devices with diverse characteristics could pose a challenge.

- **Real-Time Detection.** Latency issues related to model training and updating often prevent real-time detection. For instance, IoT-KEEPER [94] and Whisper [84], while effective for certain types of attacks, may face delays in updating models as they deal with encrypted traffic, hindering real-time responsiveness. Additionally, IoT SENTINEL [159] controls traffic for vulnerable devices, but its real-time effectiveness in situations involving ongoing attacks or compromised devices has yet to be rigorously tested, potentially limiting its responsiveness to real-time threats.
- **Generalisation.** The ability to generalise across diverse IoT environments remains a significant challenge. Approaches such as IoT-praetor [242], which focus on device-specific rules and communication traffic, may need help to generalise across IoT networks with different architectures or use cases. Furthermore, Potluri [194], which uses simulations to detect false data injection attacks, may face limitations when transitioning to more diverse real-world environments with varying attack vectors and device types. Evaluating these methods on narrow datasets or specific devices restricts their applicability.
- **Encrypted Traffic.** Handling encrypted data streams remains a challenge for many IoT security approaches. IoT-KEEPER [94] and Whisper [84] are effective at detecting various network-based cyberattacks but face limitations when dealing with encrypted traffic, as they rely on the ability to inspect network packets in cleartext. Additionally, Dong *et al.* [70], in their work on Horuseye, developed a feature extraction scheme for iForest, but their approach assumes the ability to access packet data directly, which is impossible with encrypted traffic. These limitations highlight the ongoing challenge of securing IoT networks that rely on encrypted communication for privacy and integrity.

Building on these general challenges, we present our key observations, highlighting specific limitations and opportunities in the context of IoT anomaly detection.

Our key observations. Scalability, real-time detection, generalisability and encrypted traffic are key challenges in the current anomaly detection solutions of IoT environments. These challenges are often compounded by the reliance on algorithms that are evaluated and tailored to specific attacks. While effective in controlled environments, many existing solutions must adapt when faced with real-world, large-scale IoT deployments. Therefore, an effective solution to these challenges can only partially focus on refining detection algorithms. Instead, it is crucial first to establish a robust, accurate, and adaptable device identification model. Such a model forms the bedrock for scalable and generalisable anomaly detection solutions. The importance of this foundational step is illustrated by the limitations of existing methods, which often need to pay more attention to the complexity of device identification in diverse environments, such in IoT environments. Without a solid framework for identifying devices accurately, any subsequent anomaly detection model is prone to false positives, undermining the scalability and generalisability of the entire solution. Consequently, improving the robustness and adaptability of IoT security solutions necessitates a dual focus: refining device identification and rethinking the algorithms used to process this data.

- **Focus on Algorithms Rather than Accurate Identification Models.** While much of the research has been dedicated to refining algorithms for anomaly detection and attack miti-

gation, there needs to be more emphasis on improving the accuracy and robustness of the identification models as a first step. For instance, IoT-praetor [242] focuses on extracting device usage rules and analysing traffic behaviour but needs to address the fundamental issue of how effectively the system can first identify devices before applying security measures. Similarly, Beehive [257] employs unsupervised clustering to detect threats, yet accurate device identification is necessary for any subsequent anomaly detection to be prone to false positives or misclassification, especially in large or dynamic IoT environments. This issue of inaccurate device identification also extends to PFirewall [47], where the system's ability to filter and protect data depends on the assumption that device flows are well-understood, which may only hold out if the devices themselves are identified reliably. IoT SENTINEL [159], which aims to control traffic from vulnerable devices, also overlooks the critical step of ensuring that devices are correctly identified before any mitigation measures are applied. With robust identification, these algorithms can use appropriate controls.

- **Lack of Consideration for Identification in Algorithmic Solutions.** The primary focus of these systems is often on detection algorithms such as anomaly detection, clustering, or rule extraction, with less attention paid to accurately identifying the devices within the network. As previously noted, the need for more accurate identification can exacerbate issues such as generalisation and real-time detection. For example, Potluri [194] addresses false data injection attacks using a simulation-based feature-learning approach but does not highlight how its detection framework might misidentify devices or fail to adapt in real-time if device identification is inaccurate. Similarly, Horuseye [70] develops a feature extraction scheme but assumes clear access to the network data, which can be compromised without reliable identification of the devices involved.
- **The Need for Robust Device Identification as a Foundation for Security.** Accurate and robust device identification is essential to improve the effectiveness of anomaly detection and other security measures. By addressing this as a foundational step, systems can better adapt to diverse IoT environments, handle large-scale deployments, and respond more effectively to real-time threats. Enhancing the accuracy of device identification would likely resolve several scalability and generalisation issues. Accurate models can provide more precise insights into device behaviours, allowing algorithms to operate more effectively across diverse IoT contexts, as highlighted by the challenges in encrypted traffic analysis in IoT-KEEPER [94] and Whisper [168].

In summary, while these existing works have made significant strides in developing sophisticated algorithms, there needs to be a critical gap in considering the role of robust and accurate device identification as a prerequisite for improving IoT anomaly detection.

2.4 Conclusion

In this chapter, we showed that although existing works have made significant advancements in developing solutions for detecting anomalies within IoT environments. There is a critical gap: the role of robust and accurate device identification as a prerequisite for improving IoT anomaly

detection. Evaluating the current IoT device identification solutions is essential before exploring how IoT device identification can serve as a prerequisite for anomaly detection. In the next chapter, we will evaluate current IoT device identification solutions, identify potential limitations that affect robustness, derive insights from existing solutions, and offer recommendations for developing more robust and accurate IoT identification solutions that can effectively underpin anomaly detection.

Chapter 3

Evaluating Machine Learning-Based IoT Device Identification Models for Security Applications

Associated Publication: **Eman Maali**, Omar Alrawi, Julie McCann, “**Evaluating Machine Learning Based IoT Device Identification Models for Security Applications**”, *Network and Distributed System Security Symposium (NDSS)*, San Diego (USA), February 2025.

3.1 Introduction

In Chapter 2, we showed that existing works have significantly advanced in developing algorithms for detecting anomalies within IoT environments. In Chapter 2, we also identified a critical gap: the role of robust and accurate device identification as a prerequisite for improving IoT anomaly detection. Evaluating the current IoT device identification solutions is essential before exploring how IoT device identification can serve as a prerequisite for anomaly detection. This evaluation will identify potential limitations that affect robustness, derive insights from existing solutions, and offer recommendations for developing more robust and accurate IoT identification solutions that can effectively underpin anomaly detection. In this chapter, we will evaluate device identification solutions in IoT environments, pinpoint limitations, and use these insights to propose a more robust and accurate solution.

Contributions. This chapter makes the following contributions:

- We reproduce ten state-of-the-art Machine Learning-based (ML-based) IoT identification solutions and evaluate their performance across different operational contexts. This analysis identifies key factors limiting their deployment.
- We introduce a framework for evaluating the practicality of ML-based solutions using three attributes: mode of operation, transferability, and observability.

- We pinpoint the causes of performance degradation and provide actionable recommendations for feature selection to guide future research.

The remainder of this chapter is structured as follows: Section 3.2 reviews current IoT device identification solutions and prior work addressing evaluating such solutions. Section 3.3 describes the methodology, including the data collection, reproducibility, features, performance metrics, practicality attributes, explainability analysis, and statistical analysis. All algorithms mentioned in this chapter have been freely accessible since December 2024 at GitHub [144], under an MIT license. Section 3.4 discusses the evaluation results, explainability results and improvement evaluation. Section 3.5 presents key takeaways and limitations. We conclude this study in Section 5.7.

3.2 Background

In this section, we provide a background divided into two parts: Section 3.2.1 offers an overview of ML-based identification solutions, establishing the necessary foundation for understanding their operation. Section 3.2.2 addresses the challenges of evaluating these solutions and highlights existing gaps. At the end of Section 3.2.2, we summarise the current limitations in evaluation methodologies and clarify our goal and the attributes used for evaluation.

3.2.1 Literature Review

Identification solutions, such as those by Kohno *et al.* (2005) [122] and others [88, 171, 38, 81, 66, 36], identify general network devices, relying on the TCP / IP clock skew or similar attributes and did not consider the specific attributes of IoT devices. IoT devices have distinct attributes, including various modes of operation, evolving capabilities (*e.g.*, functionality, update) and specialised communication protocols. Traditional identification solutions must be adapted for IoT identification to provide accurate, scalable, and robust identification [154, 126]. A key difference between ML-based traditional identification and IoT-specific identification is that IoT devices exhibit diverse behaviour, which requires tailored modelling per environment. IoT devices have constraints on power (battery-powered) that make their network behaviour nondeterministic or event-driven. Traditional network device identification solutions do not consider such scenarios because a common assumption is that communication of devices occurs periodically [182, 128, 120, 268, 22, 250]. However, this does not apply to IoT devices. For example, a motion sensor only activates when triggered by motion (event-driven).

The first discussions on the identification of IoT were found in Sivanathan *et al.* (2017) [219], and Lopez *et al.* (2017) [140]. Sivanathan *et al.* and Lopez *et al.* compared their approaches with Internet traffic and machine-to-machine communication classification approaches [182, 128, 120, 268, 22, 250]. Thus, we surveyed the literature in top security, Internet measurement, and network and system conferences and journals from 2017 onwards. We used IEEE Xplore and Google Scholar, with keywords such as “IoT devices,” “smart home devices,” “identification,” “fingerprinting,” “traffic analysis,” “traffic classification,” “machine learning,” “deep learning,” and “automated.” This process identified over 200 papers (2017 to 2024). This results in 96 papers with the scope of IoT device identification.

For each paper, we documented the problem, methodology, and results. We grouped the papers by applications, such as anomaly detection and device behaviour. This grouping allowed us to identify common applications that rely on IoT device identification. Additionally, we categorised the methodologies to identify common Machine Learning (ML) algorithms. Overall, we selected 17 recent papers representative of IoT device identification. The selection of the papers was based on the number of citations, visibility of the conference in which they appear, application, and seminal work (*i.e.*, the first to use algorithm, and/or feature set). Lastly, we categorised these papers by technique, covering rule-based methods [79, 203, 97] and ML-based methods [220, 152, 253, 183, 156, 58, 180, 228, 192, 69, 259, 189, 77, 266]. Many of these papers are ML-based (14 out of 17).

Category	Network Data	Paper	Venue	Algorithms	Features	Dataset	Devices	Public Code	Validation
Rule-based	Flow Level	Feng18 [79]	USENIX		Protocol		IoT		
		Saidi20 [203]	IMC		Protocol	[103]	IoT		✓
		Hamza20 [97]	IEEE Trans		Protocol	[220]	Mix	✓	✓
Learning-based	Flow Level	*Siva18 [220]	IEEE Trans.	NB, RF	Header		Mix		✓
		Marc19 [152]	IEEE Jour	KNN	Periodicity		IoT		
		*Yang19 [253]	Elsevier	LSTM	Header		IoT		✓
		*Ortiz19 [183]	IoTDI	LSTM	Protocol	[220]	IoT		
		*Meid20 [156]	Elsevier	LightGBM	Header		IoT		
		Cvit21 [58]	Springer	LogitBoost	Header	[58]	IoT		
		*Okui22 [180]	COMPSAC	LightGBM	Protocol	[180]	IoT		
	Packet Level	Than18 [228]	IEEE Trans.	K-means	Attributes		IoT		
		*Pinh19 [192]	Elsevier	RF	Header	[220]	Mix		
		*Dong20 [69]	AsiaCCS	LSTM	Header	[69]	Mix	✓	
		Yu20 [259]	USENIX	DL	Attributes		IoT		
		*Perd20 [189]	EuroS&P	TF-IDF	Attributes	[189]	IoT		
		*Fan22 [77]	IEEE Trans.	CNN	Header/Attributes	[220, 189]	Mix	✓	
		Zhao23 [266]	IEEE Tran.	LR	Header	[220, 234]	IoT		✓

We refer to real-world validation as validation. The ✓ implies the availability of code or solutions performed for real-world validation. We refer * to the algorithms that are selected for evaluation.

Table 3.1: **Qualitative comparison for selected IoT device identification current literature. Each column corresponds to a discussion of the comparison aspect of related work.**

We summarised the 17 papers in Table 3.1 and compared the network data used, algorithms, features, datasets, device settings, code availability, and validation. We find that many papers use flow-based network data as input. This usage relates to the prevalence of encrypted traffic and the overhead associated with packet-level inspection [44]. Few papers performed empirical validation with public datasets or published their code. Many papers favour deep-learning techniques for ML algorithms that rely on header information such as packet size, timing, and statistical features.

Researchers often choose deep-learning techniques over traditional ML techniques for IoT device identification. See Table 3.1. Deep-learning techniques can automatically learn complex features from high-dimensional data, reducing the need for extensive manual feature engineering. Architectures such as Convolutional Neural Networks (CNNs) and Long-Short-Term Memory (LSTMs) excel at capturing intricate patterns and temporal dependencies inherent in network traffic. In the surveyed literature, Fan *et al.* [77] employed CNN with features such as time interval, traffic, protocol, and Transport Layer Security (TLS) handshake. Dong *et al.* [69] utilised a bidirectional LSTM model with packet size, sequence, timing, and traffic volume. On the other hand, Ortiz *et al.* [183] represented TCP connections using autoencoders before training with LSTM. The advantages of deep-learning include superior performance with large and heterogeneous datasets

and adaptability to new device types and traffic patterns. However, these benefits come with drawbacks, such as significant computational resource requirements, longer training times, and the need for large volumes of labelled data. In contrast, traditional ML models are more computationally efficient, require fewer data samples, and offer greater interpretability. However, traditional ML struggles with accurately capturing the complex patterns in the data features necessary to identify IoT devices. Our work empirically documents these differences between traditional and deep-learning models.

3.2.2 Evaluating ML-based IoT Identification

Researchers have studied many evaluation attributes of ML solutions. For example, Beltiukov *et al.* [29] introduced a closed-loop ML pipeline to collect high-quality datasets and created a generalisable approach to network security. Jacobs *et al.* [111] focused on high-fidelity, low-complexity decision trees to help users identify under-specification in their models [60]. Evaluation of ML models expands beyond performance metrics such as accuracy and F1-score. The following is a set of attributes that researchers have studied in prior evaluation works:

1. **Generalisation and Robustness:** evaluating a model's ability to generalise and transfer across different domains [175, 221, 19].
2. **Fairness and Accountability:** evaluating interpretability and potential bias against specific demographic groups [175, 221].
3. **Model Scalability:** measuring effectiveness in handling increasing data volumes in distributed environments [175, 221, 19].
4. **Data Efficiency:** learning from small datasets is crucial in real-world scenarios [221, 19].
5. **Deployment Compatibility:** evaluating ease of integration across various hardware platforms.
6. **Stability Over Time:** examining consistent performance amid shifting data distributions [221, 19].
7. **Ethics and Societal Impact:** considering consequences such as energy consumption and societal effects, such as job displacement [175].
8. **Cost Metric:** evaluating deployment and maintenance costs, including hardware, software, and retraining [175, 221, 19].

However, not all these attributes account for identifying IoT specific attributes. Only some prior efforts IoT device identification reflect certain IoT attributes. Dadkhah *et al.* [61] captured network traffic from IoT devices in lab environments and analysed device behaviour in different modes of operation. However, they did not investigate how different modes affect device identification. Saidi *et al.* [203] identify IoT devices in real-world conditions by using sampled data from large networks, but Saidi *et al.* reported that their approach could have performed better when the data was incomplete or limited.

Few papers on IoT device identification evaluation reflect certain IoT attributes. Kolcun *et al.* [123] investigated performance degradation by evaluating three solutions [220, 189, 266] for IoT device identification, evaluating each model across separate time frames to ensure consistency by using the same datasets, features, and environments. Kolcun *et al.* showed that performance degradation was due to the models, not external factors such as modes of operation. Ahmed *et al.* [5], the closest work to ours, evaluated six publicly available datasets and found that Random Forest (RF) can effectively identify devices across all datasets. While reproducing Ahmed *et al.* [5], we found that the published code [4] did not correctly separate training and testing data. The authors provided a solution with a sample configuration. Still, the code’s `loadData` function always returns `None` for the test dataset, causing the training data to be reused for testing, regardless of the specified parameters [117]. We could not reconcile the paper’s claims and findings. Thus, we excluded Ahmed *et al.* [5] from our evaluation.

Although these studies explore related evaluation attributes in IoT device identification and behaviour, they differ in scope (identification vs. behaviour modelling), evaluation conditions (controlled vs. unconstrained), and data completeness (sampled vs. full data). As a result, it is challenging to combine prior works into a single evaluation for operational deployments. We aim to evaluate the practicality of IoT device identification by considering the following attributes:

- **Mode of Operation** reflects the variability introduced by the life cycle of IoT devices (*e.g.*, active vs. idle modes). This attribute aligns with stability and robustness over time.
- **Transferability** is the model’s ability to generalise across different environments (spatial) and over time (temporal). This attribute aligns with generalisation [175, 221, 19].
- **Observability** is the model’s ability to maintain accuracy across different network conditions and sampling rates (*i.e.*, congestion, packet loss or incomplete data). This attribute aligns with robustness [175, 221, 19].

We exclude other attributes since they are less relevant to deploying IoT identification solutions. For example, it may be unnecessary to evaluate adversarial robustness if the IoT identification techniques suffer in a trusted environment. Similarly, demographic bias focuses on human factors, which is less relevant for deployment considerations.

In summary, this section has detailed the background, including the overview of ML-based identification solutions and the challenges of their evaluation. Next, we will detail the methodology used in our evaluation.

3.3 Evaluation Method

In this section, we describe our data collection, reproducibility, features, performance metrics, practicality attributes, explainability analysis, and statistical analysis. The evaluation design aimed to answer three research questions by applying the three proposed attributes:

- **RQ1:** how effectively can ML-based IoT device identification solutions operate across different modes of operation and network environments?

- **RQ2:** how can performance degradation in ML-based IoT device identification solutions be pinpointed across different operational deployments, and what are the underlying causes?
- **RQ3:** can the performance of ML-based IoT device identification solutions be improved by modifying their feature selection to address practical challenges?

3.3.1 Practicality Attributes

As mentioned in Section 3.2, our work focuses on practical attributes that are relevant to IoT deployment, thus:

Mode of Operation

IoT devices, once powered on, exhibit four possible modes: setup, idle, active, and update [154]. However, in typical operating environments, IoT devices are predominantly idle or active [154, 219]. To simplify our evaluation, we focus on the two fundamental modes, idle and active, to reduce complexity while demonstrating our evaluation. We initially train the original model on a mixed-mode dataset that contains idle and active samples and test the model for (1) idle scenario, (2) active scenario, and (3) mixed scenario, both idle and active samples. We evaluated traffic from the two testbeds and averaged the results across each scenario.

Transferability

This attribute evaluates the temporal or spatial generalisability of the model. **Temporal Transferability** evaluates a model’s capacity to maintain accurate predictions over different periods. ML solutions try to learn the statistical properties of the target variable, which may change over time. This change is known as concept drift [185, 19]. We evaluated the performance of IoT identification papers with the following time delta periods between training and testing: 1 week, 2 weeks, 3 weeks and 1 month, 3 months, 6 months and 12 months. We tested the same devices across different time deltas while keeping the mode of operations mixed for both labs. **Spatial Transferability** evaluates a model in a different environment setting. We performed the spatial transferability experiment in two distinct geographical locations focusing on common IoT devices overlapped in the collection time: (1) trained on the UK testbed and tested on the US testbed, and (2) trained on the USA testbed and tested on the UK testbed. In both scenarios, the devices’ mode of operation was idle.

Observability

This attribute evaluates the robustness of IoT identification solutions when limited data is available. Most scenarios in the current ML-based IoT identification solutions use full sample rates (1:1). The default sampling rate for large networks such as *sFlow* is 1:100 [215]. However, to reflect real-world sampling, we performed experiments using three further rates to represent data loss: 1:100, 1:1000, and 1:5000 [147, 215]. These sampling rates provide an upper limit for network congestion, interference, and packet loss.

3.3.2 Explainability Analysis

To indicate features that are more influential for the model’s predictions, we relied on ML explainability [143, 237, 93] and tailored them to study IoT identification solutions. We designed our ML explainability around fundamental principles for interpreting ML-based systems: t-Distributed Stochastic Neighbour Embedding [237], Shapley Additive exPlanations [143] and feature importance [93]. We do not claim any of the techniques as a novel contribution. We limit our ML explainability to traditional ML features and not deep-learning. Deep-learning features have complex transformations and interactions within neural network architectures. Interpreting feature importance is challenging [258, 262]. For example, a deep-learning model may accurately classify an IoT device, but it is difficult to determine which specific features—like packet size or flow volume—were most critical. This difficulty arises because the model processes these features across multiple and embedded layers, making it hard to trace their contributions to the final decision [258, 262].

t-Distributed Stochastic Neighbour Embedding

To evaluate how effectively feature sets, drawn from various studies, differentiate between device categories (*e.g.*, type or vendor). We used t-Distributed Stochastic Neighbour Embedding (t-SNE), a dimensionality reduction technique commonly used to visualise high-dimensional data in two or three-dimensional space [237]. We evaluate the degree of separation between these categories in the training datasets. Well-separated clusters in the visualisation indicate that the feature set captures relevant patterns effectively, while overlapping points suggest less discriminatory power between clusters.

Shapley Additive exPlanations

To understand how each group of features collectively influences the model’s predictions, we treated the ML-based IoT device identification as a black-box model, *i.e.*, we do not examine internal workings [143]. We used Shapley Additive exPlanations (SHAP), a local explainability technique. A positive SHAP value for a feature indicates that its presence increases the prediction (*i.e.*, more important), while a negative value suggests that its absence increases the prediction (not so influential). SHAP values reveal how a group of features interact with each other to influence predictions. We optimised SHAP calculations by leveraging efficient implementations from the SHAP library, which was designed explicitly for tree-based models. We ensured the features represent the correct sequence of events, keeping the order in which data points occur, thus preserving temporal relationships and applying SHAP values to these features. This optimisation ensures we effectively indicate the model’s behaviour while handling time-dependent data.

Feature Importance

To evaluate whether the patterns learned from the training data apply to new instances. We performed permutation importance tests that evaluate the importance of features by shuffling the values of each feature and observing the impact on model performance [11, 93]. Features with more significant performance drops when shuffled are considered more important. We perform

permutation importance: first, for each feature independently while keeping the values of other features constant to evaluate its importance. Secondly, we shuffle the full feature vector for each sample in the dataset to evaluate the importance of the whole feature set collectively rather than individual features.

3.3.3 Statistical Analysis

To identify changes in feature distributions that may indicate model degradation or feature instability. We relied on comparing distributions. We designed our distribution comparison based on the fundamental principle of similarity: Population Stability Index [141], and Kolmogorov-Smirnov Test [153]. We do not claim any of these techniques as novel contributions.

Population Stability Index

We seek to detect the feature distribution shifts that influence the performance of ML models. We used the Population Stability Index (PSI), a statistical measure used to quantify changes in the distribution of a variable over time or between different datasets [141]. PSI provides insight into the model's adaptability and potential performance when exposed to new data by evaluating whether the characteristics of a training and testing dataset remain consistent. The PSI results are interpreted across three ranges. A PSI value below 0.1 indicates a negligible shift. Values between 0.1 and 0.25 indicate a slight shift, while a PSI value of 0.30 or greater is a major shift [141].

Kolmogorov-Smirnov Test

We seek to measure distributional shifts and evaluate whether the training data adequately represents the testing environment, a factor in determining the robustness of models across varied conditions. We used the Kolmogorov-Smirnov (K-S) Test, a non-parametric statistical test, to evaluate the similarity between two distributions [153]. The (K-S) test measures the maximum distance between the Cumulative Distribution Functions (CDFs) of two datasets, providing a way to determine whether they differ significantly. The (K-S) test results are interpreted using the p-value. A p-value < 0.05 indicates that the distributions are significantly different, while a p-value > 0.05 suggests negligible difference [153].

This methodology provides a framework for evaluating the practicality of ML-based IoT device identification, enabling a comprehensive analysis of the results presented in the following section.

3.4 Results

In this section, we detail the experimental setup and present the results, accompanied by interpretation, highlighting their significance and implications.

3.4.1 Experimental Setup

In this section, we detail the experimental setup, including the evaluating environment, implementation of the learning models, feature extraction, performance metrics, and the steps taken to

ensure the reproducibility of prior work.

Environment. For all experiments, we deployed ML models on a local server running Ubuntu 20.04.4, with Nvidia GPU RTX A5000, Intel CPU 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz, and 128GB RAM.

Learning Models. We implemented the ML model for LightGBM, Random Forest (RF), Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Term Frequency-Inverse Document Frequency (TF-IDF) in Python 3.9 using scikit-learn for RF, TensorFlow for LSTMs and CNNs, and LightGBM. The parameters were inherited from the baseline model to provide a fair comparison, and no hyper-parameters tuning was performed.

Algorithm	Experiments Setup	Set of Features
LightGBM	Meid20 [156]	Incoming bytes, incoming packets, TCP flags, flow duration, dst/src ports, Prot. src_TOS IPv4 dst address
	Okui22 [180]	octet Total Count Bytes reverse Octet Total Count Bytes, packet Total Count , reverse Packet Total Count
Random Forest	Siva18 [220]	Flow volume, flow duration SLD entropy, sleep time src/dst port, Cipher-suites DNS/NTP-intervals
	Pinh19 [192]	Total, mean, and standard deviation bytes transmitted
LSTM	Yang19 [253]	TOS field, TTL, IP DF, src/dst port, TCP Win, TCP RTO, TCP MSS, TCP OPT
	Ortiz19 [183]	TCP connection fields Dst port, protocol, frame length
	Dong20 [69]	time interval
CNN	Fan22 [77]	Total packets, total bytes flow rate, flow duration min/MAX/AVG time interval NTP/DNS duration entropy (10 domains) number of DNS queries TLS handshaking count
TF-IDF	Perd20 [189]	DNS-domain frequency

Table 3.2: List of features used in each experimental setup under evaluation.

Features and Features Extraction

We must account for the different features used by ML-based IoT identification solutions to enable a rigorous evaluation. Thus, we must be consistent with features employed in existing literature in IoT device identification. Features are commonly extracted at three distinct levels of abstrac-

tion: (1) packet-level features, which describe individual packet attributes. (2) Flow-level features, which capture aggregated patterns over communication sessions. (3) Deep-learning features designed to leverage advanced representation-learning techniques [175, 40].

In practice, we used *tshark* [236] to parse raw packets and extract flows. We used the feature extraction tools YAF [108] and CICFlowmeter [3]. We used *Softflowd* [110] to sample traffic for observability evaluation. Note that *Softflowd* extracts only simple features *e.g.*, packet size or time between packets. This extraction balances the trade-off between detail and volume of data storage and processing [147, 215, 116]. To ensure compatibility with ML models and preserve categorical variable information, we used standard encoding (*e.g.*, one-hot or label encoding) [175, 93].

Reproducibility of Prior Work

To ensure that the models and configurations were as consistent as possible, we matched hyperparameters and data preprocessing steps and directly used code where publicly available, *e.g.*, [5]. Code was only modified when we encountered an outdated library, such as [69], to ensure library compatibility. For [77], we used a docker container with the same library version found in the original code. Where code was unavailable [156, 180, 219, 192, 253, 183, 189], we re-implemented the code described in the original papers assuring configurations were as consistent as possible.

Similarly, some datasets used in the papers were private. Therefore, we reused the code on our datasets and carefully adjusted parameters to maintain the integrity of the experiments, such as input size or preprocessing steps, to align with the structure of our data. For example, we modified window size parameters to match the sampling frequency of our dataset and ensured that this change did not affect the overall feature extraction process. Reproducing prior work is challenging - we made conservative parameter assumptions and chose the best-performing results. We aimed to align with published results, and of the top 14 papers, we reproduced 10. The remaining four needed more feature extraction information.

Performance Metrics

It is well established that accuracy metrics are unsuitable for evaluating imbalanced datasets because the majority class can dominate predictions [170, 19]. The True Positive Rate (TPR) and False Positive Rate (TPR) are the established approach [11, 170, 19, 64]. For our experiments, we use the area under the Precision-Recall Curve (AUCPR) metric to provide insights into a model’s ability to correctly identify instances, particularly those belonging to the minority class in imbalanced datasets [11, 64].

3.4.2 Dataset Collection and Labeling

Here, we outline the data collection and labelling process, detailing the testbed setup, instrumentation, and validation methods used to ensure data reliability.

Testbed Setup. To rigorously evaluate IoT device identification solutions, we need to capture temporal, geographical, and behavioural diversity. We build two testbeds that reflect real-world scenarios across geographically distinct regions. This dual testbed approach exposes differences in

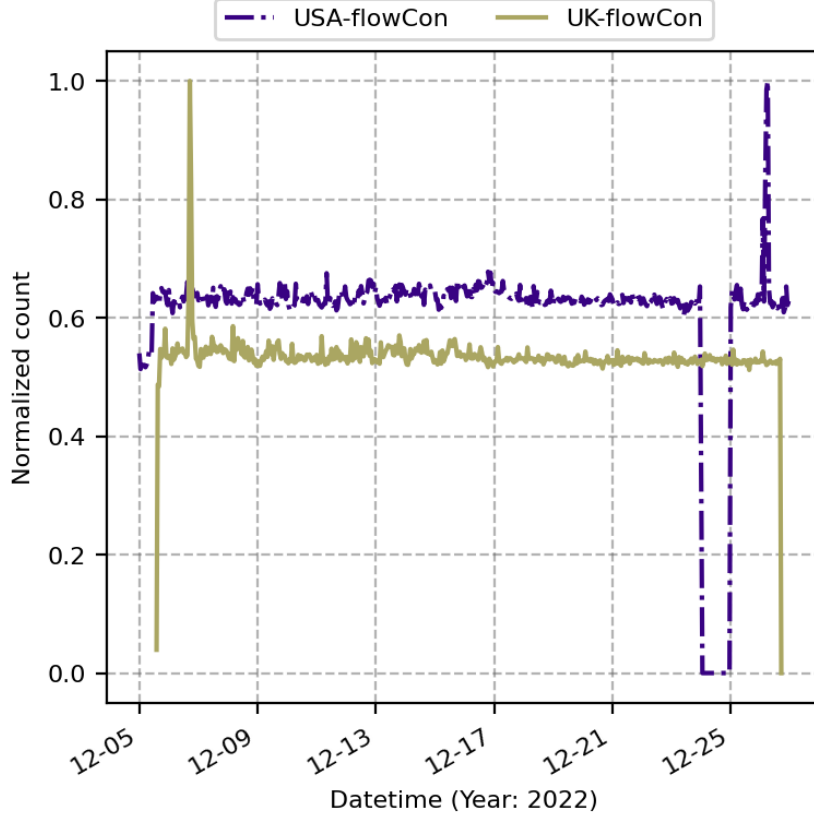


Figure 3.1: **The UK and USA testbed devices’ number of flows per day between 5th – 27th December, and the devices’ mode of operation was idle.**

market availability, network infrastructure, and user interaction patterns across regions, including overlapping and region-specific devices. The two testbeds in the UK and the USA collect data from 90 IoT devices. The USA has 75 devices, and the UK has 15 devices. See Table 3.4 for a complete list that reflected popular consumer choices [57, 54, 1], and represented the major vendors for the US and UK markets. Both testbeds have 13 common IoT devices with some non-IoT devices such as tablets, smartphones, and laptops, spanning the following categories: Cameras (security and video doorbells), smart hubs (home automation bridges for Zigbee, Z-wave, Insteon devices), home automation (smart lights, outlets, thermostats), TVs (including dongles), audio (smart speakers with voice assistants), and appliances (cleaning, cooking, weather stations). Each testbed has a Linux gateway (Ubuntu 20.04.4) with two wired network interfaces: one for Internet access via a public IP and another for the IoT devices on a private network. The testbeds have Wi-Fi adapters that connect to the IoT network and support 2.4GHz and 5GHz wireless devices. We connected the wired devices directly to a network switch. All devices communicate through standard Network Address Translation (NAT) implemented at the gateway. We used *tcpdump* to capture incoming and outgoing Wi-Fi traffic on the gateway. We configured the DHCP service on the gateway to assign static IP addresses to each device based on their MAC address.

Testbed Instrumentation. We ran idle experiments from 6th to the 27th of December 2022 while students were on break to capture the baseline behaviour of devices with no user interference. Figure 3.1 shows the daily traffic in testbeds in the UK and the US. The average flow on the testbed

in the UK is 6,177 per day, while the US is 33,843 per day. We ran active experiments on the UK testbed from the 6th to the 27th of February 2024. The users interacted with the devices by issuing commands to smart speakers and activating lights. On average, we recorded 20 lab accesses daily, each involving at least one device interaction. See Table 3.3 for a summary of device activities. We also collected traffic traces from both testbeds in January 2023, March 2023, June 2023, and December 2023. We divided all of them into continuous 21-day periods to capture IoT devices' behavioural patterns [154, 126]. This approach accounted for both temporal and geographical variability. To simulate different network conditions such as congestion and interference [35, 118, 184], we collected traffic data in April 2024 using four sampling rates to represent data loss, which we used in our evaluation. We considered public datasets as [186, 3, 104] but found their collection periods are short. We labelled all these traffic traces by year, location, device mode of operation, sample rate, and device model.

Device	Description
Amazon Echo Show	"Alexa, what's the time?" "Alexa, play relaxing music." "Alexa, How is the time?" "Change the volume of the Echo Show." "Alexa, set the alarm for 50 minutes."
Google Home Mini	"Hey Google, what's the time?" "Google, play jazz music." "Hey Google, Do I need to shop today?" "Hey Google, Change the volume of the Echo Show." "Hey Google, turn on the WEMO plug."
Ring Viedo Doorbell	Watch live video feed. Change the view of camera. Alerts of door ringing. communicate with visitors. Recording the video feed.
Wansview Camera	Watch live video feed. Change the view of camera. Motion detection. Recording the video feed.
Belkin WEMO Plug	Turn(on/off) the plug. Control the switch (on/off) using Google.
TPLINK Smart Bulb	Control the light (on/off) using Alexa. Turn (on/off) the bulb via switch. Adjust the Colour Temperature Change the Colour of lighting.
Philips Hue Bridge	Control the Philips Hue lights (on/off). Schedule the Philips Hue lights. Change the Philips Hue lights brightness levels.

Table 3.3: A sample activities of IoT devices performed during active experiments in the UK testbed.

Testbed Validity. To systematically control different variables and enhance the reliability of the results, our testbeds provide a controlled environment that mimics real-world home deployments. Our testbeds instrument everyday tasks typical of home users and provide diverse evaluation sce-

narios. Unlike real-world datasets [104], that may contain uncontrolled and latent factors that impact the evaluation of IoT identification. This setup allows us to set an upper bound on ML-based IoT identification performance under pristine conditions. This setup also supports the reproducibility and repeatability of experiments, which is essential to empirical research. Lastly, the lab-based nature of the testbeds avoids the privacy concerns inherent in real-world data collection while still allowing the simulation of diverse network conditions and real-world interactions.

Category	Devices
Home Appliances	iRobot Roomba, Belkin WeMo Crockpot, Belkin WeMo Motion Sensor, Eufy Vacuum
Audio/Speakers	Google Home Mini •, Google Home, Amazon Echo Show •, Amazon Dot, Amazon Echo, Apple HomePod, Sonos Speaker
Smart Cameras	Yi Camera, Wansview Camera, Nest Camera, Ring Video Doorbell •, August Doorbell, AVTech Push, Axis Camera, D-Link Camera, Belkin NetCamera HD+, Canary Camera, Logitech Logi Circle, Nest Bell, Nest Cam IQ, Netgear Arlo Camera, Skybell Video Doorbell, Wyze Cam Pan, Wyze Cam
Home Automation	Amazon Smart Plug, Belkin WeMo Link, Belkin WeMo Switch, Chamberlain Garage Opener, Eufy Light, Eufy Plug, Nest Guard, Nest Protect, Piper NV, Rachio 3, Sonos Beam, TPLink Kasa Smart Plug •, TPLink Kasa Smart Bulb •, Wyze Bulb, YeeLight
Hub/Controller	Philips Hub •, Eufy HomeBase, Google OnHub, Google HomeHub, HomeTroller, Insteon Hub Pro, Logitech Harmony Hub, MiCasaVerde VeraLite, Samsung SmartThings Hub, Wink 2 Hub, Withings Home
Smart TV	Amazon Fire TV, Apple TV, LG webOS TV, Roku TV, Samsung SmartTV
Game Consoles	Sony PlayStation 4, Xbox One X
Non-IoT	Bose SoundTouch, Kodi (RPi), Nvidia Shield, iPhone10, iPad Mini (Gen2), Galaxy S10, Openhab (RPi), QNAP TS-120, Securifi Almond, Switch, Synology NAS, Webthing (RPi), Webthing Gateway (RPi), Western Digital EX2 Ultra

The number of devices is summarised as follows: $N_{uk} = 15$, $N_{us} = 75$, $N_{uk} \cap N_{us} = 13$, and $N_{uk} \cup N_{us} = 90$.

We use • to indicate the presence of the device in both the UK and USA testbeds.

Table 3.4: **IoT devices under test.**

3.4.3 Evaluation Results

In this section, we detail the evaluation results, accompanied by interpretation, to highlight their significance and implications.

Algorithm	Experiments Setup	Reported Results	Reproduced AUCPR	Train: Mix		
				Test: Idle	Test: Active	Test: Mix
LightGBM	Meid20 [156]	AUCPR 0.96	0.88	0.54	0.57	0.78
	Okui22 [180]	AUCPR 0.98	0.90	0.59	0.57	0.78
Random Forest	Siva18 [220]	Accura. 0.99	0.87	0.49	0.27	0.86
	Pinh19 [192]	F-score 0.95	0.89	0.74	0.58	0.87
LSTM	Yang19 [253]	Precis. 0.95	0.78	0.11	0.12	0.11
	Ortiz19 [183]	F-score 0.82	0.76	0.11	0.11	0.10
	Dong20 [69]	Accura. 0.99	0.67	0.11	0.10	0.11
CNN	Fan22 [77]	Accura. 0.99	0.68	0.12	0.12	0.12
TF-IDF	Perd20 [189]	pAUC 0.99	0.80	0.56	0.59	0.59

We serve reproducibility results as a baseline for comparison throughout practicality metric scenarios.

Table 3.5: **Reproducibility and mode of operation evaluation for learning-based IoT identification solutions.**

Reproducibility of Prior Work

We reproduced all results using a baseline dataset with full sampling (1:1), mixed (idle and active), and testbed locations. Table 3.5 shows the reported results in the 3rd column and our reproduced results in the 4th column. We use this as the baseline for all subsequent comparisons. For example, Meid20 [156] reported an AUCPR of 0.96 while we produced an AUCPR of 0.88. Siva18 [220] reported an accuracy of 0.99. Our reproduction achieved an AUCPR of 0.87. Pinh19 [192] reported an F-score of 0.95, and our reproduction achieved an AUCPR of 0.89. Pred20 [189], the pAUC 0.99, and our reproduction produced an AUCPR of 0.80.

Mode of Operation

We train all models baseline using a dataset with full sampling (1:1), mixed (idle and active), and from both testbed locations. Idle scenario data ranges from the 6th to 27th of December 2022, and active scenario data from the 6th to 27th of February 2024.

Table 3.5 shows that the AUCPR of LightGBM reduced by $\approx 35.23\%$ in both idle and active scenarios compared with an AUCPR of 0.78 in the mixed scenario for Meid20 [156] and Okui22 [180]. In the mixed scenario, the testing samples resemble those of the training dataset (both idle and active mode). Interestingly, we observed that devices such as Amazon Echo Show, Roku TV, and Facebook Portal exhibit multiple levels of idleness. For example, the Amazon Echo Show can perform background tasks without receiving user commands, *e.g.*, shuffling photos on display. Alternatively, it can be completely idle and not engage in any activity. Indeed, a device-wise statistical analysis of the multiple idleness levels confirms different distributions, *i.e.*, p-values of the (K-S) test were < 0.01 . In active mode, IoT devices exhibit more frequent and varied communication patterns based on user behaviour, *i.e.*, users activate a smart speaker to play music or an e-book. These wide ranges introduce non-static behaviour observed when devices transit between different active patterns. In these cases, static feature-based models such as LightGBM struggle

to learn changing patterns, causing a reduction in performance. For example, the Amazon Echo Show has an average daily inter-arrival time for packets of 48.75 seconds and a standard deviation of 154.08. Yet, Google Home shows an inter-arrival time of 7.51 seconds and a standard deviation of 33.20.

For RF, Siva18 [220] AUCPR was reduced by 43.68% and 68.97% in idle and active scenarios, respectively, and achieved an AUCPR of 0.86 in the mixed scenario. We investigated this behaviour by comparing the test-train distribution in the mixed scenario using features from Siva18 [220] listed in Table 3.5. We find $PSI < 0.001$ in the mix scenario, indicating no change in the test-train feature distribution. Yet, in idle and active scenarios, some devices did exhibit a change in their distribution, such as the Google Home Mini with a PSI value of 0.34 and the TP-Link plug with a PSI value of 3.09. While RF with Pinh19 [192], AUCPR reduced by 16.85% for idle and 34.83% for active mode, and achieved an AUCPR of 0.87. We measured PSI values using features from Pinh19 [192] (Table 3.5) for both idle and mixed datasets, again observing no change in the test-train distribution, *i.e.*, $PSI < 0.001$. However, in active mode, we observe a change in one feature (the mean of the transmitted bytes) with a PSI of 0.15, indicating a slight shift in distribution. Overall, these results show that RFs perform well in specific modes of operation but do not adapt to changes in active mode.

For LSTM and CNN in Yang19 [253], Ortiz19 [183], Dong20 [69] and Fan22 [77], the AUCPR reduced by 82.35% in idle, active, and mix scenario. The AUCPR reduction indicates that LSTM and CNN models overfit or memorise patterns in the training modes. We can see the lack of generalisation between idle to active modes concretely in the Amazon Echo Show, where packet sizes change from 9.89KB to 14.36KB, and packet frequency changes from 2pps to 5pps. Lastly, for TF-IDF in Pred20 [189], the AUCPR reduced by 30% in idle, active, and mixed scenarios. The cosine similarity of the TF-IDF feature vectors for all devices in all scenarios is between 0.1 – 0.3. These similarity values indicate a change in DNS frequency patterns across different modes, which results in performance reduction.

Algorithm	Experiments Setup	Baseline AUCPR	Train: Now						
			Test: 1W	Test: 2W	Test: 3W	Test: 1M	Test: 3M	Test: 6M	Test: 12M
LightGBM	Meid20 [156]	0.88	0.71	0.73	0.70	0.76	0.72	0.65	0.65
	Okui22 [180]	0.90	0.81	0.82	0.80	0.60	0.59	0.53	0.52
Random Forest	Siva18 [220]	0.87	0.87	0.87	0.87	0.83	0.80	0.29	0.37
	Pinh19 [192]	0.89	0.89	0.87	0.87	0.84	0.84	0.64	0.67
LSTM	Yang19 [253]	0.78	0.14	0.14	0.11	0.11	0.11	0.12	0.11
	Ortiz19 [183]	0.76	0.11	0.11	0.11	0.11	0.11	0.11	0.11
	Dong20 [69]	0.67	0.11	0.11	0.10	0.11	0.10	0.11	0.16
CNN	Fan22 [77]	0.68	0.14	0.14	0.14	0.11	0.10	0.11	0.11
TF-IDF	Perd20 [189]	0.80	0.60	0.60	0.60	0.65	0.61	0.62	0.54

We refer to Week as 'W', meanwhile 'M' for Month.

Table 3.6: Temporal transferability detailed AUCPR evaluation for learning-based IoT identification solutions.

Temporal Transferability

We train all models baseline using a dataset with full sampling (1:1), mixed (idle and active), and from both testbed locations. For training, we used December 2022, while for testing, we used datasets on the following dates: January 2023, March 2023, June 2023, and December 2023, using mixed-mode operations in both testbeds.

Table 3.6 shows that for periods < 3 months the LightGBM AUCPR in Meid20 [156] reduced by 20.45%, and by 11.11% in Okui22 [180]. Equally, for periods > 3 months we see a reduced by 26.14% and 42.22% for Meid20 [156] and Okui22 [180], respectively. We measured PSI values using features from Meid20 [156] and Okui22 [180] for all train-test periods, observing slight change in the test-train distribution, *i.e.*, PSI 0.1 – 0.25. Notably, AUCPR of LightGBM in Meid20 [156] varies over the first month, from 0.73 at 2 weeks, 0.70 by 3 weeks, and increases to 0.76 in 1 month. LightGBM attempts to adjust to new patterns in Meid20 [156], causing this behaviour. Between 2-3 weeks, there is a change in the test-train data, causing a PSI value of > 0.35 . LightGBM adjusts to these patterns to improve performance between 3 weeks and 1 month to increase the AUCPR. However, beyond 1 month, there was a change in the test-train distribution PSI value, which was > 0.25 . While LightGBM attempts to adjust for this change in distribution, it is too great, causing a reduction in AUCPR value.

RF in Siva18 [220] results in a reduce of AUCPR in both periods < 6 months (by 8.05%) and > 6 months (by 57.47%). For features in Siva18 [220], we measure PSI values for periods < 6 months, observing no change in the train-test distribution (PSI < 0.001). Conversely, there is a change in the train-test distribution when > 6 months, *i.e.*, PSI > 0.35 . The AUCPR for RF in Pinh19 [192] reduced by in both periods < 6 months (by 5.62%) and > 6 months (by 28.09%). PSI values reveal stability in train-test distributions as the PSI value is < 0.25 . The variable performance of RF in both Siva18 [220] and Pinh19 [192] indicate the model’s sensitivity to temporal change, especially when PSI values indicate greater than moderate drift (PSI 0.1 – 0.25).

The AUCPR of LSTM and CNN in Yang19 [253], Ortiz19 [183], Dong20 [69], and Fan22 [77] reduce by 85.90% across all periods. When evaluating the feature distribution, we observe a significant change in train-test distribution in all experiment setups, *i.e.*, PSI > 0.35 . Indeed, the reduction in AUCPR (0.16 to 0.10) again indicates that LSTMs and CNNs overfit training patterns, failing to generalise to new patterns over time. Lastly, the AUCPR of TF-IDF in Perd20 [189] initially reduced from 0.80 during the 3 weeks to 0.60 before rising to 0.65 (*i.e.*, TF-IDF score $< 0.05 - 0.1$) in the 1 month, then reduced again to 0.54 (*i.e.*, TF-IDF score $> 0.05 - 0.1$). By empirical observation, DNS records change in the first 3 weeks and repeat at 1 month. This phenomenon is known as Domain Cycling [165, 53], where domains undergo regular changes or rotations, often reverting to previous states after certain intervals. Furthermore, this indicates that DNS domain frequency can effectively be used to distinguish between IoT devices over long temporal periods.

Algorithm	Experiments Setup	Baseline AUCPR	Train: US	Train: UK
			Test: UK	Test: US
LightGBM	Meid20 [156]	0.88	0.27	0.31
	Okui22 [180]	0.90	0.26	0.50
Random Forest	Siva18 [220]	0.87	0.50	0.64
	Pinh19 [192]	0.89	0.50	0.34
LSTM	Yang19 [253]	0.78	0.20	0.20
	Ortiz19 [183]	0.76	0.20	0.20
	Dong20 [69]	0.67	0.20	0.20
CNN	Fan22 [77]	0.68	0.20	0.20
TF-IDF	Perd20 [189]	0.80	0.74	0.57

Table 3.7: **Spatial transferability detailed AUCPR evaluation for learning-based IoT identification solutions.**

Spatial Transferability

We train all models baseline using a dataset with full sampling (1:1), mixed (idle and active), and from both testbed locations. For both training and testing, we used devices common in both testbeds, examining idle mode data collected on the 6th – 27th of December 2022 with full sampling (1:1).

Table 3.7 shows that LightGBM AUCPR in Meid20 [156] reduced by 69.32% in “train: USA, test: UK” and reduced by 64.77% in “train: UK, test: USA”. Similarly, the AUCPR in Okui22 [180] reduced by 71.11% in “train: USA, test: UK” and reduced by 44.44% in “train: UK, test: USA”. For each of these train-test combinations, we observe a difference in distribution as PSI values are > 0.35 . Such distribution changes indicate LightGBM’s difficulty in capturing changes in different spatial features.

The AUCPR of RF in Siva18 [220] reduced by 42.53% in the “train: USA, test: UK” scenario and by 26.44% in “train: UK, test: USA”. The AUCPR of RF in Pinh19 [192] also reduced by 43.82% in “train: USA, test: UK” and in the “train: UK, test: USA” scenarios reduced by 61.80%. Again, we observe that these train-test combinations have differences in distribution (PSI values > 0.35). Interestingly, we can see this shift using the port feature used in Siva18 [220]. For example, the Google Home Mini’s average daily frequency for destination port 123 (Network Time Protocol (NTP) [160]) was 37 in the UK and 97 in the USA. Similarly, port usage varies spatially, concretely: Ring Doorbell uses port 5001 in the UK, and Amazon Echo Show uses port 5000 in the USA. Such changes indicate challenges in generalising to different locations.

For LSTM and CNN in Yang19 [253], Ortiz19 [183], Dong20 [69], and Fan22 [77], we observe the same pattern as in previous cases the inability to generalise (AUCPR reduced by 73.68% to a consistent AUCPR of 0.20), due to the change in distribution between locations (PSI values > 0.35). Lastly, the AUCPR of TF-IDF in Perd20 [189] reduced by 7.5% in the “train: USA,

Algorithm	Experiments Setup	Baseline AUCPR	Train: 1:1		
			Test: 1:100	1:1000	1:5000
LightGBM	Meid20 [156]	0.88	0.30	0.23	0.25
	Okui22 [180]	0.90	/	/	/
Random Forest	Siva18 [220]	0.87	0.29	0.25	0.25
	Pinh19 [192]	0.89	/	/	/
LSTM	Yang19 [253]	0.78	/	/	/
	Ortiz19 [183]	0.76	0.071	0.072	0.071
	Dong20 [69]	0.67	0.071	0.072	0.071
CNN	Fan22 [77]	0.68	/	/	/
TF-IDF	Perd20 [189]	0.80	/	/	/

We use / when the features are packet-based the *Softflowd* could not capture it.

Table 3.8: **Observability detailed AUCPR evaluation for learning-based IoT identification solutions.**

test: UK“ scenario, and reduced by 29% in the “train: UK, test: USA“ scenario. Our empirical observations show that devices query more domains in the US than in the UK. We observed 56 unique domains in the USA and 35 in the UK during the same 21-day observation period, with all devices across both testbeds operating in idle mode. Concretely, Amazon Echo Show queries 25 domains in the US and 22 in the UK. Additionally, there are uneven queries per domain across regions, *e.g.*, DNS queries to ‘amazonalexa.com’ are more frequent in the USA (442 per day) than in the UK (180 per day). This is further evident by the similarity scores between the USA and UK domains being < 0.05 , but > 0.05 between the UK and USA domains. Such variance is due to several factors. USA domains act as testing grounds for updates and beta testing, so devices access more servers, causing more DNS queries [14]. Complex Content Delivery Network (CDN) routing in the USA allows connections to a broader range of endpoints, increasing diversity in how DNS queries are resolved [76]. Lastly, USA data privacy laws permit frequent data collection, causing numerous DNS queries for analytics, while UK laws are less permissive [14, 76, 225]. These variations indicate that TF-IDF lacks the flexibility to adapt to the distinct characteristics of IoT data across different geographical regions.

Observability

We train all models baseline using a dataset with full sampling (1:1), mixed (idle and active), and from both testbed locations. We test 1:1 against sample rates of 1:100, 1:1000, and 1:5000 collected in April 2024. Table 3.8 shows the AUCPR of LightGBM in Meid20 [156] reduced with the sample rate (65.91% at 1:100, by 73.86% at 1:1000, by 71.59% at 1:5000). Our empirical observation that the increased number of packets reduced the granularity of IoT device behaviours. For example, the number of packets sampled per-flow for Amazon Echo Show at full sample(1:1) the mean is $\mu = 1.0438 \times 10^5$ with a standard deviation of $\sigma = 3.9539 \times 10^4$ of mean packet frequency increase by 11983.8% at 1:100, by 93949.1% at 1:1000, by 396057.3% at 1:5000). In such cases, static feature-based models such as LightGBM struggle to capture changing patterns, caus-

ing a reduce in performance when varying data sampling. The AUCPR of RF in Siva18 [220] also reduced with the sample rate (66.67% in 1:100, by 71.26% in both 1:1000 and 1:5000). Similarly to LightGBM, we observe high variance in features in Siva18 [220], particularly between the 100- and 1000-second intervals. For example, flow volume for the TP-Link Lightbulb increased from 0.0004MB ($\sigma = 0.001MB$) at 1s to 0.44MB ($\sigma = 7.59MB$) at 1:100, to 1.93MB ($\sigma = 3.33MB$) at 1:1000, to 0.39MB ($\sigma = 0.66MB$) at 1:5000. This makes it harder to identify the patterns, preventing RF from fully capturing the necessary dynamics. Lastly, for LSTM in Ortiz19 [183] and Dong20 [69], we observe a reduction in AUCPR (by 90.66% to a consistent AUCPR of 0.0071). For example, in Ortiz19 [183], the increased number of TCP packets reduces the granularity of IoT device behaviours. The number of TCP packets sampled per-flow for WEMO Plug at full sample(1:1) the mean is $\mu = 1.2289 \times 10^4$ with a standard deviation of $\sigma = 3.6884 \times 10^3$ increase by 7382.5% at 1:100, by 179742.3% at 1:1000, by 382868.6% at 1:5000. In such cases, the increase in sampled data results in losing important temporal patterns on which the LSTM was trained, leading to a significant reduce and lack of generalisation.

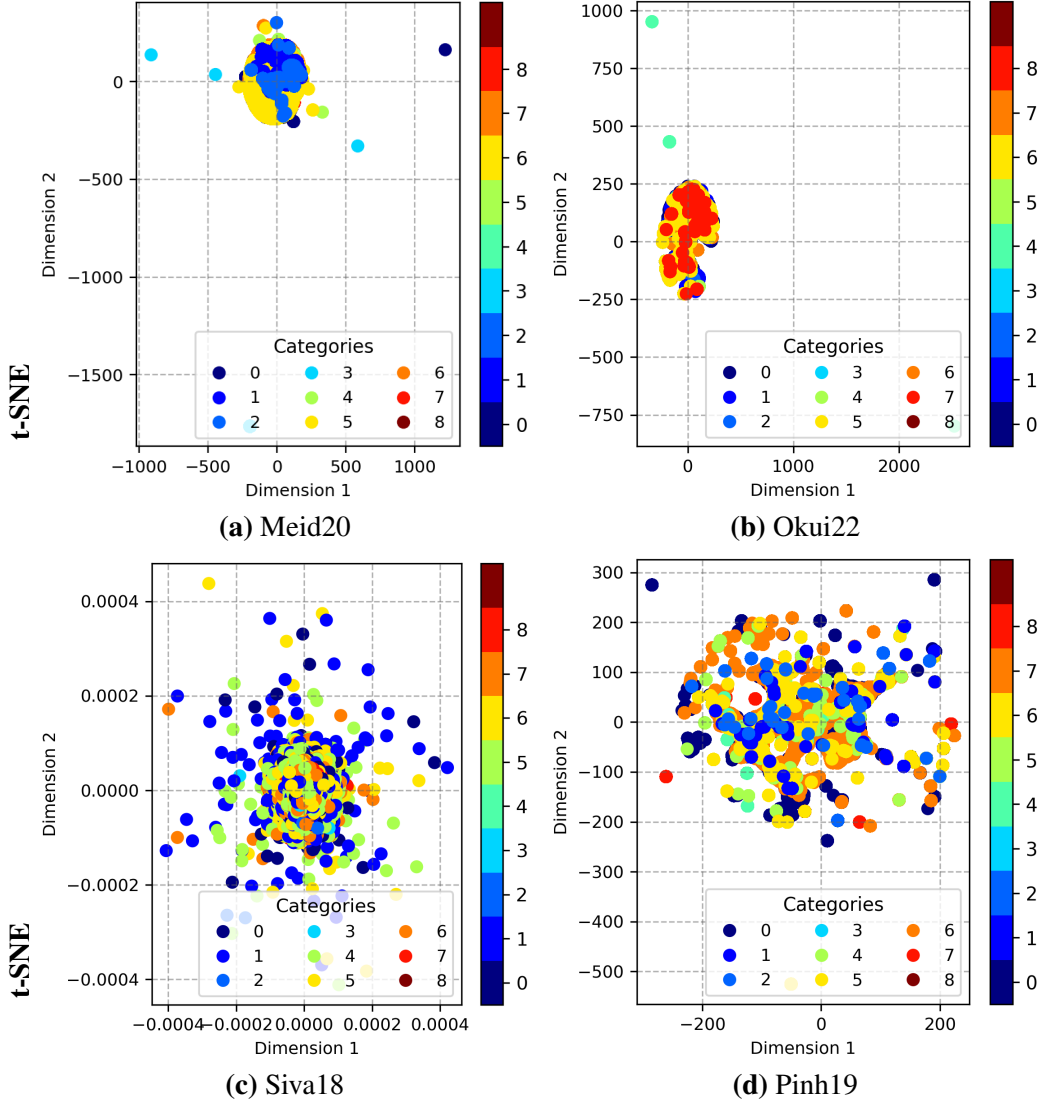
3.4.4 Explainability Results

In this section, we explain the reduction in AUCPR performance observed in Section 3.4.3 using ML explainability.

We selected experiments using packet- or flow-level features, as explained in Section 3.3.2. We analyse the following papers, Meid20 [156], Okui22 [180], Siva18 [220], and Pinh19 [192]. See Table 3.2 for the full features of each paper.

Figure 3.2 shows Meid20 [156], Okui22 [180], Siva18 [220], and Pinh19 [192] projected into a 2D space using t-SNE features from the original papers. We use Amazon Echo Show, Google Home Mini, Ring Doorbell, TP-Link Lightbulb, TP-Link Plug, Philips Hub, Wansview Camera, WEMO Plug, and Yi Camera. We focused on device-type instead of vendors because our primary goal is to evaluate features. Device-type identification presents more distinct and separable characteristics than vendor-specific identification. For example, a smart thermostat and a smart speaker exhibit different network traffic patterns based on their functions, making them easier to differentiate by device type. Two smart speakers from different brands (e.g., Amazon and Google) perform similar functions with subtle network differences, making vendor-specific identification far less distinct. However, we observe no clear separation by device-type. The clusters are neither tightly grouped nor distinctly separated.

Figure 3.3 shows the SHAP beeswarm plot for Meid20 [156], Okui22 [180], Siva18 [220], and Pinh19 [192] for the baseline training dataset. Figure 3.2(e-h) shows that 0.33% of the total features have zero SHAP values in different experiments. The zero SHAP value features do not contribute to the model predictions. In Meid20 [156], features such as *TCP Urgent (URG)*, *Congestion Window Reduced (CWR)* and *Explicit Congestion Notification (ECN) flags* have zero SHAP values. In Okui22 [180], the *total forward packet count* feature has zero SHAP value. In Siva18 [220], the *duration of NTP/DNS* features have zero SHAP values. We observed features with higher absolute SHAP values, which indicates their significance. For example, in Meid20 [156] *TCP Finish (FIN) flag* showed a high SHAP value, in Siva18 [220] *second level domain (SLD) entropy* a high SHAP



We use 0: Amazon Echo Show, 1: Google Home Mini, 2: Ring Doorbell, 3: TP-Link Lightbulb, 4: TP-Link Plug, 5: Philips Hue Hub, 6: Wansview Camera, 7: WEMO Plug, 8: Yi Camera.

Figure 3.2: **t-SNE analysis of IoT devices from the UK testbed, from 6th to the 27th of December 2022. Devices' modes of operation were mixed.**

value. Other features, such as *destination port* in Meid20 [156] and Siva18 [220], have minimal impact on the model's predictions.

The lack of distinct clustering in the t-SNE analysis and the presence of zero SHAP values show that the features used in the training datasets lack sufficient discriminatory power to differentiate between device categories.

Next, for Meid20 [156], Okui22 [180], Siva18 [220], and Pinh19 [192], we present the most important features for the testing datasets. We provide empirical interpretations for each feature only once for space limitation unless a feature's behaviour changes across different practicality attributes.

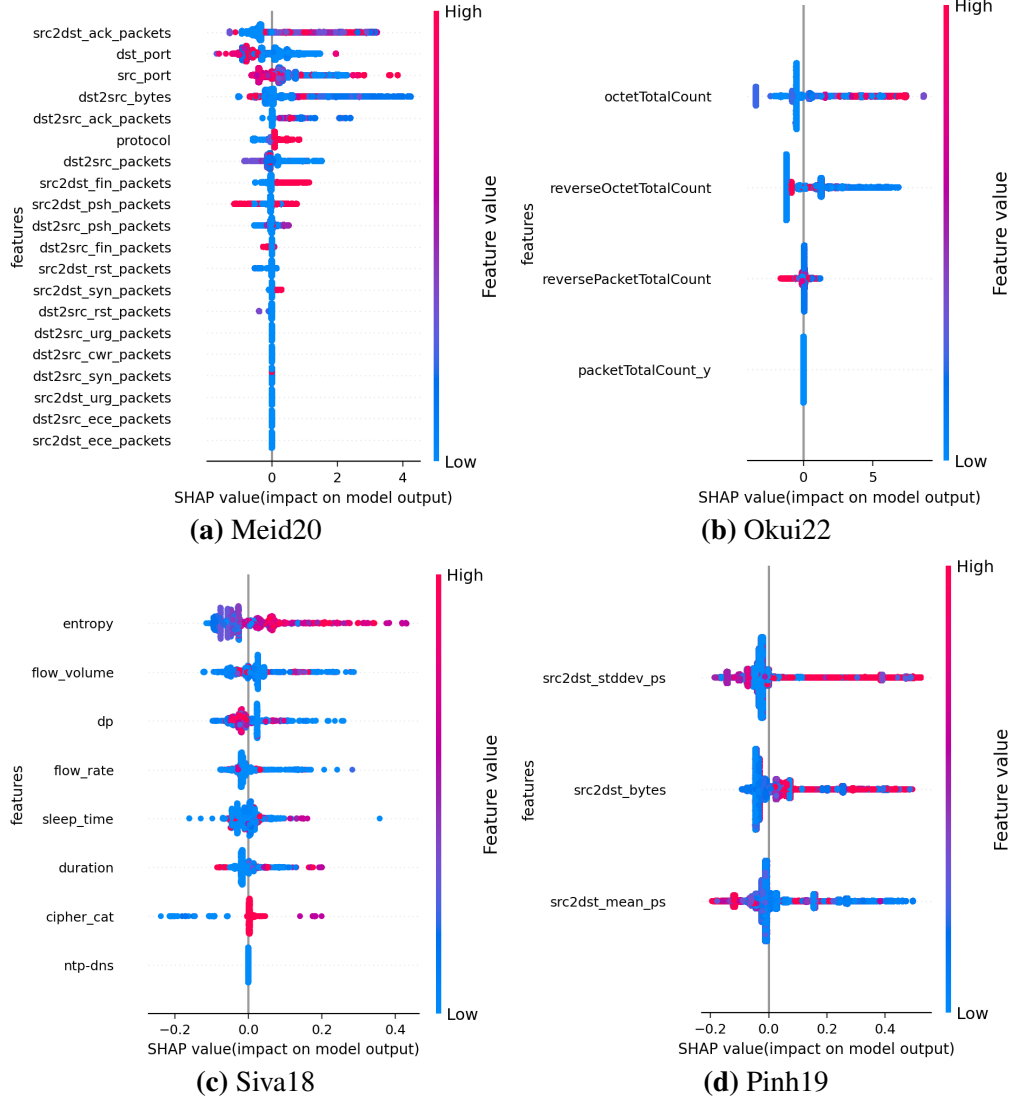


Figure 3.3: SHAP of IoT devices from the UK testbed, from 6th to the 27th of December 2022. Devices' modes of operation were mixed.

Mode of Operation

For the full permutation of the three modes of operation using Meidan20 [156] we find that the *TCP CWR flag* is the most important feature with a median of 2.60. The *TCP ECN-Echo (ECE) flag* is the least important with a median of 0.0. In the single permutation of the three modes of operation, the *TCP Push (PSH) flag* is the most important feature with a median of 0.006, while the *TCP ECE flag* is the least important feature with a median 0.0. The above results occurred because devices with high data rates, such as Yi Camera, Wansview Camera, and Ring Doorbell, set *TCP CWR flag* more frequently than devices with lower data rates, such as TP-Link Lightbulb or a thermostat. The Yi Camera set the *TCP CWR flag* 10 times per hour in idle mode and 5 times per hour in active mode while streaming video. The TP-LINK Lightbulb set *TCP CWR flag*, 1-3 times per day in both idle and active modes. Similarly, devices such as Yi Camera and Wansview Camera use *TCP PSH flag* more frequently than devices such as TP-Link Lightbulb or Plug. We observed that all devices set *TCP ECE flag* in response to congestion in the same way, regardless

of their type.

For the full permutation of the three modes of operation using Okui22 [180] we find that *total forward packet* is the most important feature with a median of 0.50. The *total reverse packet* is the least important with a median of 0.12. In the single permutation of the three modes of operation, *total forward packet* is the most important with a median of 0.51. The *total reverse packet* remains the least important with a median of 0.12. *total forward packet* is the total sending packet. The above results occurred because devices such as Philips Hue Hub, TP-Link Lightbulb, and WEMO Plug report usage data every hour to their manufacturer’s cloud. Devices such as Amazon Echo Show, Google Home Mini, and Roku TV generate traffic based on user interactions, causing event-triggered traffic patterns. The *total reverse packet* are the packets the device receives. In idle mode, devices from the same vendor, such as the TP-Link Lightbulb and TP-Link plug, receive approximately 50 packets from the TP-Link Server.

For the full permutation of the three modes of operation using Siva18 [220], the *SLD entropy* is the most important feature with a median of 0.11. The *flow duration* is the least important with a median of -0.02 . However, for the single permutation, *flow rate* becomes the most important with a median of 0.06, and *cipher suites* is the least important with a median of 0.03. Siva18 [220] used entropy to measure the most common internet domain contacted by the device. The top domain used by Yi Camera in idle mode is ‘www.yitechnology.com’ for status updates. While in active mode, while streaming video, the Yi Camera connects to ‘www.yitechnology.com’ and ‘www.yicloud.com’. The top domains used by the Amazon Echo Show in idle mode are ‘alexa.amazon.com’, ‘api.amazonalexa.com’, and ‘devices.amazon.com’, and in active mode are ‘alexa.amazon.com’, ‘video.amazon.com’, and ‘api.music.amazon.com’. The top domain for the TP-Link Lightbulb in idle and active modes is ‘cloud.tplinkwifi.com’.

For the full permutation of the three modes of operation using Pinh19 [192], the *total transmitted bytes in one minute* is the most important feature with a median of 0.15. The *mean of transmitted bytes in one minute* is the least important with a median of 0.09. For the single permutation of the three modes of operation, the *mean of transmitted bytes in one minute* is the most important feature with a median of 0.07, and *total transmitted bytes in one minute* is the least important with a median of 0.04, showing a reversal in feature importance. *total transmitted bytes in one minute* and *total forward packets* are different names for the same feature and exhibit the same behaviour. However, *total transmitted bytes in one minute* and *mean of transmitted bytes in one minute* are perfectly correlated. This correlation explains the reversal in feature importance.

Temporal Transferability

For the full permutation of the seven periods using Meid20 [156], the *TCP CWR flag* is the most important feature with a median of 3.3. The *TCP ECE flag* is the least important with a median of 0.0. For the single permutation of the seven periods, the *total bidirectional bytes* is the most important feature with a median of 0.002. The *TCP ECE flag* is the least important with a median of 0.0. The above results occurred because devices such as Yi Camera, Wansview Camera, and Ring Doorbell set *TCP CWR flag* more frequently than devices such as TP-Link Lightbulb over the seven periods. We observe a trend of consistent bytes traffic from scheduled devices such as

Philips Hue Hub and TP-Link Lightbulb. Interactive devices such as the Amazon Echo Show have a sporadic traffic pattern.

For the full and single permutations of the seven periods using Siva18 [220], the *destination port* is the most important feature with a median of 0.07. For the full permutation of the seven periods, the *sleep time* is the least important with a median of 0.05. For the single permutation, the *flow rate* is the least important with a median of -0.003 . The above results occurred because TP-Link devices use port 9999 for communication with their cloud services over the seven periods. Amazon Echo Show uses port 3478 for STUN (Session Traversal Utilities for NAT), which helps with NAT traversal for VoIP and video calls, and Wansview Cameras uses port 8899 for their cloud communication and for accessing the camera's stream. We observed that devices such as the Philips Hue Hub might have had fixed sleep times, resulting in predictable intervals of activity (e.g., sending packets every hour). Later, after a set of inactivity duration, the device enters sleep mode (longer than the initial interval). We observed a significant overlap between devices' flow rate (ratio between flow volume and flow duration), for example, Amazon Echo Show and Google Home Mini, Amazon Echo Show and Yi Camera.

For the full and single permutations of the seven periods using Okui22 [180], the *total forward bytes* is the important feature with a median of 0.23. The *packet total count* and *packet total count* are the least important, with a median of 0.0. We observe consistent byte traffic (*total forward bytes*) from scheduled devices such as Philips Hue Hub and TP-Link Lightbulb. Interactive devices such as the Amazon Echo Show have a sporadic traffic pattern. We observed the same total packet count for the Amazon Echo Show, Google Home Mini, Wansview Camera, Yi Camera, TP-Link Plug, and WEMO Plug.

For the full and single permutations of the seven periods using Pinh19 [192], the *total transmitted bytes in one minute* is the most important feature with a median of 0.31. The *standard deviation of transmitted bytes in one minute* is the least important with a median of 0.20. For the single permutation of the seven periods, the *standard deviation of transmitted bytes in one minute* is the most important with a median of 0.16. The *transmitted bytes in one minute* is the least important with a median of 0.09. The above results occurred because we observed a trend of consistent transmitted bytes traffic from scheduled devices such as Philips Hue Hub and TP-Link Lightbulb. Interactive devices such as the Amazon Echo Show have a sporadic traffic pattern. However, *total transmitted bytes in one minute* and *standard deviation of transmitted bytes in one minute* are perfectly correlated. This correlation explained the reversal in feature importance.

Spatial Transferability

For the single and full permutations of both the UK and the USA datasets Meid20 [156], the *destination port* is the most important feature with a median of 0.10. The *TCP ECE flag* and *TCP CWR flag* are the least important, with a median of 0.0. The above results occurred because we observe that in both the UK and USA, devices such as Yi Camera, Wansview Camera, and Ring Doorbell, which have a high data transmission rate, set *TCP CWR flag* more frequently than devices such as TP-Link Lightbulb or thermostat, which have low data transmission rate. Devices in the USA are more likely to set *TCP CWR flag* than devices in the UK. The USA does not have

a robust regulatory framework for IoT device services and advertisement.

For the single and full permutations of both the UK and the USA datasets using Okui22 [180], the *total forward bytes* is the most important feature, with a median of 0.50. The *packet total count* is the least important with a median of 0.0. The above results occurred because we observe that *total forward bytes* in the USA is higher than in the UK, even though both regions show similar trends. This difference is mainly due to the greater freedom that devices in the USA have regarding GDPR and the range of connections they can establish. We observed the same devices' packet count in both locations, for example, TP-Link Lightbulb (US) and Google Home Mini (UK).

For the single and full permutations of both the UK and the USA datasets using Siva18 [220], the *flow volume* is the most important feature with a median of 0.13. The *cipher suites* is the least important with a median of -0.003 . For the single permutation of the UK and the USA datasets, the *flow rate* is the most important feature with a median of 0.07. The *flow duration* is the least important with a median of -0.05 . The above results occurred because of an observed *flow volume* in the USA that is more than double that in the UK, although both regions show similar trends. This difference is mainly due to the greater freedom that devices in the USA have regarding GDPR and the range of connections they can establish. We observed *cipher suites* for the same device in the UK and USA are different, again due to the existence of GDPR in the UK. We observed the same value for devices' *flow rate* and *flow duration* in both locations, for example, YeeLight (US) and TP-Link Lightbulb (UK).

For the single and full permutations of both the UK and the USA datasets using Pinh19 [192], the *total transmitted bytes in one minute* is the most important feature with a median of 0.11. The *standard deviation of transmitted bytes in one minute* is the least important feature with a median of 0.023.

Observability

Our observability results are limited to Meid20 [156] and Siva18 [220] because the sampling rate used in the experimental set-up decreased the number of features [147, 215, 116] (see Section 3.4.3).

For the single and full permutations of four sampling rates using Meidan20 [156], the *flow duration* is the most important feature with a median of 0.31. The *incoming bytes* is the least important feature with a median of 0.002. For the single permutation, the *incoming bytes* is the most important feature with a median of 0.14. The *flow duration* is the least important feature with a median of 0.04. At a 1:100 sampling rate with a single permutation, we lost the exact flow duration due to sampling gaps compared to a sampling rate of 1:1. We observed only flow duration from devices that transmitted continuously (e.g., Yi Camera). We could still observe information for *incoming bytes*. The disrupted data pattern makes *incoming bytes* more stable and indicative of device type. At 1:1000 and 1:5000 sampling rates with full permutations, *incoming bytes* vary widely compared to a 1:1 sampling rate. We observed that the Yi Camera has a longer *flow duration* because it streams video, while the TP-Link Plug has a shorter flow duration, only sending intermittent data for status updates. We were able to observe how long each device transmits data continuously.

For the full permutations of four sampling rates using Siva18 [220], the *flow duration* is the most important feature with a median of 0.11. The *flow rate* is the least important feature with a median of -0.01 . For the single permutation of four sampling rates, the *flow rate* is the most important with a median importance of 0.07. The *destination port* is the least important with a median of 0.02. The *flow rate* and *flow duration* are perfectly correlated. This correlation explains the reversal in feature importance. With sampling, we could not observe the ports 9999 used by TP-Link devices, 3478 for the Amazon Echo Show, and port 8899 for the Wansview Camera.

3.4.5 Feature Selection Improvement Results

In this section, we explore modifications to feature selection designed to enhance IoT device identification performance, addressing RQ3: can the performance of ML-based IoT device identification solutions be improved by modifying their feature selection to address practical challenges?"

Experimental Setup

Here, we detail the experimental setup, including the dataset, learning model, adjusted feature engineering process, and the baseline used for evaluation:

Collected Dataset. We used the same traffic traces from the two testbeds at different periods (*i.e.*, 2022, 2023, 2024). We used 21 days to capture the behaviours of IoT devices.

Learning Model. Section 3.4.3 showed that RF consistently shows the least decline in performance for IoT device identification performance across all practicality evaluation scenarios. Through the practicality of implementing the improvement process, we used an RF model as a baseline using tailored features and hyper-parameter tuning the model. The resulting model is an RF with 121 trees, each with a maximum depth of 13.

Tailored Features. From the literature review, we compiled a list of features, removed duplicates, and eliminated correlated features to ensure independence (Figure 3.4). We evaluated feature importance, discarding any with zero importance and retaining those. The feature significantly impacts the model's predictions with an importance score above 0.5. This evaluation resulted in 81 features. Since we initially had over 117 features, we also applied the Pareto Principle (80/20 rule) [93, 68], where 20% of features often explain 80% of performance. From our 81 features, we found 17 that fit this rule. Further feature importance analysis showed that only 10 features consistently dominated and removed seven features with zero importance. We focused on the top 10 because fewer features allow us to understand decision-making better. For example, while sleep time may suggest a device is idle when combined with packet size, it becomes clear that it is an Amazon Echo waiting for user interaction rather than a lightbulb. The final tailored feature list is: *source port, destination port, second level domain entropy, sleep time, flow volume, TCP flags, DNS queries, cipher-suites, maximum inter-arrival time, TLS handshaking*.

Baseline. The baseline samples were from both geographical locations, with devices operating in mixed modes at a 1:1 sample rate. Starting with the tailored feature list, we systematically reduced the feature set by removing those with the lowest importance. After each reduction, we retrained the model, establishing a new baseline. We then evaluated the retrained model on the scenarios.

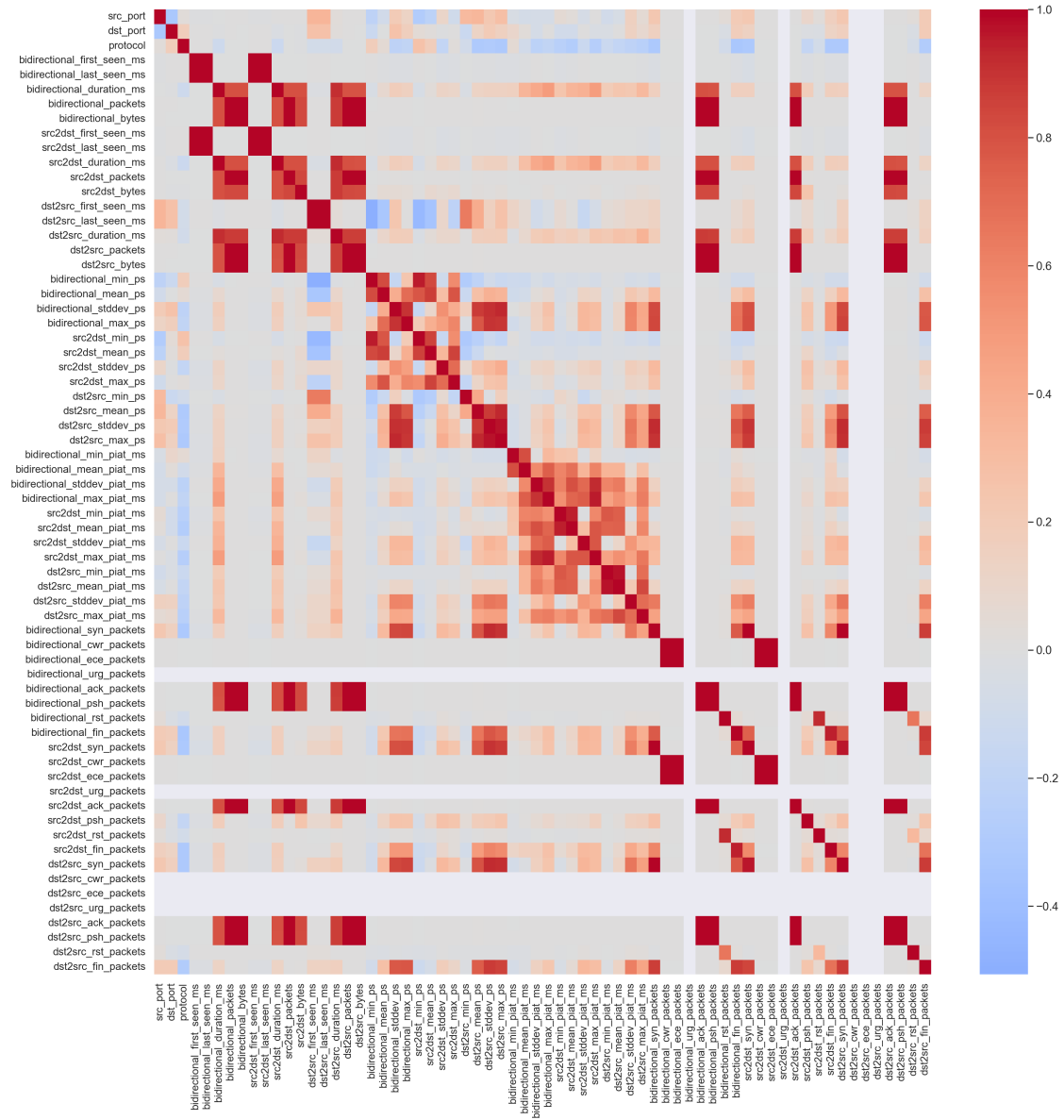


Figure 3.4: Correlation matrix for uncorrelated features used in IoT device identification solutions.

This reduction continued until we identified the minimal feature set that maintained acceptable performance across all scenarios.

Feature Selection Improvement Results

Table 3.9 shows that Iteration 1 is the most effective across all evaluated scenarios. Iteration 1 achieves AUCPR values exceeding 80% in 70% of scenarios. Iteration 1 shows the best performance in the following scenarios: Train: Mix, Test: Idle 0.64, Train: Mix, Test: Active with 0.69, Train: Mix, Test: Mix with 0.42, Train: US, Test: UK with 0.80, Test: 1 month with 0.93, and Test: 3 months with 0.82. The PSI values for tailored features across all these scenarios maintain a value < 0.01 , with (K-S) test p-values > 0.05 , indicating that the distributions are similar. However, Iteration 1 AUCPR in mixed scenario reduced from 0.94 to 0.42. The PSI values for tailored

Iteration	Features Subset	Baseline AUCPR	Mode of Operation			Spatial Transferability		Temporal Transferability			
			Train: Mix Test: Idle	Train: Mix Test: Active	Train: Mix Test: Mix	Train: UK Test: US	Train: US Test: UK	Train: Now Test: 1M	Train: Now Test: 3M	Train: Now Test: 6M	Train: Now Test: 12M
0	dst port, src port SLD entropy, sleep time TCP flags, flow volume Cipher-suites, DNS queries TLS handshaking MAX inter-arrival time	0.94	0.62	0.29	0.29	0.91	0.86	0.85	0.84	0.27	0.33
1	dst port, src port SLD entropy, sleep time TCP flags, DNS queries MAX inter-arrivaltime	0.98	0.64	0.69	0.42	0.74	0.80	0.93	0.82	0.27	0.30
2	dst port, SLD entropy, Sleep time MAX inter-arrivaltime	0.97	0.63	0.68	0.42	0.75	0.79	0.93	0.80	0.28	0.30
3	dst port, SLD entropy, flow volume	0.96	0.36	0.36	0.36	0.73	0.78	0.91	0.80	0.28	0.31
4	dst port, SLD entropy, sleep time	0.98	0.35	0.91	0.91	0.80	0.80	0.83	0.74	0.26	0.31
5	Sleep time, SLD entropy	0.90	0.52	0.72	0.23	0.66	0.71	0.84	0.69	0.23	0.20
6	Sleep time	0.75	0.34	0.43	0.14	0.55	0.56	0.53	0.49	0.16	0.15

We use 'M' to refer to months.

Table 3.9: Improvement evaluation results AUCPR for IoT identifications tailored features, under device mode of operation and transferability metrics.

features in mixed mode are > 0.30 , indicating that the distributions have changed. This decrease in AUCPR shows that mixed operational modes are challenging for the model.

Table 3.9 shows that Iteration 6 is the worst across all evaluated scenarios, achieving AUCPR values < 0.60 in all scenarios. For example, the performance in the mixed mode AUCPR reduced from 0.75 to 0.14. Similarly, in Iteration 6, AUCPR reduced from 0.75 to 0.16 in the temporal transferability tests. The PSI between the training and testing dataset features was > 0.35 . These PSI results show that the feature distribution has changed. Iteration 2 and 4 succeeded in 7 out of 10 scenarios. For example, in Iteration, 3 recorded success in 6 out of 10 scenarios, with an AUCPR of 0.96. However, the AUCPR for the idle, active, and mixed modes was only 0.36. Iteration 5 succeeded in 6 out of 10 scenarios while the baseline AUCPR of 0.90 reduced to 0.52 in idle mode, 0.32 mix mode, and the temporal transferability was 0.20 at 12 months. Across all these iterations, the PSI between the training and testing dataset features was > 0.35 . These values show that the feature distributions remain unchanged.

This results section has presented evaluation results, causes of performance degradation, and feature selection modification results. Next, we provide insights and recommendations for researchers and network operators to enhance IoT device identification solutions.

3.5 Discussion

In this section, we will detail the key takeaways and lessons from our results while also highlighting the limitations of our evaluation.

3.5.1 Takeaways and Lessons

Here, we present the key takeaways and lessons learned from our evaluation, highlighting the most important insights gained throughout the evaluation process:

Our evaluation indicates that network operators would benefit from RF models with feature sets

that reflect device outbound traffic patterns such as those seen in Pinh19 [192] as they generalise well, particularly across modes of operation. Training the model in idle mode and then conducting predictions for different periods was also beneficial as IoT devices remain idle for extended periods, and the range of possible behaviours beyond that are too large to model.

Models generally degrade in performance over time due to the changes in device capability, *e.g.*, updates or functionalities. Combining LightGBM for longer-term predictions and RF for shorter-term identification can achieve more consistent temporal performance, as seen from their AUCPR in Table 3.6. Additionally, retaining models more frequently (*e.g.*, every two weeks) helps prevent the drop-off in temporal degradation.

We observe models with DNS query (as in Siva18 [219]) features reduce temporal degradation. As such, we suggest using RF with DNS frequency. We reason this is due to the cyclical nature of DNS domains combined with the reduced overfitting from the ensemble learning of RF. We also see DNS frequency as a useful feature for spatial generalisation using TF-IDF. This is because TF-IDF represents DNS domains more nuancedly, capturing the significance of specific domains for spatial identification.

Our evaluation (Table 3.8) shows that full packet captures sample and transferring to higher sample rates reduces performance. While using full packet sampling benefits test time identification, it can be impractical in larger networks (*e.g.*, overhead, packet loss, packet delay). Thus, alternative techniques can be employed, such as network segmentation, where networks can be divided into smaller segments.

The observations made in feature expandability suggest the importance of how features are represented. First-order statistical features (*e.g.*, mean, variance) may not capture the distributional information, and instead, features such as entropy should be used. Variables with high variance between high- and low-transmission devices should also be avoided. Such variance can lead to overlapping patterns, reducing the ability to distinguish device types accurately. We suggest using features with binary values, such as being set or not set. An example is the TCP congestion windowing flag, which can provide a clear, non-overlapping indicator of device transmission behaviour.

The observations made in improvement evaluation suggest that a broad range of features better capture the device behaviours, maintaining high AUCPR scores, while models with single features suffer when distribution shifts. Careful consideration should ensure that models learn to detect the most relevant environmental features. For example, adaptive algorithms can select important features based on the real-time context, such as mode of operation, transferability, or observability.

3.5.2 Limitations

Here, we discuss the limitations of our evaluation, acknowledging factors that may have influenced the results and suggesting areas for improvement in future research:

The data used in our evaluation is limited because it does not cover all IoT smart home categories; for example, we did not include smart kitchen appliances. We did this exclusion to ensure that

the same model of IoT device was used in both US and UK labs to ensure data comparability. Our data included only Wi-Fi IoT communication. We excluded Zigbee and Z-Wave to establish a solid methodological foundation before extending to other communication technologies. Including multiple protocols would add significant complexity due to differences in their communication stacks, hardware requirements, and operational behaviours. Future evaluation would include other IoT communication protocols.

Our data analysis methods were limited by excluding feature selection techniques such as Mutual Information Analysis (MIA) or Principal Component Analysis (PCA). We did this to focus on the features examined in the original works that we evaluated. However, adding MIA or PCA to future work could offer further insights into how feature combinations contribute to device identification, revealing the most relevant features that provide complexity reduction.

We excluded deep learning features from our ML explainability as those features have complex transformations and interactions within neural network architectures. This complexity arises because the model processes these features across multiple and embedded layers, making it hard to trace their contributions to the final decision [258, 262]. Future evaluation could explore deep learning explainability.

3.6 Conclusion

In this chapter, we critically examined current IoT device identification solutions, evaluating their strengths and limitations. Our evaluation identified key gaps and areas for improvement, particularly concerning the high number of false alarms associated with machine learning-based IoT device identification. Relying on these identification solutions alone is unlikely to be sufficient for reducing false alarm rates. We uncovered several factors that impact the performance of IoT identification models in real-world network deployments, including variability in device operation modes and spatial, temporal, and portability factors. Additionally, our evaluation provided valuable insights into enhancing existing identification solutions for improved performance in operational settings. Building on these recommendations, in the next chapter, we aim to develop a robust identification solution for IoT environments and investigate how such a solution can serve as an initial step for anomaly detection.

Chapter 4

A Hybrid Model for IoT Device Traffic Identification on the Home Gateway

Associated Publication: **Maali, E.**, Boyle, D. and Haddadi, H., 2020, November. *Towards identifying IoT traffic anomalies on the home gateway*. In Proceedings of the 18th Conference on Embedded Networked Sensor Systems (pp. 735-736).

Prof. Haddadi and Dr. Boyle supervised part of this chapter. I acknowledge their fruitful discussions and advice.

4.1 Introduction

In Chapter 2, we identified a critical gap: the role of robust and accurate device identification as a prerequisite for improving IoT anomaly detection. In Chapter 3, we examined how ML-based IoT device identification models degrade over time and provided recommendations for improvement. Building on these recommendations, in this chapter, we aim to develop a robust identification solution for IoT environments, which can serve as an initial step for anomaly detection. One of the key takeaways from Chapter 3 suggests that careful consideration is needed when detecting the most relevant environmental features. For example, adaptive algorithms can select important features based on the real-time context, such as mode of operation, transferability, or observability. Building upon these recommendations, we first develop a fingerprinting model that focuses on features related to the mode of operation. And as our primary goal is to answer our research question, "The role of robust and accurate device identification as a prerequisite for improving IoT anomaly detection." We will also investigate how our enhanced fingerprinting can be a foundation for malicious traffic identification.

In particular, we will investigate how combining the enhanced fingerprinting model with behavioural profile and blocklisting ensures robust identification of what normal behaviour of IoT devices should follow. Thus, in this chapter also we explore how to combine this fingerprinting model with a behavioural profile, which defines the intended communication patterns for a device (e.g., , MUD [199]), and a predefined blocklist (which specifies non-functional destina-

tions for each device [150]), to create a hybrid model for traffic identification. A hybrid approach that combines behavioural profiling, fingerprinting, and blocklisting outperforms single-method solutions by leveraging their complementary strengths. Behavioural profiling adapts to evolving device activities, fingerprinting ensures consistent identification based on inherent characteristics, and blocklisting provides a direct defence against known threats. Relying on a single method often results in blind spots—behavioural profiling alone may struggle with adversarial evasion, fingerprinting can fail with dynamic device changes, and blocklisting is ineffective against novel attacks. By integrating these techniques, a hybrid model enhances accuracy, resilience, and adaptability, providing a more comprehensive and robust security framework for IoT environments. In our approach, the MUD profile serves as the currently available behavioural profile; however, our hybrid model is adaptable, allowing any newly proposed behavioural profile to seamlessly replace the MUD profile and operate within the identification and validation phases.

Contributions and Roadmap. This chapter makes the following contributions:

- Building upon the evaluation and the recommendations derived through Chapter 3, we develop multi-model devices’ fingerprints to cope with diverse environments such as those in IoT networks. The fingerprints consist of two phases: in the first phase, binary classification, where we distinguish between IoT and non-IoT devices, in the second phase consists of multi-classification, where three distinct Random Forest models to classify the IoT devices, each trained on device data corresponding to a specific mode of operation (setup, idle, and active), the identification results is majority voting of three Random Forest models.
- To provide better effectiveness, we develop a way to combine individual, currently used IoT device identification techniques with the multi-model devices’ fingerprints. We combine behaviour profiles (Manufacturer Usage Description) and destination blocklists to develop our Hybrid Traffic Identification Model (iHybrid) that enhances detection performance compared to the MUD or BlockLists alone and reduces detection latency. This combination is a step towards finding a broader definition of ‘normal’ in dynamic environments such as those in IoT networks.

The remainder of this chapter is structured as follows: in Section 4.2, we review current IoT device identification solutions. In Section 4.3, we describe the iHybrid Model Architecture. All algorithms mentioned in this chapter are freely accessible since December 2024 at GitHub [144], under an MIT license. In Section 3.4, we discuss the experimental setup, data collection and iHybrid evaluation results. In Section 4.5, we present key takeaways and limitations. We conclude this study in Section 4.6.

4.2 Background

In this section, we review approaches used for IoT identification with the goal of malicious traffic identification, focusing on device model identification and traffic analysis. The first part reviews the current state-of-the-art fingerprints generated from network traffic characteristics. In the second part, we review IoT behavioural profiles. In the third part, we review the use of blocklisting

for identification purposes.

While some of the state-of-the-art papers discussed here overlap with those found in Chapter 3, this section provides a more detailed review of the techniques, whereas Chapter 3 focuses on evaluating these solutions and understanding the performance degradation.

4.2.1 IoT Fingerprinting

Here, we review the current state-of-the-art fingerprints generated from network traffic characteristics:

Numerous papers have investigated using network traffic analysis to detect IoT devices in a network or examine their behaviour. Miettinen *et al.* [159] presented a Random Forest (RF) classification capable of automatically identifying the types of devices being connected to an IoT network and enabling enforcement of rules for constraining the communications of vulnerable devices to minimise damage resulting from their compromise. Feng *et al.* [79] proposed an Acquisitional Rule-based Engine (ARE) that automatically generates rules for discovering and annotating IoT devices without any training data. Feng *et al.* method leverages application-layer response data from IoT devices and product descriptions from relevant websites to create device annotations. However, the reliance on transactions—the association between response data from IoT devices and web pages—means that if response data is faked, as in the case of honeypots, the transaction set may contain erroneous information, leading to inaccurate rules. Sivana *et al.* [220] focused on developing a classification technique that distinguishes between IoT and non-IoT devices and uniquely identifies IoT devices. Sivana *et al.* developed a multi-stage ML-based classification algorithm and demonstrated its ability to identify specific IoT devices with over 99% per cent accuracy based on their network activity. Thangavelu *et al.* [228] proposed a centralised approach for solving the problem of fingerprinting IoT devices is to train a supervised ML model using network traffic of devices and use that model at a centralised controller in the network, to classify devices based on the traffic characteristics.

Ortiz *et al.* [183] presented stacked autoencoders for automatically learning features from device traffic. Ortiz *et al.* learning the classes of traffic observed and probabilistically modelling each device as a distribution of traffic classes. Ortiz *et al.* focused on identifying new devices and providing class-level feedback about the kind of device being observed (IoT vs. Non-IoT). Yang *et al.* [253] utilised the feature of network protocols and generated IoT device fingerprints based on neural network algorithms. Dong *et al.* [69] proposed a traffic analysis framework based on sequence-learning techniques such as LSTM and leveraged the temporal relations between packets for the attack of device identification. Saidi *et al.* [203] utilised a combination of passive, sampled network traces and active probing experiments to identify IoT devices based on the type of destinations they contacted, such as manufacturers, advertisers, and third parties. Saidi *et al.* evaluated their approach using controlled experiments that involved tunnelling the traffic of two IoT testbeds comprising 96 devices to an Internet Service Provider (ISP). However, the Saidi *et al.* method requires sample devices to determine which domains are contacted and relies on the network activity of IoT devices, leading to decreased detectability and increased detection time when traffic volume is low. Additionally, the Saidi *et al.* approach struggles to detect IoT devices using

shared infrastructures.

Perdisci *et al.* [189] developed *IoTFinder*, which utilises statistical Domain Name System (DNS) traffic fingerprints from real devices to efficiently match these fingerprints across large volumes of DNS traffic from diverse geographical network locations. *IoTFinder* operates as a multi-label classifier capable of detecting devices, even when they are behind Network Address Translation (NAT) and their traffic is mixed with other and non-IoT devices on the same network. *IoTFinder* similar to Saidi *et al.* relies on the availability of sample devices and network activity, limiting its effectiveness in scenarios with low traffic volume or when devices operate on shared infrastructures. Yu *et al.* [259] presented OWL, a multi-view wide and deep learning (MvWDL) framework that extracts network traffic features from passively received broadcast and multicast (BC/MC) packets. Yu *et al.* assessed the inconsistencies across views in the multi-view classifier to identify malicious events.

Aun *et al.* [23] presented a two-level ML pipeline for IoT device recognition, using (1) Support Vector Machine (SVM) and Decision Tree (DT) to model the spatial behaviours and (2) Recurrent Neural Network (RNN) to model the device-to-device temporal actions. Aun *et al.* ground truth is that the communication behaviours of IoT devices are more deterministic using control plane traffic instead of data plane traffic. Liu *et al.* [137] proposed a device identification method for the IoT based on directional packet length sequences in network flows and a deep convolutional neural network. Liu *et al.* methods' constructed device fingerprints from packet length sequences and uses convolutional layers to extract deep features from the device fingerprints. Salman *et al.* [206] proposed a framework specifically for IoT devices identification and malicious traffic detection. This framework extracts features per network flow to identify the source and the type of the generated traffic and to detect network attacks. Fan *et al.* [77] proposed AutoIoT, a CNN classifier for all devices, and decides the type of another device by the output of softmax layers of the CNN classifier, a semi-supervised learning method. Fan *et al.* evaluation on two public datasets shows that AutoIoT can identify new device types only using 1.5 2.5 hours' traffic and still have high accuracy after updating.

Chowdhury *et al.* [48] proposed a deep learning-based device fingerprinting model using these two features for the classification task, with 100 consecutive packets' information utilised to generate fingerprints as graphs. Chowdhury *et al.* device fingerprinting model demonstrates over 99% and 95% precision in distinguishing between known and unknown traffic traces and in identifying IoT and non-IoT traffic traces, respectively. Zhao *et al.* [266] presented IoTPROFILE, an IoT device identification framework via a time-series dictionary. IoTPROFILE trained random convolutional kernel transformations in two ways and aggregations to capture discriminative patterns and form the frequency count of recurring patterns to profile the network behaviours of IoT devices over some time. Zhang *et al.* [264] proposed IoT-Siamese, a type identification method for IoT devices based on few-shot traffic, which mainly relies on the Siamese network to solve the problem of few samples. Thom *et al.* [229] applied a combination of locality-sensitive hashing and neural networks to identify specific devices based on their network traffic, eliminating the need for complex feature engineering and model retraining.

While the aforementioned approaches are either rule-based or ML-based, other researchers use hybrid approaches using either majority voting of a training group of ML models such as Pinheiro *et al.* [192] or LogitBoost and LightGBM (LGBM), as Cvitic *et al.* [58], Meidan *et al.* [156], and Okui *et al.* [180]. Pinheiro *et al.* evaluation included the algorithms k-Nearest Neighbors (k-NN), DT, RF, SVM and Majority Voting, some of the most popular algorithms for traffic classification. Pinheiro *et al.* [192] used packet length statistics from encrypted traffic to characterise the behaviour of IoT devices and events in a smart home scenario. Pinheiro *et al.* used only the statistical mean, the standard deviation and the number of bytes transmitted over a one-second window, which can be extracted from the encrypted traffic, making using Transmission Control Protocol (TCP) vectors unnecessary. Cvitic *et al.* trained LogitBoost as a Multiclass classification model using 13 network traffic features generated by IoT devices. Cvitic *et al.* analysed the possibilities of applying such features for classifying devices, regardless of their functionality or purpose. Meidan *et al.* trained the LGBM algorithm to identify devices behind a NAT in the presence of encrypted, non-TCP or non-DNS traffic. Meidan *et al.* [156] proposed a ML-based method that can detect specific vulnerable IoT device models connected behind a domestic NAT. Okui *et al.* [180] used Internet Protocol Flow Information Export (IPFIX) as the traffic data to train LightGBM as a multiclass classifier to generate the device identification model.

Despite the promising results of these approaches for identifying connected IoT devices, these approaches often experience performance degradation (high false positive rates) when deployed in real-time environments. This performance degradation issue arises from the need to account for the distinct attributes of IoT devices, including various modes of operation, evolving capabilities (*e.g.*, functionality and updates), and specialised communication protocols. Challenges persist even when specific attributes, such as the mode of operation, are considered. For example, Ammar *et al.* [13] proposed a Locality-Sensitive Hashing for IoT Fingerprinting (LSIF) framework that uses the hash of network traffic flow for IoT device identification. LSIF does not require feature extraction from the data, tuning the model's parameters, and retraining the model if there is new labelled data. They evaluated LSIF on three different IoT device traffic data, including setup, active, and idle state of the devices. Marchal *et al.* [152] trained a kNN model identifying the type of an IoT device in any operation mode. Experimental evaluation of AUDI using a large dataset comprising 33 typical commercial IoT devices, an autonomous, distributed system for learning and identifying the type of an IoT device. However, these fingerprinting approaches are vulnerable to spoofing, and device identification validation during attacks still needs to be addressed.

These limitations highlight the need for a fingerprinting model to account for the distinct attributes of IoT devices, including various modes of operation, evolving capabilities (*e.g.*, functionality and updates), and specialised communication protocols. These limitations also highlight the need for hybrid models that integrate multiple identification strategies to effectively enhance IoT devices' security and functionality. Hybrid models could offer a more robust solution by combining various identification techniques to address these challenges.

4.2.2 IoT Behavioural Profiles

Here, we review the current state-of-the-art IoT behavioural profiles:

Hamza *et al.* [97] focused on enhancing the security and compatibility of IoT devices by developing and verifying MUD profiles. Hamza *et al.* assisted IoT manufacturers in creating these profiles and helped organisations ensure that the devices comply with their policies while tracking network behaviour based on the MUD profiles. Hamza *et al.* proposed a tool that automatically generates MUD profiles by analysing traffic traces from arbitrary IoT devices. The Mudgee tool was tested on 28 consumer IoT devices, providing valuable insights and identifying several challenges. Despite these enhancements, current MUD implementation faces challenges that hinder its widespread adoption and effectiveness. These challenges include managing device diversity, slow adoption rates, and limited compliance from manufacturers [95, 97]. These limitations highlight the need for hybrid models to address these issues by integrating multiple strategies to enhance IoT devices’ security and functionality effectively.

4.2.3 Blocklisting Profiles

Here, we review the current state-of-the-art of blocklisting for identification purposes:

Internet Protocol (IP) blocklists are a common approach to mitigating malicious traffic by blocking known malicious IP addresses. Notable implementations include Pi-hole and Easylist [100, 74]; those lists are widely used in network security. Ramanathan *et al.* [195] proposed an enhanced IP blocklist aggregating multiple blocklist feeds to identify and block attackers more effectively. This aggregation method improves coverage but relies heavily on the accuracy and timeliness of the feeds. Mandalari *et al.* [150] proposed a DNS blocklist method to distinguish between functional and non-functional destinations contacted by IoT devices. While the Mandalari *et al.* method offers improved identification, it is still insufficient to mitigate malicious events fully. A significant limitation in both methods is their reliance on static lists, which attackers can circumvent by utilising legitimate destinations or route traffic through them to perform attack [154]. This limitation highlights the need for hybrid models that can dynamically adapt and incorporate multiple layers of security to address these vulnerabilities effectively.

We reviewed various approaches for IoT device identification in this background, including fingerprinting, behavioural profiles, and blocklisting. We highlighted the need for a fingerprinting model to account for the distinct attributes of IoT devices, including various modes of operation, evolving capabilities (*e.g.*, functionality and updates), and specialised communication protocols, as well as the need for a hybrid model to address their limitations. Next, we will detail our proposed hybrid model to enhance the robustness and accuracy of IoT device identification.

4.3 Hybrid Model Architecture

In this section, we begin by detailing the architecture and phases of our hybrid model, outlining the key components and their roles in achieving IoT device identification across diverse operational contexts. In each phase, we will detail the key findings and recommendations from Chapter 3 that influenced our approach; however, it is important to note that not all recommendations apply to our Hybrid model.

4.3.1 Hybrid Model Overview

As stated at the beginning of this chapter, our goal is to build a robust identification solution for IoT environments, which serves as a foundational step for anomaly detection. Building on the takeaways from Chapter 3, we emphasise the importance of carefully identifying the most relevant environmental features (*i.e.*, mode of operation, transferability, or observability). We first develop a fingerprinting model that focuses on features associated with the mode of operation. This focus addresses the diverse and evolving nature of IoT environments, as IoT devices exhibit distinct patterns across operational modes. Early identification of deviations from these patterns is critical for effective malicious behaviour detection. Because such deviations often serve as the first indicators of potential security attack [126, 223]. Next, we combine this fingerprinting with devices' behavioural profiles and destination blocklists to perform traffic identification. This combination represents a step towards a broader definition of normality within IoT environments while reducing false alarms to a minimum and ensuring scalability [154, 12].

Additionally, following the observability results discussed in Chapter 3, our design of iHybrid aims to support deployment on home gateways. This implementation enables full traffic capture near the user to preserve privacy by avoiding transferring traffic data to external cloud services while allowing for using sampled traffic provided by ISPs in scenarios with limited information. We designed iHybrid to perform sequential traffic validation of received flows from the IoT network. Upon receiving new traffic, the framework executes the following steps:

1. Check whether the traffic has a valid fingerprint.
2. Verify if the traffic originates from a functional destination.
3. Confirm whether the traffic matches the device's valid Access Control List (ACL).

A full cycle of traffic identification begins with recognising the device model via its fingerprint, followed by traffic validation based on functional destination blocklists and behavioural profiles (*e.g.*, MUD profiles). The fingerprinting component generates a probability score indicating the most likely device model, while the behavioural profile and non-functional blocklist components perform binary classification, outputting either a match or mismatch. The validation process results in two outputs:

1. A list of identified flows that adhere to the device ACL and originate from functional destinations.
2. A list of non-identified flows that violate the device ACL or originate from non-functional destinations.

Identified flows can pass through the home gateway, while non-identified flows are blocked. To prevent model drift, iHybrid updates its behavioural profiles, non-functional blocklists, and fingerprints weekly, as Chapter 3 suggests.

In summary, iHybrid is designed for deployment on home gateways to perform online traffic identification. iHybrid has three phases: baseline generation, pre-processing, and identification. We

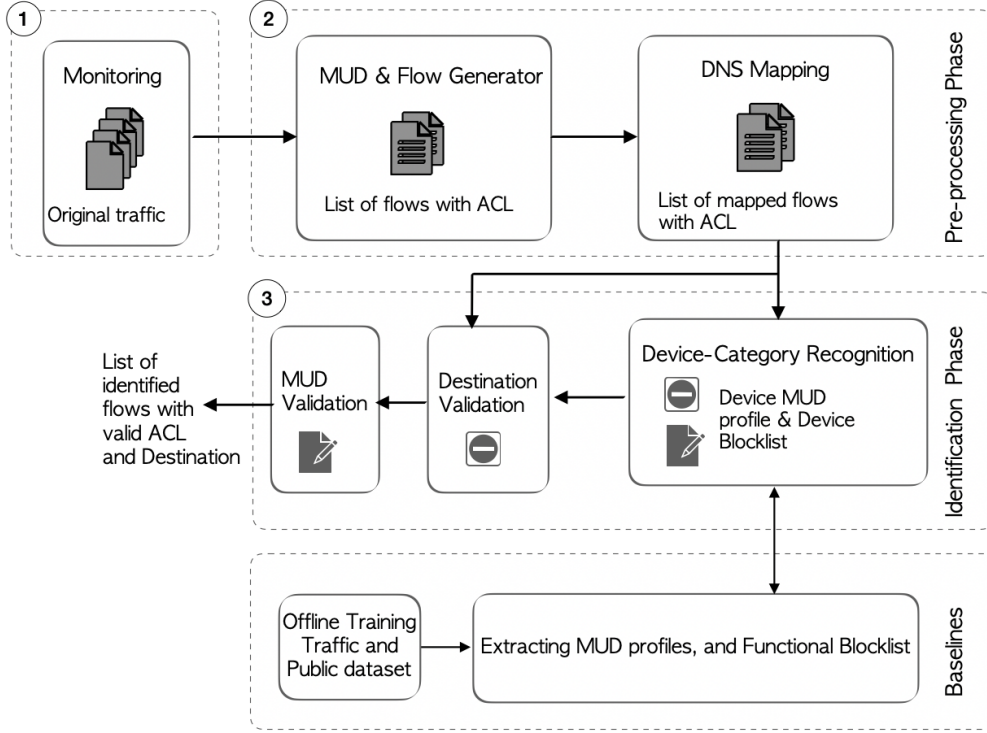


Figure 4.1: **High-Level architecture diagram for iHybrid traffic identification.**

will detail these phases in the subsequent sections. Figure 4.1 presents a high-level architectural diagram of iHybrid.

4.3.2 Baseline Phase

As discussed in Chapter 3, IoT devices exhibit various modes of operation, such as idle, active, setup, and update. Additionally, we observed that even within the same mode, such as idle, devices of the same model might experience varying levels of idleness. We build on this observation, and to establish a robust baseline (ground truth) for calibrating and comparing new traffic, we utilised traffic collected from our testbed alongside publicly available IoT device datasets. This combination allowed us to construct a comprehensive library that captures all possible normal behaviours for IoT devices across different modes of operation. Moreover, we grouped devices from the same vendors with similar functionalities, as these may share common blocklists or ACL requirements. We also generated a Manufacturer Usage Description (MUD) profile specific to each operational mode and created an updated functional blocklist for each device. We followed the recommendation from Chapter 3 regarding frequent model updates; we maintain and update these MUD profiles and blocklists every week to minimise false positives, ensuring they reflect the latest device behaviours and network conditions.

4.3.3 Pre-processing Phase

iHybrid starts with the Pre-processing Phase, which is essential for preparing and processing incoming network traffic. The Pre-processing begins with the monitoring component, which receives as input the network traffic data collected from IoT devices in the network. The monitoring

component’s primary function is to collect raw network packets and group and label these packets based on the MAC address of each device. Then, we fed these packets into the MUD & Flow generator component. The MUD & Flow generator component receives the traffic data the monitoring component collects and processes it to generate a list of network flows.

The decision to use flow data in our model, rather than packet data, is driven by recent research highlighting network flow relevance in addressing challenges such as traffic encryption and privacy preservation, as discussed in Chapter 2 (Section 2.2.6), and we rely on header information to extract meaningful features for the model.

We constructed each flow by aggregating packets based on a 5-tuple consisting of source IP, destination IP, source port, destination port, and protocol. We generated an associated ACL for every flow, which organises these flows for further validation. We extract the ACL using the MUDgee tool [168]. If the Mudgee tool does not extract the associated access list, we leave the access list blank. The output of this step is a structured list of flows, each linked with ACL rules. Then, We passed these flows to the DNS Mapping step, where the input is the list of flows with their ACLs from the MUD & Flow Generator. The DNS Mapping function enhances flow information by associating DNS data with each flow, mapping them to domain names that provide meaningful context. We extract the domain name using DNS responses and Transport Layer Security (TLS) handshakes (via the Server Name Indication field). If these methods do not reveal a name for the destination IP in a flow (*e.g.*, when DNS/SNI information is encrypted or appears outside of our data collection period), we rely on reverse DNS lookups [109]. If none of the above approaches yields a domain name, we leave the IP’s domain name blank. The output of this phase is a refined list of flows with ACLs and mapped DNS destinations detailing where the traffic is directed or originating from.

4.3.4 Identification Phase

This phase identifies the device model and validates device traffic using the processed traffic from the pre-processing phase. The fingerprinting model (*i.e.*, device category recognition) generates a list of flows and passes them to sequential validation. We will explain both the fingerprinting process and the sequential validation in the following sections:

1. Fingerprinting Model

Our goal is to build a model for identifying IoT device models under different modes of operation. Consider using MAC addresses or serial numbers, but these identifiers have significant limitations. During packet exchanges, the MAC addresses of the sender and receiver are unencrypted by the transmitter, making them susceptible to spoofing and privacy issues [154, 203, 127, 146]. As such, relying solely on these identifiers is insufficient [203, 127, 146]. Traffic fingerprinting has emerged as a promising technique for identifying IoT devices and their activities [101, 189, 2, 235].

In a typical home network, a mix of IoT and non-IoT devices exists, such as smartphones, PCs, and laptops. Our model determines whether a flow originates from an IoT or non-IoT device. If it is an IoT device, the model identifies the specific device model (*e.g.*, Amazon Echo Show,

TP-Link LightBulb). The pseudo-code for this fingerprinting IoT device model identification is shown in Algorithm 1. The fingerprinting process consists of two phases:

1. **Binary Classification** – In this phase, we distinguish between IoT and non-IoT devices.
2. **Multi-Classification** – This phase uses three distinct Random Forest models, each trained on device data specific to one of the modes of operation (setup, idle, and active).

We use Random Forest (RF) as the model due to its robustness against model degradation and overfitting, as highlighted in the practical evaluation results in Chapter 3. Based on results from Section 3.4.5 in Chapter 3, we trained the Random Forest model for binary classification using the following tailboard features: *source port, destination port, second level domain entropy, sleep time, flow volume, TCP flags, DNS queries, cipher-suites, maximum inter-arrival time, TLS handshaking*. These features enable us to distinguish between IoT and non-IoT devices. For the multi-classification phase, we used the features that performed best across different modes of operation (0.91): sleep time, SLD entropy, and destination port.

For each incoming flow, predictions are made by all three Random Forest models, each trained for a specific mode of operation: setup, idle, and active. We determined the final classification based on the majority voting. If at least two models agree on the same device model, that model is selected as the identified device model. For example, if both Model_Setup and Model_Active predict "Device A" while Model_Idle predicts "Device B", "Device A" is chosen due to the majority vote. In cases with no majority, such as when each model predicts a different device model, the prediction from Model_idle is selected. We built this voting based on the assumption that the idle mode provides the most reliable indication of the device model (see Chapter 3, as it involves more extended periods that better capture the device's behaviour. This specialisation allows each model to excel in identifying device models based on mode-specific characteristics.

We performed hyper-parameter tuning using grid search for each RF, finding the optimised number of trees and maximum depth to enhance identification performance. Additionally, we performed cross-validation for each RF to evaluate model performance and ensure generalisation to unseen data. Also, following chapter 3 recommendations, we weekly update the Random Forest models to incorporate new data and adjust to evolving device behaviours. This update includes retraining models weekly with the latest data to improve accuracy.

The identified device model is outputted from the classification process, providing the most reliable prediction based on aggregated model outputs. We used the device model identification results to retrieve the associated MUD profile baseline and the functional blocklist for each IoT device.

2. Functional Destination Validation.

IoT devices frequently contact various destinations [196, 154]. These destinations can be categorised into three main types: those necessary for the device's functionality, those providing outsourced computing resources, and those related to advertising and analytics [196]. Attackers can exploit advertising and analytics destinations to misuse IoT devices for launching cyberat-

Algorithm 1 Fingerprinting Model

Require: Network traffic data

Ensure: Identified device model (IoT) or non-IoT

```
1: Initialise IoT Classifier
2:  IoT_Classifier ← TrainIoT_Classifier(data)
3: Initialize Random Forest models for each mode:
4:  Model_Setup ← TrainRandomForest(data_setup)
5:  Model_Idle ← TrainRandomForest(data_idle)
6:  Model_Active ← TrainRandomForest(data_active)
7: for each flow in network traffic do
8:   Determine if flow is IoT:
9:   is_IoT ← IoT_Classifier.predict(flow)
10:  if is_IoT then
11:   Determine current mode (setup, idle, active)
12:   Obtain predictions from each model:
13:   pred_setup ← Model_Setup.predict(flow)
14:   pred_idle ← Model_Idle.predict(flow)
15:   pred_active ← Model_Active.predict(flow)
16:   Perform majority voting:
17:   preds ← {pred_setup, pred_idle, pred_active}
18:   device_model ← majorityVote(preds)
19:   Output device_model as the identified model
20: else
21:   Output "Non-IoT device"
22: end if
23: end for
```

tacks [154].

While several IoT security frameworks have implemented blocklisting models such as Pi-hole and Easylist [100, 74], these models often do not consider the specific functionalities of IoT devices when blocking traffic. Prior research by Mandalari *et al.* [151, 150] showed that we can further categorise that destination into:

1. Functional Destinations: these are necessary for maintaining the standard functionality of an IoT device. Blocking these destinations would disrupt the device's intended operations [151, 150].
2. Non-Functional Destinations: these do not affect the device's core functionality when blocked. However, traffic from non-functional destinations can pose a security risk, as it contribute to malicious activities [151, 150].

We aim to reduce network traffic under inspection for malicious events by prohibiting flows that do not adhere to the device's specified functional destinations. Blocking these non-functional destinations can significantly reduce the attack surface in IoT environments [196, 154, 126]. In our model, we utilise a predefined blocklist to validate traffic. The blocklist specifies non-functional destinations for each IoT device, helping to validate whether the flow is directed toward or coming from a functional destination. This validation provides an additional layer of security by ensuring that flows from non-functional destinations are blocked, thereby preventing potential security

threats. We implement this functionality using *IoTrim*, an open-source functionality-based blocklist [151, 150]. We integrated the blocklist into the traffic validation process. Each incoming flow is validated to check if it is directed toward or coming from a functional destination, in which case it is permitted to pass through the gateway. Conversely, flows directed toward or coming from a non-functional destination are blocked. This validation process outputs a list of flows directed toward or coming from functional destinations, which will be further investigated in the MUD validation phase.

3. Behavioral Profile Validation

A Manufacturer Usage Description (MUD) profile defines the intended communication patterns for IoT devices when installed in a network [106]. The MUD profile acts as an Access Control List (ACL) for each device, specifying which network actions are permissible for the device to function as intended. A device exhibiting behaviour deviating from its MUD profile may indicate a potential malicious activity.

We aim to reduce network traffic under inspection for malicious events by prohibiting flows that do not adhere to the device’s specified ACL. Prior research by Hamza *et al.* [96] has demonstrated the effectiveness of MUD profiles in detecting volumetric attacks, highlighting their potential to identify malicious network behaviour.

While MUD profiles offer valuable insights into device behaviour, there are limitations in their implementation, as discussed in Chapter 3. For example, the accuracy of the MUD profile may vary depending on how well it reflects the actual communication patterns of IoT devices in dynamic environments. Despite these challenges, MUD remains a valid concept for defining malicious-free communication patterns. It complements the fingerprinting model by providing an additional validation layer, ensuring traffic adheres to known good behaviours for each device. In our model, we validate the ACL associated with each device against the list of permitted flows obtained through functional destination validation. This validation process generates two outputs:

1. Identified Flows, which match the valid ACL, indicating that the device behaves as expected
2. Non-Identified Flows, which do not match the valid ACL and signal potentially unauthorised activity.

We investigated the identified flows for anomalies (as detailed in Chapter 5) while we blocked the non-identified flows to prevent potential security breaches. We used Adhoc Overlay Networks (AON) to block unwanted traffic, following prior work by Hafeez *et al.* [94] to adopt such a method. AONs create a virtual, logical network layer over the physical network. This layer controls traffic flows, enabling features such as traffic filtering, routing, and blocking without requiring changes to the underlying physical network infrastructure [94]. In our validation phase, an AON was a lightweight, customisable way to ensure that only validated flows pass through the network [94].

Integrating multiple components within our hybrid model ensures robustness by leveraging the strengths of each element to mitigate potential failures. For example, suppose one component,

such as the fingerprinting model, fails to identify a device due to incomplete data correctly. In that case, the validation steps using the functional destination blocklist and the Manufacturer Usage Description (MUD) profile provide additional layers of verification to support accurate identification. In the event of a discrepancy between these components, the majority voting mechanism ensures that the most likely device model is selected, reducing the risk of errors. This collaborative approach strengthens the robustness of the IoT device identification process and provides a solid foundation for classifying malicious traffic.

By combining device model identification, behavioural profiling, and destination validation, the hybrid model reduces the amount of traffic under inspection by early prohibiting traffic that does not adhere to the behavioural profile and blocklisting, thereby filtering out potentially malicious traffic before it reaches further stages of analysis. In the next section, we will investigate the percentage of traffic under investigation resulting from this combination and compare the effects of integrating these identifications through an ablation study.

4.4 Results

In this section, we outline the experimental setup and data collection process, followed by a detailed analysis of the evaluation results for our iHybrid model. We present the percentage of traffic under investigation resulting from device identification, behavioural profiling, and destination validation. Additionally, we compare the effects of integrating these components through an ablation study and benchmark our model against the state of the art.

4.4.1 Experimental Setup

In this section, we describe the experimental setup, including the testing environment, implementation of the model phases, and performance metrics.

Environment. We deployed the model on Raspberry Pi 4 Model B running Linux-OS 10 (buster). The Raspberry Pi has 8 GB RAM and a 64-bit quad-core Cortex-A72 processor. We implemented all hybrid model phases in Python 3.9. We ran the model as a background process and automatically performs detection on an one minutes basis.

Performance Metrics. The distribution of IoT and non-IoT devices flows in our datasets are unequal (*i.e.*, imbalanced dataset). Having an imbalanced dataset leads to an imbalanced classification problem [221]. Maximising accuracy is not a suitable performance metric to use in evaluating an imbalanced classification problem [221], however, metrics such as True Positive Rate (TPR) and False Positive Rate (FPR) are more suitable for evaluating such an imbalanced classification problem [221]. In both binary classification (IoT and non-IoT device) and multi-classification (different IoT devices models), to evaluate the performance of our framework, we measured the proportion of actual positive instances (instances belonging to a class) that are correctly identified by the model, TPR (recall). FPR measured the proportion of negative cases (cases not belonging to a class) that are incorrectly identified as belonging to that class. The F1-score is the harmonic mean of Precision and Recall, providing a balance between them. Consider the sum

of True Positives, False Positives, and False Negatives for all classes and compute the global F1-score. AUCPR is particularly useful in imbalanced datasets as it focuses on the performance of the positive class. For multi-classification, we aggregate the contributions of all classes to compute a Micro-Averaging across four metrics. For example, to calculate AUCPR, aggregate the Precision and Recall values across all classes and then compute the area under the curve.

4.4.2 Dataset Collection and Labelling

Collected Dataset. The data collection phase consists of two testbeds we build on our own. Our testbeds cover 90 IoT devices with IP connectivity, including 75 from the USA and 15 from UK stores. Our testbeds incorporate non-IoT devices, such as tablets, smartphones, and laptops. The complete device list is in Table 3.4. We selected devices covering a wide range of functionalities. Our selection criteria included price, popularity, and customer ratings, mirroring the factors an average consumer considers when buying an IoT device [57, 189, 203]. To ensure availability, we prioritised well-known IoT vendors.

For our data collection setup, we deployed *tcpdump* on a Raspberry Pi 4 Model B, which serves as a gateway, and *hostapd* is used for this purpose [90]. *tcpdump* captures our setup’s incoming and outgoing Wi-Fi traffic from 90 devices. We developed a script to automate the process of data collection and storage. To distinguish and isolate each device’s traffic from the heterogeneous network mix, we employed the device’s MAC (Media Access Control) address as a unique identifier.

Dataset Partitioning. We divided the collected dataset into (1) **Baselines Set:** we require traffic to establish our baselines (MUD profiles, blocklisting) to be used to calibrate the results. We performed baseline generation using passive traffic. We utilised 21 days spanned over different device modes of operation. We assumed that 21 days is sufficient for most of the different pattern behaviours of IoT devices in an environment. We generated three MUD profiles for all IoT devices in three different modes of operation. We downloaded most of the updated functional blocklisting for all devices. (2) **Training and Validating Sets:** we used this subset to train and validate the identification phases, device model recognition, MUD validation, and functional destination validation. The percentage of device samples in this set was 25%. (3) **Testing Set:** we used this subset of the dataset to evaluate the final performance of the model after it has been trained and validated. The percentage of device model samples in this set was 75%. We ensured that the test set contained new and unknown IoT device traffic to provide an unbiased evaluation of the model’s ability to detect new, unseen samples.

4.4.3 Fingerprinting Model Results

In this section, we present the fingerprint model (device recognition) phase results; the fingerprinting has two main steps: (1) binary classification, where the classification is to differentiate between IoT and non-IoT devices. (2) Multi-class classification results, where the task is to recognise specific IoT device models.

Detection level	Binary-classification		Multi-classification	
Metrics	UK	USA	UK	USA
TPR	0.97	0.93	0.81	0.86
FPR	0.01	0.01	0.05	0.01
F1-score	0.96	0.92	0.80	0.90
AUCPR	0.99	0.97	0.85	0.95

Table 4.1: **Detection performance of device model recognition phase, binary classification represents IoT vs non-IoT classification, while multicalssification represents the device mode recognition in IoT devices.**

Distinguishing between IoT and non-IoT Devices

Table 4.1 presents the performance of a device model recognition system in binary classification, where the classification is to differentiate between IoT and non-IoT devices. In the binary classification results, the model shows exceptional performance across both the UK and the USA. For the UK, the TPR is 0.97, indicating that 97% of IoT devices were correctly identified, while the FPR is a low 0.01, showing that only 1% of non-IoT devices were misclassified. The F1-score of 0.96 reflects a high level of accuracy and balance between precision and recall, and the AUCPR is almost perfect at 0.99, demonstrating excellent performance in distinguishing IoT from non-IoT devices. Similarly, in the USA, the TPR is slightly lower at 0.93, but the FPR remains consistent at 0.01. The F1-score is 0.92, and the AUCPR is 0.97, both slightly lower than the UK but still indicative of strong overall performance.

IoT Devices Identification

In the multi-class classification results, where the task is to recognise specific IoT device models, For Table 4.1, we can see that the model's performance shows some variation between the UK and the USA. In the UK, the TPR is 0.81, meaning 81% of IoT device models were correctly identified, while the FPR is 0.05, indicating a 5% misclassification rate among IoT devices. The F1-score is 0.80, reflecting a reasonable balance between precision and recall, though lower than in binary classification. The AUCPR is 0.85, suggesting good, but not outstanding, performance in recognising different device models. In contrast, the results for the USA show a higher TPR of 0.86, with a significantly lower FPR of 0.01, indicating fewer misclassifications. The F1-score is higher at 0.90, pointing to better accuracy and consistency in this region, and the AUCPR of 0.95 further confirms the model's strong ability to distinguish among various IoT device models in the USA.

Comparison with Existing Identification Solutions

Table 4.2 presents the performance comparison between our fingerprinting and the state-of-the-art IoT identification solutions. With a TPR of 0.93, Our fingerprinting closely equals the highest-performing model, Pinh19 (0.95), and surpasses other models such as Meid20 (0.92) and Okui22 (0.81). Our fingerprinting maintains an impressively low FPR of 0.01, matching the best-performing

Paper	TPR	FPR	F1-score	AUCPR
Meid20 [156]	0.92	0.01	0.94	0.88
Okui22 [180]	0.81	0.02	0.84	0.90
Siva18 [220]	0.67	0.02	0.64	0.87
Pinh19 [192]	0.95	0.02	0.94	0.89
Yang19 [253]	0.56	0.01	0.68	0.67
Ortiz19 [183]	0.56	0.01	0.68	0.67
Dong20 [69]	0.56	0.01	0.68	0.67
Fan22 [77]	0.56	0.01	0.68	0.67
Perd20 [189]	0.75	0.08	0.73	0.80
Ours (iHybrid)	0.93	0.01	0.93	0.97

Table 4.2: **Performance comparison between our fingerprinting and the IoT identification state of the art.**

models and outperforming others such as Pinh19 (0.02) and Perd20 (0.08). In terms of the F1-score, Our fingerprinting achieves 0.93, nearly equaling Pinh19 and Meid20 (both at 0.94) and significantly outperforming models such as Okui22 (0.84) and Perd20 (0.73). Most notably, our fingerprinting excels in the AUCPR metric, achieving a score of 0.97, substantially higher than all other models, indicating superior precision-recall trade-off and overall robustness in IoT device identification.

We evaluate multiple metrics to determine the best model for IoT device identification rather than relying on a single metric, especially when dealing with imbalanced data. For example, while the highest TPR might suggest a model performs well in identifying positive cases, it does not fully capture the model’s ability to minimise FPR or maintain a balanced precision-recall trade-off. Our fingerprinting, with an AUCPR of 0.97, demonstrates superior overall performance, achieving a TPR of 0.93 and a low FPR of 0.01, showing robustness across all critical metrics. Although Pinh19 has a slightly higher TPR (0.95), its FPR (0.02) is higher than our fingerprinting’s, which impacts its overall effectiveness. By considering multiple metrics like AUCPR, TPR, and FPR together, we can confidently conclude that our fingerprinting outperforms others in balancing accuracy and precision while effectively handling imbalanced data.

4.4.4 Traffic Identification and Validation Results

In this set of experiments, we evaluated how well the full cycle (fingerprinting, destination validation, and MUD profile) validates a flow. Thus, we compared the received traffic with the baseline traffic we produced in the data collection phase (see Section 4.4.2). In particular, we calculated the number of flows identified as normal when they are indeed normal, matching the MUD ACL and coming from a functional destination. We performed this validation for all devices in both the UK and USA datasets. We filtered the Address Resolution Protocol (ARP), broadcast, Internet Control Message Protocol (ICMP) port unreachable and local network packets (these packets formed 10%

of total captured traffic) before calculating the percentage of the identified traffic. The percentage of the identified traffic as normal compared with the total number of captured traffic during the experiments was between 80 – 85%. The remaining traffic between 5% – 10% consisted of packets that had never been seen before, which may have anomalous and/or malicious traffic. The model marked them as not verified and blocked them.

In both testbeds, iHybrid detected regular packets when normal, with no false positives and an accuracy of 97.8%. Despite this level of accuracy, when subjecting iHybrid to traffic coming from cyberattacks such as DDoS or port scanning, the identified traffic yet some of these flows were marked as normal while they were infected flows, while some of the infected flow the model marked them as had never been seen before and marked as no verified. This identification percentage was arbitrary, depending on the device or the attack performed. For example, when subjecting Amazon Echo Show to Port scanning, 5% of total infected traffic was marked as unverified while TP-Link Lightbulb 1% of infected were marked as unverified. This is because the port scanning follows a similar ACL in the normal MUD profile of the Amazon Echo Show and TP-LINK Lightbulb. These results urge further investigation of the verified normal traffic, which might mimic a normal ACL or come from a valid destination.

Ablation Setup	TPR	FPR	F1-score	AUCPR
MUD	0.56	0.46	0.00	0.50
IoTrim	0.67	0.38	0.75	0.78
MUD + IoTrim	0.67	0.38	0.75	0.78
iHybrid	0.93	0.01	0.93	0.97

Table 4.3: **Ablation study comparing different setup: MUD, IoTrim, MUD + IoTrim, and iHybrid.**

4.4.5 Ablation Study Results

Figure 4.2 compares different detection setups (without identification model, IoTrim, MUD, and Hybrid Detection) based on their traffic classification into distinct categories: Normal, Essential, Non-Essential, and Anomaly traffic. Figure 4.2 shows that without an identification model (standalone), it does not categorise non-essential or other normal traffic. IoTrim mirrors the standalone approach in performance but seems to have no additional granular classification. While MUD introduces the classification of Non-Essential Traffic, its ability to identify Essential traffic decreases. With MUD 67%. Essential traffic is identified correctly and is lower than IoTrim and standalone setups. Hybrid Detection achieves the most detailed categorisation, correctly identifying 66% Essential traffic, 34% Non-Essential traffic, and maintaining 4% Anomaly detection accuracy. However, splitting Normal traffic into 65% misclassified as Non-Essential and only 30% correctly classified as Normal highlights potential misclassification challenges.

Table 4.2 presents the results of an ablation study, comparing the performance of different setups (MUD, IoTrim, MUD + IoTrim, and iHybrid). Table 4.2 shows that MUD alone demonstrates limited capability (AUCPR 0.50), with high false positives significantly impacting its performance

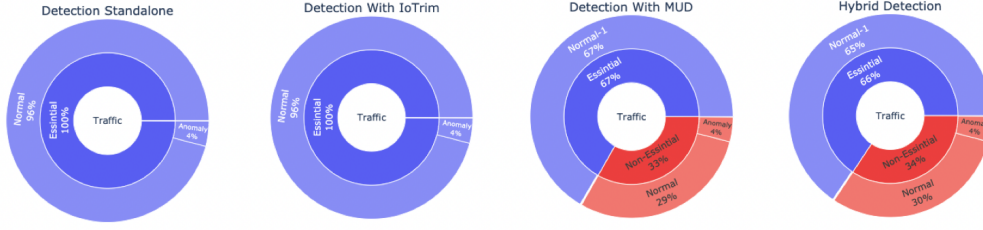


Figure 4.2: **Percentage of traffic under detection comparing different setup: MUD, IoTrim, MUD + IoTrim, and iHybrid.**

(FPR 0.46). IoTrim shows considerable enhancements over MUD regarding precision-recall balance and robustness, as reflected by F1-score (0.75) and AUCPR (0.78). Combining MUD with IoTrim does not yield any additional improvement over IoTrim alone. The combination is identical to IoTrim: TPR (0.67), FPR (0.38), F1-score (0.75), AUCPR (0.78). iHybrid demonstrates a superior performance across all metrics, significantly outperforming the other setups. The low FPR 0.01 highlights its robustness against misclassification, and the high TPR 0.93, F1-score 0.93, and AUCPR 0.97 suggest it is highly reliable for traffic identification.

This results section has presented the evaluation results of iHybrid. Next, we provide takeaways and lessons to enhance IoT device identification solutions and malicious traffic identification.

4.5 Discussion

In this section, we will detail the key takeaways and lessons from our results while also highlighting the limitations of our evaluation.

4.5.1 Takeaways and Lessons

Here, we present the key takeaways and lessons learned from our approach evaluation, highlighting the most important insights gained throughout the evaluation process:

- Hybrid models integrate multiple sources of normality to enhance their ability to detect abnormalities. Each source represents a different dimension of normal behaviour, such as patterns in network traffic, behavioural profiles, or functional services. Combining these diverse perspectives gives the model a more robust and multidimensional understanding of normal behaviour. This comprehensive view enables the solution to identify better subtle, complex, or rare deviations that may signal harmful or malicious activities. Defining normal behaviour is crucial, and incorporating multiple sources of normality improves the model's ability to detect abnormal behaviour, especially in malicious events.
- Considering the different modes of operation for each device can improve the detection and identification of the device model. While different IoT devices may share specific operational characteristics, they can differ significantly in others depending on their mode of operation. For example, two smart Lightbulbs might show similar power consumption patterns in a basic on/off mode, but their behaviour could vary in dimming mode or when

responding to different network protocols. By training the classification model on data from various modes of operation, it becomes better at recognising these subtle differences, ultimately improving accuracy in identifying the specific device model.

4.5.2 Limitations

Here, we discuss the limitations of our hybrid model, acknowledging factors that may have influenced the results and suggesting areas for improvement in future research:

- We can use our framework for Wi-Fi-connected devices or any devices that can route their traffic via a Wi-Fi hub or IP gateway. In future work, we will examine whether other wireless communications standards like Bluetooth [169], Z-Wave [256], or other IEEE 802.15.4-compliant RF standards make any difference. We expect to deploy iHybrid provided it can (a) monitor devices' packets when transmitting and receiving Internet data and (b) distinguish plain text header data from encrypted payload (or higher layer) data.
- The implementation of *MUDgee* introduces commensurate limitations [96], particularly noticeable in the time required to generate a MUD profile. Sometimes, it took *MUDgee* 30 to 40 minutes or up to several hours. Leaving the identification framework to wait until the MUD profile is generated will affect the overall identification time and invalidate the design requirements to maintain low-latency identification. We solved this by skipping the MUD generation step in cases of generation times exceeding 10 minutes. Performing identification without MUD profile validation may affect identification performance and potentially increase false negative or false positive rates. A possible solution is adopting a different implementation for behavioural profiles or a different implementation of MUD if available.
- We found some cases where we mixed detect IoT devices that shared manufacturer/functionalities. For example, we found that the Amazon Echo Show is classified as a Yi Camera rather than an Amazon Echo Show; the same case is true with the TP-Link Lightbulb and TP-Link Plug. A possible solution would be adopting a separate model for Shared manufacturer/functionalities in advance similar to IoT non-IoT phase, then feed to the multi-model for device category. If the IoT devices change their backend infrastructure, *e.g.*, after an update, we may have to update our identification MUD profile and blocklisting, too.

4.6 Conclusion

In this chapter, we build on Chapter 3 recommendations to develop a fingerprinting model that focuses on features related to the mode of operation for IoT environments. We then combine the fingerprinting model with behavioural profile and blocklisting to ensure robust identification of what normal behaviour of IoT devices should follow. We showed how this combination validated up to 80% of incoming traffic. We showed how this combination encountered challenges when attackers mimicked valid behaviours, rendering some flows indistinguishable from normal traffic. In the next chapter, we will delve deeper into the 80% of incoming verified flows, using a detection model to examine the identified traffic further and identify any malicious behaviour.

Chapter 5

iDetector an Anomaly Detection for IoT Environments on Edge Devices

Associated Publication: **Eman Maali**, Julie McCann, “**Privacy-Preserving IoT Malicious Traffic Classification: SAX Encoding and Dissimilarity Compression on Flow-Based Features**”, *Privacy Enhancing Technologies Symposium (PoPETS)*, Washington (USA), July 2025. (Under review).

Prof. Haddadi and Dr. Boyle supervised part of this chapter. I acknowledge their fruitful discussions and advice.

5.1 Introduction

In Chapter 2, we identified a critical gap: the role of robust and accurate device identification as a prerequisite for improving IoT anomaly detection. In Chapter 3, we examined how machine learning-based IoT device identification models degrade over time and provided recommendations for improvement. Building on these recommendations, in Chapter 4 we introduced IoT device fingerprints based on modes of operation and developed the hybrid traffic identification model, iHybrid, which combined these fingerprints with behavioural profiles (MUDs), Access Control Lists (ACLs), and functional destination blocklisting to validate incoming traffic. In Chapter 4, we showed that iHybrid successfully validated up to 80% of incoming traffic. However, we are uncertain whether 80% of the traffic is truly free of malicious activity or if any verified traffic is mimicked legitimate behaviours. To address this, we focus on the 80% of verified traffic in this chapter. We aim to propose a detection model to examine the verified traffic further and identify any malicious behaviour.

Contributions. The main contributions of this chapter are summarised as follows:

- iHybrid effectively identifies valid IoT devices but cannot identify attacks that mimic valid behaviour. To solve this problem, we present iDetector, a malicious traffic detection solution that uses Symbolic Aggregate Approximation (SAX) and compression-based dissimilarity operating in unsupervised mode to solve this problem. We develop a theoretical analysis

framework to prove the properties of iDetector, explicitly focusing on the accuracy of traffic pattern extraction. We demonstrate that SAX-encoding accurately extracts data patterns by proving that the information loss in SAX-encoding over collected traffic is approximately zero. This validation supports the robustness and effectiveness of iDetector, ensuring that our approach can efficiently detect malicious traffic while maintaining minimal information loss, reinforcing the system’s reliability and performance.

- SAX-encoding and Compression-based dissimilarity facilitate the transition from offline to online detection by representing high-dimensional time-series data as simple patterns, directly addressing the challenge of high processing overhead. We show how SAX-encoding can represent different traffic patterns efficiently, and the Compression-based dissimilarity enables lightweight, online traffic classification near the user, such as at home gateways or routers. SAX-encoding and Compression-based dissimilarity (iDetector) directly address the challenge of predefined anomalies and evolving attack patterns, enabling it to detect both seen and unseen malicious traffic. We demonstrate the effectiveness of compression-based dissimilarity using publicly available datasets and replicating various cyberattack scenarios, showing its scalability and low false-positive rates.

5.2 Background

Some papers discussed here overlap with those in Chapter 2. However, in Chapter 2, we reviewed anomaly and malicious detection solutions for IoT environments, highlighting key challenges in the domain. In this chapter, we focus exclusively on techniques for identifying malicious IoT traffic to understand the design requirements for a detection model capable of identifying malicious traffic that mimics valid behaviours.

An enormous amount of works addressed the malicious IoT traffic detection [71, 155, 158, 96, 152, 174, 168, 39, 198, 15, 39, 198, 163, 49, 94, 84, 269, 30, 73, 234, 211, 204, 205, 261, 9, 149, 224, 37, 82, 79]. We reviewed these papers based on the number of citations, visibility of the conference in which they appear, application, and seminal work (*i.e.*, the first to use an algorithm and/or feature set). Lastly, we categorised these papers by technique, covering rule-based methods [15, 82, 84, 96], and ML-based methods [174, 158, 39, 155, 163, 71, 224, 269, 94, 70]. Many of these papers are ML-based (12 out of 15). For each paper, we documented the methodology and results. We categorised the methods to identify common ML algorithms. Overall, we selected 15 recent papers representative of IoT malicious detection. We summarised 15 papers in Table 5.1, comparing network data, algorithms, feature types, dimensionality, targeted attacks, online detection, and evaluation strategies. Many papers used flow-based network data, likely due to the prevalence of encrypted traffic and the overhead of packet-level inspection [44]. Deep-learning techniques were commonly favoured, relying on header information such as packet size, timing, and statistical features. Numerical features were the most widely used, derived from measurable quantities such as packet sizes and durations. Feature dimensionality varied, with papers using dimensions such as 1, 7, 11, or 23 features. Eight papers specifically targeted attack types such as port scanning, DDoS, and botnets, while others did not specify attacks. Only five papers supported

online detection. Finally, 10 out of 15 papers validated their approaches empirically using public datasets.

To further understand the specific approaches used in these studies, we now detail the methodologies and findings of the papers summarised in Table 5.1. We begin by detailing rule-based methods. Fu *et al.* [84] proposed Whisper, an online identification model, the first to use frequency domain features in statistical clustering. Fu *et al.* showed they can perform online identification with high detection accuracy and throughput. However, Fu *et al.* envisioned Whisper as a middle-box framework (*i.e.*, Firewalls, load balancing), which may not be feasible for constrained traffic collection where detection latency might be hours. Fu *et al.* [82] proposed HAWatcher as explainable hypothetical correlations based on semantics on different channels (smart-app, physical, and user-activity). HAWatcher experienced false alarms when there were user-activity deviations. Hamza *et al.* [96] proposed an SDN-based system empowered by machine learning to monitor and learn the behavioural pattern of MUD rules to detect volumetric attacks. The SDN-based system showed promising results in volumetric detection. However, the implementation of SDN has challenges in system scalability and handling data generated by IoT devices.

We now detail machine learning approaches. Nguyen *et al.* [174] proposed DIOT autonomous self-learning identifies compromised IoT devices in botnets locally based on the Audi approach [152] to detect volumetric attacks. Although the DIOT showed promising results in detecting Mirai malware, the number of included devices during the evaluation was only 33 consumer IoT devices. Meidan *et al.* [158] and McDermott *et al.* [155] deployed deep-learning techniques to detect botnets in the IoT ecosystem; however, these models have been evaluated in a simulated environment and have yet been tested to detect other types of attacks. Brun *et al.* [39] proposed a dense random neural networks approach to detect network attacks such as DDoS and Botnets. Although attacks were successfully detected, Brun *et al.* used only a testbed of three devices for evaluation.

Mirsky *et al.* [163] used statistical features to measure the behaviour of the network, such as the mean or variance of the packet size or the correlation coefficient. Mirsky *et al.* model deployment was offline because model training/updating latency is high over large datasets. Thus, it could not perform online identification. Doshi *et al.* [71] employed ensemble machine learning models to detect DDoS attacks. Doshi *et al.* results proved that focusing on IoT-specific network behaviours to inform feature selection increases the accuracy of volumetric detection in IoT network traffic, but their experiment focused on DDoS attacks.

Sriram *et al.* [224] focused on detecting IoT botnet attacks using a deep neural network (DNN) model with four hidden layers and uses Rectified Linear Unit (ReLU) activation functions, dropout regularisation, and the Adam optimiser. Sriram *et al.* used two primary datasets for evaluation N-BaIoT and BoT-IoT, which are transformed into different subsets to aid in classification tasks such as detecting normal versus attack traffic, distinguishing between different only botnet families (BASHLITE and Mirai), and further categorising each botnet type.

Li *et al.* [131] proposed ADRIOT, an anomaly detection framework for IoT networks that uses edge computing to identify potential threats. ADRIOT includes a traffic capturer, traffic preprocessor, and multiple anomaly detectors tailored to different device types. Each detector is based on

Category	Network Data	Paper	IoT Specific	Machine Learning Algorithm	Feature Type	Features Dimensionality	Targeted Attack(s)	Online Detection	Evaluation Strategy
Rule-based	Flow Level	Fu21 [82]	✓		Semantic				Empirical (54 devices)
		FuLi21[84]			Frequency			✓	Empirical
		Hamz19 [96]	✓		Semantic			✓	Empirical (33 devices)
Learning-based	Packet-level	Nguy18 [174]	✓	Federated	Numerical	7	✓		Empirical
		Meid18 [158]	✓	Deep autoencoders	Numerical	23	✓		Empirical (9 devices)
		Brun18 [39]	✓	RNN			✓	✓	Empirical (3 devices)
		McDe18 [155]	✓	BLSTM-RNN	semantic		✓		Simulation
		Mirs18 [163]	✓	Deep autoencoders	Numerical	23		✓	Empirical (9 devices)
		Dosh18 [71]	✓	Ensemble	Numerical	11	✓		Simulation
		Srir20 [224]	✓	LSTM			✓		Simulation
		Li21 [131]	✓	LSTM					Simulation
	Flow-level	Zong18 [269]		Deep autoencoders		1	✓		Simulation
		Baha19 [30]	✓	Autoencoders and PCA			✓		Simulation
		Hafe20 [94]	✓	Fuzzy C-means	Numerical	38		✓	Empirical (8 devices)
		Dong23 [70]	✓	iFrost	Semantic				Empirical (26 devices)

Table 5.1: **The qualitative comparison for the most recent malicious IoT traffic detection. Each column corresponds to a discussion of the comparison aspect of related work.**

an LSTM autoencoder and operates unsupervised, meaning it does not require labelled attack data. This design enabled ADRIOT to detect emerging zero-day attacks effectively, making it adaptable to new and previously unknown threats. Although Li *et al.* has designed ADRIOT to detect zero-day attacks, ADRIOT’s capacity to adapt to entirely new attack strategies might be constrained by the variety of normal traffic patterns it has been trained on. Additionally, the high computational demands of LSTM Autoencoders can cause delays in networks with high data throughput, potentially hindering ADRIOT’s online detection capabilities.

Zong *et al.* [269] presented a novel approach for anomaly detection using a deep Autoencoder framework combined with a Gaussian mixture model (GMM). The method is designed for unsupervised learning, meaning it does not require labelled data, which is crucial for detecting rare or unseen anomalies. Zong *et al.* method relies heavily on having a well-represented set of normal data for training the Autoencoder and GMM. The model only detects anomalies effectively if the normal data is complete and representative. Additionally, the deep Autoencoder requires significant computational resources, particularly when working with high-dimensional data, which limits its scalability and performance in online applications.

Hafeez *et al.* [94] have proposed IoT-KEEPER that used a combination of fuzzy C-means clustering and fuzzy interpolation. With this combination, they succeed in performing online detection and identification of a malicious event. However, IoT-KEEPER cannot deal with encrypted traffic. Dong *et al.* [70] proposed a rule generation algorithm for iForest and a new flow feature extraction scheme; however, simple statistical features collected by network operators, such as the mean or variance of the packet size, can not be tracked in the programmable switches, since floating-point arithmetic is not supported.

This background review highlighted that while the current state of the art often favoured deep learning algorithms for detecting malicious traffic in IoT environments due to their strong performance, they come at the cost of high computational requirements and offline operation. Additionally, these approaches typically focus on detecting specific attack types, which can limit their ability to handle the fast-evolving attacks in IoT environments. As these network attacks evolve rapidly and exploit vulnerabilities in connected IoT devices, detection solutions must be capable of quickly identifying new malicious events and adapting to emerging attack patterns. A malicious

traffic detection solution should provide online detection, preserve user privacy, and minimise computational demands. Moreover, these solutions must be adaptive enough to handle diverse IoT environments while effectively detecting known and unknown malicious events without previous information, significant delays, or resource consumption.

Before outlining our approach's design goals and phases, we will detail the threat model considered throughout our malicious traffic detection methodology.

5.3 Threat Model

Before detailing our approach design, in this section, we detail the threat model considered throughout our malicious traffic detection methodology:

IoT devices may have unpatched holes, default passwords, or poor security mechanisms due to low cost and low capability, making them easier to exploit than routers and gateways. We assume that any Internet-connected device is susceptible to remote attacks launched by firmware or setting updates delivered over the Internet. For our model, the gateway is not included in the attacks, as it is often equipped with more robust security mechanisms and is generally managed by a centralised entity. However, IoT devices may still have vulnerabilities that make them the primary target.

Additionally, if an attacker is within the wireless or physical range of the devices, these can be reprogrammed or compromised through side-channel attacks. We assume a local attacker can passively monitor and capture traffic from all devices within range. However, capturing traffic from low-energy-based devices may require resources not typically available in a home-based environment.

In all cases, the attacker knows the type of the targeted IoT device(s). The attacker's target may also be any device that uses default settings and is on most of the time. An attacker's goals may be one or both: (1) substantially obtain users' personal/sensitive information, or (2) obtain control over one or more IoT devices from which to launch large-scale attacks on local or Internet-connected services or systems. We assume that our detection solution has yet to gain prior knowledge of the performed attack and that all packet contents are appropriately encrypted.

In this section, we presented the threat model considered throughout our malicious traffic detection methodology. In the next section, we will detail our approach's design goals and phases.

5.4 iDetector Design

In this section, we detail the design goals for malicious traffic detection and provide an overview of our approach, explaining the phases and their roles in effectively detecting malicious traffic on a home gateway for IoT environments.

5.4.1 Design Goals

From a high level, a detection solution must react quickly to notify users as is reasonably possible. A detection solution should preserve users' privacy insofar as possible, require low computational

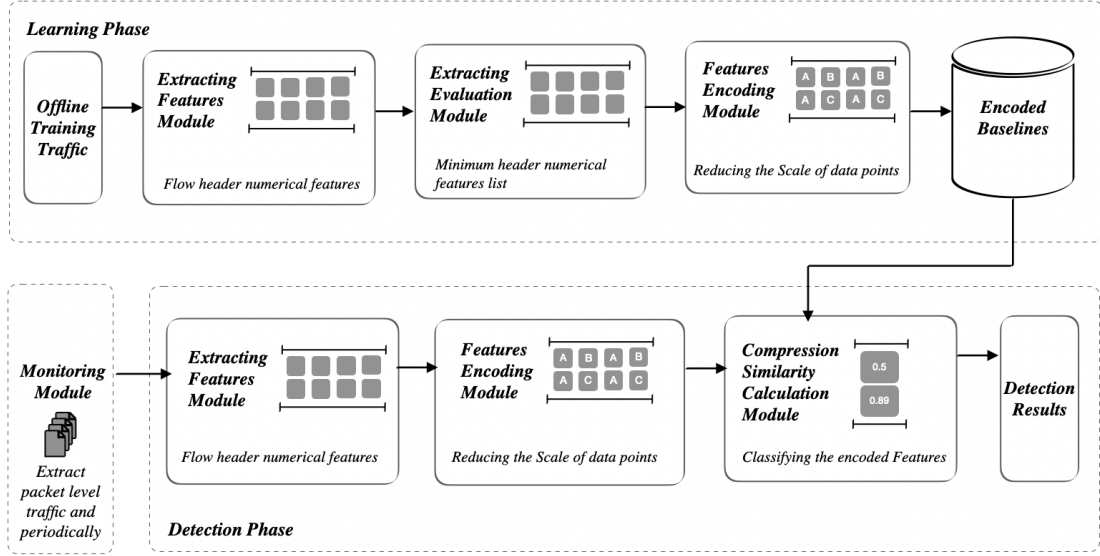


Figure 5.1: High-level design of iDetector full cycle as implemented for a single device.

processing power and memory resources, adapt to rapid changes in the environment over time, and be able to detect unseen/new malicious events.

Responsive. Network operators are often only aware of a malicious event after it has occurred. Network operators collect simple traffic measures using SNMP/sFlow [50, 215], but the disparity between malicious and normal traffic is significant [55, 223]. Extracting meaningful information about malicious activities from high-dimensional and sparse traffic data is challenging. Therefore, we need a solution that can detect malicious traffic quickly with limited information while requiring low computational processing power and memory resources.

Adaptive. The normal behaviour of IoT devices continually evolves [154]. What is considered normal behaviour today may be deemed malicious tomorrow, while some malicious activities may remain undetected. Additionally, attackers continuously develop new techniques to evade or fool detection solutions. A detection solution should not be limited to a predefined set of anomalies. Thus, we need a robust solution against performance degradation [221, 19], and adaptive enough to be deployed in different environments, regardless of the number of devices in the network.

5.4.2 Design Overview

iDetector comprises two main phases as shown in Figure 5.1: learning phase, and detection phase:

Learning Phase. In this phase, we generate and store baselines of malicious-free traffic for each device to use in the detection phase. We create these baselines manually and offline using data we passively collected over 21 days from our testbed, where devices operate under various behavioural patterns. We assume that 21 days is enough to capture all possible behavioural patterns of each device [154, 104]. We categorise the traffic into three types based on the device’s mode of operation: setup, idle, and active. This focus addresses the diverse and evolving nature of IoT environments, as IoT devices exhibit distinct patterns across operational modes. Early identification of deviations from these patterns is critical for effective malicious behaviour detection. Because

such deviations often serve as the first indicators of potential security attack [126, 223].

For each device, we conduct a depth search within each mode to identify the optimal time window that minimises the false positive rate compared to similar windows within the same mode. We then save these baselines for comparison with new traffic during the detection phase (See Section 5.5.3). The Learning phase includes three modules. ① **Feature Extraction Module**; this module extracts features from the header information of network traffic collected by the monitoring module. ② **Feature Evaluation Module**; this module assesses the variability of the extracted features, evaluates correlations between them, and provides a minimal list of features for the feature encoding module. ③ **Feature Encoding Module**; this module encodes the statistical numerical features of malicious-free traffic, identifies the optimal baseline encoded features, and determines the optimal detection threshold. Full details about this calculation are in Section 5.5.3. We will detail each module in Section 5.4.3.

Detection Phase. In this phase, we periodically compare and classify new network traffic from network devices as benign or malicious. The detection phase includes three modules. ① **Feature Extraction Module**; this module extracts features from the header information of newly received network traffic. ② **Feature Encoding Module**; this module encodes the statistical numerical features of the new incoming traffic. We used those two modules from the learning phase modules. ③ **Compression Similarity Calculation Module**; this module calculates the distances between the encoded baseline features and the new traffic. We classify traffic as malicious if these distances exceed a significant threshold (greater than 0.85 via numerical validation). We will detail each module in Section 5.4.3.

5.4.3 Design Details

In this section, we present the design details of iDetector, *i.e.*, the design of modules in iDetector.

Monitoring Module

The monitoring module collects the network traffic from devices in the local network. For both phases of learning and detection, the monitoring module provides time-series traffic for the feature-extracting module. We filter collected traffic by MAC address into a separate pcap file. We empirically validate the optimal time window to buffer the traffic; we provide the details for this validation in Section 5.5.3. Those pcap files contain network packet sequences transmitted to/from an IoT device. For each sequence, we first assemble the packets into flows and bursts (analogous to related work on encrypted traffic for IoT [179, 265]). We define a flow as a chronologically ordered set of TCP segments/UDP datagrams with the same 5-tuple (source IP, source port, destination IP, destination port, transport protocol). Because flows can last hours or even days, we may need to divide flows into smaller chunks of data for event extraction. Similar to prior work, we use the notion of bursts [265, 227]. A flow burst is a consecutive chunk of packets from the same flow in which the interval between any two successive packets is less than a threshold, which we set at 1-second, as empirically validated by prior work [265, 227].

Feature Description	Feature Name
sum of bytes of sent packets	<i>sum_opkt</i>
sum of received packets	<i>sum_ipkt</i>
sum of sent bytes	<i>sum_oby</i>
sum of received bytes	<i>sum_ibyt</i>
average size of sent bytes	<i>avg_oby</i>
average size of received bytes	<i>avg_ibyt</i>
average size of sent packets	<i>avg_opkt</i>
average size of received packets	<i>avg_ipkt</i>
list of unique dest. ports	<i>list_prt</i>
maximum interval between packets	<i>max_ipkt</i>
minimum interval between packets	<i>min_ipkt</i>

(a) Full extracted features from flows.

<i>Malicious Event</i>	<i>Features Name</i>
Information gathering	<i>sum_opkt, sum_ipkt</i>
	<i>sum_oby, sum_ibyt</i>
	<i>avg_ipkt, avg_ibyt</i>
DDoS	<i>avg_ipkt, avg_ibyt</i>
	<i>avg_opkt, avg_oby</i>
	<i>max_ipkt, min_ipkt</i>
Botnet	<i>avg_ipkt, avg_ibyt,</i>
	<i>max_ipkt, min_ipkt,</i>
	<i>list_prt</i>

(b) Minimum list of features based on malicious event.

Table 5.2: List of features extracted from flows, and minimum list of features used in detection.

Feature Extraction Module

In Chapter 3, we discussed how network operators often rely on sampled traffic and traffic encryption, which provide them with limited information. In Chapter 2 (Section 2.2.6), we highlighted researchers using network flow relevance in addressing challenges such as traffic encryption and privacy preservation. In Section 5.4, we highlighted that our detection model must react quickly to notify users as soon as reasonably possible. The solution should also preserve users' privacy as much as possible, require minimal computational power and memory, adapt to rapid changes in the environment, and detect new or unseen malicious events. To achieve these objectives, we use simple traffic measurements that reflect real-world practices typically followed by network operators [215].

In the training and detection phases, we look separately at the TCP/UDP flows of each IoT device. Each device can generate multiple flows. Per device, per flow, we extract features taken from information that is not encrypted, usually header information. We extracted source and destination

IP addresses, source and destination port numbers, numbers of packets sent and received, the total number of bytes of packets sent and received, and the start and end times of the flow. Table 5.2a for full extracted statistical features extracted from those features. These features are numerical values.

In the learning phase, we pass these features to the feature evaluation module to evaluate their variability. After that, we pass the minimum list resulting from feature evaluation to the feature encoding module. In Table 5.2b, we present the minimum set of features associated with each malicious event (as discussed in Chapter 2), focusing on the most common types, including Distributed Denial of Service (DDoS) attacks, scanning, and botnets [34, 62, 230, 231]. While in the detection phase the extracted features are sent directly to the encoding module during the detection phase since we have already determined the minimum set to use during the learning phase.

Feature Evaluation Module

In Chapter 2 (Section 2.2.6), we highlight that in IoT environments, traffic data is typically time-series and multidimensional, requiring methods capable of handling temporal dependencies and multiple attributes. As IoT traffic consists of time-series data with high dimensionality, it can be challenging to analyse and interpret IoT behaviour efficiently. We aim to extract the most relevant features from IoT traffic, enabling us to focus on critical information and simplify the overall analysis. This approach is essential to handle large time-series datasets effectively, reduce computational complexity, and provide more precise insights for decision-making and predictive tasks. This reduction makes the data more manageable, ensuring it can be processed efficiently and quickly. We reduce the number of data points while preserving the essential patterns and trends using approximation algorithms.

Approximation algorithms for time-series, which “try to capture the most important information from time-series” and “can be seen as simple feature extraction algorithms” [78, 166]. There are four approaches for approximating the time-series: Piece-wise Aggregate Approximation (PAA), Discrete Fourier Transform (DFT), multiple coefficient binning, and Symbolic Aggregate approximation (SAX). We use SAX-encoding, specifically designed for time-series data, that transforms the original time-series into a lower-dimensional symbolic representation (*i.e.*, letters) [135]. We use SAX-encoding as it offers several distinct advantages, but mainly because SAX-encoding significantly reduces the dimensionality of time-series data, as well as SAX-encoding is robust to noise and minor variations in data [135].

In the training phase, this module evaluates the variability of features that can be encoded using SAX-encoding, aiming to preserve the patterns in time-series data representing network traffic [134]. To achieve this, we first assess the correlation between features extracted by the feature extraction module, ensuring that the selected features are uncorrelated [134, 93]. Ideally, completely uncorrelated features have a correlation value of 0. However, since this is rarely the practice case, we set a threshold of 0.1 to allow for slight, negligible correlations [93]. Then, we calculate the variance for each feature in the uncorrelated list to evaluate feature variability. High variability is desirable because it indicates that the feature has meaningful fluctuations over time, which SAX-encoding can effectively encode [134]. Features with very low variability (near-zero

variance) are less useful for SAX-encoding, as they need to provide sufficient information for meaningful symbolic transformation [134]. This module then provides a refined list of features with high variability to the feature encoding module.

Feature Encoding Module

In Chapter 2 (Section 2.2.6), we highlight that in IoT environments, traffic data is typically time-series and multidimensional, requiring methods capable of handling temporal dependencies and multiple attributes. As mentioned above, we use SAX-encoding as it offers several distinct advantages, but mainly because SAX-encoding significantly reduces the dimensionality of time-series data, as well as SAX-encoding is robust to noise and minor variations in data [135]. Here is how SAX-encoding works [135]:

(1) **Normalisation.** SAX-encoding assumes that the time-series data has been z-normalised, meaning it has a mean of zero and a standard deviation of one. This normalisation is crucial for SAX-encoding to compare different time-series data effectively. As a first step, SAX-encoding ensures that the time-series data has a mean of zero and a standard deviation of one. Given a time-series $T = t_1, t_2, \dots, t_n$; first SAX-encoding compute the mean (μ) and standard deviation (σ) of T [135]:

$$\mu_j = \frac{1}{n} \sum_{i=1}^n t_{ij} \quad (5.1)$$

$$\sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^n (t_{ij} - \mu_j)^2} \quad (5.2)$$

then normalise each data point t_i using the following equation [135]:

$$t'_{ij} = \frac{t_{ij} - \mu_j}{\sigma_j} \quad \text{for } i = 1, 2, \dots, n \text{ and } j = 1, 2, \dots, p \quad (5.3)$$

(2) **Segmentation.** The normalise time-series $T = t_1, t_2, \dots, t_n$ is divided into w equally sized segments. Let w be the desired number of segments. Each segment will have a length of n/w [135].

(3) **Piecewise Aggregate Approximation (PAA).** Each segment is represented by its mean value. Divide T into w segments, where each segment S_j contains $m = n/w$ points [135]:

$$S_j = \{t'_{(j-1)m+1}, t'_{(j-1)m+2}, \dots, t'_{jm}\} \quad \text{for } j = 1, 2, \dots, w \quad (5.4)$$

then Compute the mean of each segment [135]:

$$\bar{S}_{jk} = \frac{1}{m} \sum_{i=(j-1)m+1}^{jm} t'_{ik} \quad (5.5)$$

The result is a reduced representation of the time-series,

$$T_{PAA} = \{\bar{S}_{1k}, \bar{S}_{2k}, \dots, \bar{S}_{wk}\} \quad \text{for } k = 1, 2, \dots, p$$

(4) **Discretisation** The PAA values are mapped to discrete symbols based on predefined breakpoints. Define an alphabet size α and determine the breakpoints $\beta_1, \beta_2, \dots, \beta_{\alpha-1}$ that divide a standard normal distribution $N(0,1)$ into α equal sized areas.

Map each PAA value $\bar{S}_j$ to a symbol [135]:

$$a_{jk} = \begin{cases} \text{symbol}_1 & \text{if } \bar{S}_{jk} \leq \beta_1 \\ \text{symbol}_2 & \text{if } \beta_1 < \bar{S}_{jk} \leq \beta_2 \\ \vdots & \\ \text{symbol}_\alpha & \text{if } \bar{S}_{jk} > \beta_{\alpha-1} \end{cases} \quad (5.6)$$

The final symbolic representation of the time-series is [135]:

$$T_{SAX} = \{a_{1k}, a_{2k}, \dots, a_{wk}\} \quad \text{for } k = 1, 2, \dots, p$$

Similarity Calculation Module

In this module, we calculate the probability of how likely new encoded traffic is similar to malicious-free baseline traffic. A simple approach for detecting a change in time-series data (*i.e.*, malicious event) is to formulate the problem as an estimation of similarity between sequential data sets. Many similarity-based techniques such as [248, 245, 246, 244, 130] gave promising results when used in malicious events detection. We use the Compression-based Dissimilarity Measure (CDM) to detect malicious events in network traffic because the CDM can deal with high dimensional, large, and variable lengths of time-series data without affecting the time, complexity, and resources [114]. Additionally, time-series data contain many repetitive patterns, which the CDM is good at identifying [114].

A similarity is a relation between two sets of data and scalar numbers. The similarity measurement maps to intervals $[0, 1]$ or $[-1, 1]$, where 1 is either the minimum or maximum of similarity based on the adopted technique [114, 232]. We assume one as the maximum dissimilarity. The closer the dissimilarity is to 1, the greater the distance between the two behaviours (meaning they are more likely different). Conversely, the closer the dissimilarity is to 0, the lower the distance between the two behaviours (meaning they are more likely similar).

To calculate CDM, let x and y be two strings in which we are interested in measuring their similarity. Let xy be the concatenation string of x and y . Let $C(x)$, $C(y)$, $C(xy)$ be the size of the compressed file, where the compressed file consists of string x , y and xy , respectively. The CDM (Eq 5.7) dissimilarity is a non-zero value. When the $CDM(x, y)$ is closer to 1, this means x and y are more likely different, while when the $CDM(x, y)$ is closer to 0, this means x and y are more likely similar. The CDM is defined as follows [114]:

$$CDM(x,y) = \frac{C(xy)}{C(x) + C(y)} \quad (5.7)$$

We need to choose a compressor to calculate CDM, for example, a statistical-based or dictionary-based compressor (*e.g.*, gzip and bzip2). We use zip compressor, a deflate compression algorithm [207, 243]. We use a zip compressor because it is fast [207, 243], and no memory resources are required as we do not save the compressed file. The proposed detection approach aims to run efficiently on a device commensurate with a home gateway, where resource efficiency and low latency are essential. This module's output is the probability of how likely traffic is similar to malicious free traffic. We use this probability to mark the traffic as malicious or malicious-free.

5.4.4 Theoretical Analysis

Our goal is to show that the entropy difference $\Delta H = |H_{\text{original}} - H_{\text{SAX}}|$ between the original multivariate time-series data and its SAX-encoded version is minimised, thereby preserving information [216].

Before stating and proving the theory, we establish some mathematical baselines to ensure a foundation for the subsequent analysis.

Calculate Entropy for Original Time-Series Data

The general formula to calculate the entropy $H(X)$ for time-series traffic data with a probability density function (PDF) is defined as follows [216, 145]:

$$H(X) = - \int_{-\infty}^{\infty} p(x) \log p(x) dx \quad (5.8)$$

However, this formula is designed for scalar univariate time series and is not suited for multivariate time-series commonly measured in experimental and IoT environments [8, 200, 41]. We use the Multivariate Sample Entropy (MSampEn) [8, 7] for multivariate time-series data.

MSampEn quantifies the regularity or complexity of the multivariate time-series by computing the conditional probability that two similar sequences for m points remain identical at the next point within a tolerance r .

The MSampEn of a multivariate time-series T with d dimensions and length N is defined as follows:

Form the vectors $\mathbf{X}(i)$ and $\mathbf{X}(j)$ for $i, j = 1, 2, \dots, N - m$:

$$\mathbf{X}(i) = [\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{i+m-1}]$$

where $\mathbf{x}_i$ is a d -dimensional vector at time i .

Compute the distance d_{ij} between vectors $\mathbf{X}(i)$ and $\mathbf{X}(j)$ using a norm (*e.g.*, Euclidean distance) [247]:

$$d_{ij} = \max_{k=1,\dots,d} \|\mathbf{x}_{i+k-1} - \mathbf{x}_{j+k-1}\| \quad (5.9)$$

Count the number of vector pairs (i, j) such that $d_{ij} \leq r$ [191, 200]:

$$B_i^m = \frac{1}{N-m} \sum_{j=1, j \neq i}^{N-m} \mathbb{I}(d_{ij} \leq r) \quad (5.10)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Form $\mathbf{X}'(i)$ and $\mathbf{X}'(j)$ by extending $\mathbf{X}(i)$ and $\mathbf{X}(j)$ to $m+1$ dimensions:

$$\mathbf{X}'(i) = [\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{i+m}] \quad (5.11)$$

Compute the distance d'_{ij} between vectors $\mathbf{X}'(i)$ and $\mathbf{X}'(j)$ [247]:

$$d'_{ij} = \max_{k=1,\dots,d} \|\mathbf{x}_{i+k-1} - \mathbf{x}_{j+k-1}\| \quad (5.12)$$

Count the number of vector pairs (i, j) such that $d'_{ij} \leq r$ [191, 200]:

$$A_i^m = \frac{1}{N-m} \sum_{j=1, j \neq i}^{N-m} \mathbb{I}(d'_{ij} \leq r) \quad (5.13)$$

Compute the MSampEn as [8, 7]:

$$\text{MSampEn}(m, r, N) = -\ln \left(\frac{\sum_{i=1}^{N-m} A_i^m}{\sum_{i=1}^{N-m} B_i^m} \right) \quad (5.14)$$

Calculate Entropy for SAX-encoded Data

SAX-encoding transforms and discretises the continuous time-series data into symbols. We estimate the PDF of the SAX-encoded data by modelling the SAX-encoding symbols as a discrete probability distribution and calculate the entropy using the formula for discrete entropy [216].

Let the SAX-encoded time-series be T_{SAX} with symbols

$\{a_{i,1}, a_{i,2}, \dots, a_{i,w}\}$ for the i^{th} variable and their respective probabilities $P(a_{i,j})$.

The entropy $H_{\text{SAX},i}$ for the i^{th} variable in the SAX-encoded data is given by [216]:

$$H_{\text{SAX},i} = - \sum_{j=1}^{m_{\text{SAX},i}} p_{i,j} \log_2(p_{i,j}) \quad (5.15)$$

where $m_{\text{SAX},i}$ is the number of unique SAX-encoding symbols for the i -th variable, and $p_{i,j}$ is the probability of the j^{th} symbol occurring.

To compute the overall entropy of the SAX-encoded data, we sum the entropies of individual variables:

$$H_{\text{SAX}} = \sum_{i=1}^d H_{\text{SAX},i} \quad (5.16)$$

This formulation assumes that the variables are independent, meaning the joint probability distribution factorises into the product of marginal distributions. While this assumption simplifies computation, it may not always hold in practice. However, as SAX encoding transforms each variable separately, this approach provides a reasonable approximation of the total entropy when dependencies between variables are weak or negligible. If strong dependencies exist, a more accurate measure would require computing the joint entropy directly.

Quantify Loss in Information

The difference in entropy ΔH between the original time-series and the SAX-encoded data is computed as follows [216]:

Calculate the MSampEn of the original multivariate time-series data [8, 7]:

$$H_{\text{original}} = \text{MSampEn}(m, r, N) \quad (5.17)$$

Calculate the overall entropy of the SAX-encoded data:

$$H_{\text{SAX}} = \sum_{i=1}^d H_{\text{SAX},i} \quad (5.18)$$

The difference in entropy ΔH is given by [216]:

$$\Delta H = |H_{\text{original}} - H_{\text{SAX}}| \quad (5.19)$$

A higher value of ΔH indicates greater information loss during SAX-encoding, whereas a lower value indicates less loss [216].

Theorem: Entropy Preservation in SAX-Encoding

Building upon the mathematical baselines and to formally establish the minimal information loss in SAX-encoding, we present the following theorem, which provides a theoretical bound on the entropy difference and substantiates our empirical observations.

Let X be a multivariate time-series with entropy $H(X)$, and let X_{SAX} be its SAX-encoded representation with entropy H_{SAX} . Under appropriate SAX parameter selection (alphabet size m_{SAX} , window size, and segmentation size), the entropy difference $\Delta H = |H(X) - H_{\text{SAX}}|$ is bounded by:

$$\Delta H \leq \varepsilon \log(1/\varepsilon), \quad (5.20)$$

where $\varepsilon = \sup_j |p(x_j) - p_{\text{SAX}}(x_j)|$ is the discretisation error. As $m_{\text{SAX}} \rightarrow \infty$, $\Delta H \rightarrow 0$, ensuring entropy preservation [56].

Mathematical Proof of Minimal Information Loss in SAX-Encoding

To mathematically prove that SAX encoding minimally impacts the information content of multivariate time-series data, we analyse the convergence of probability distributions and its effect on entropy. Specifically, we show that as resolution increases, SAX-discretised distributions approximate the original probability distributions. To support this, we employ both the Approximation of Probability Distributions and Empirical Bounds methods.

1. Approximation of Probability Distributions

Step 1.1: Defining Entropy in the Original Continuous Distribution

The entropy $H(X)$ of the original continuous multivariate time-series distribution captures the information content or uncertainty of the continuous data. This is generally expressed as [216]:

$$H(X) = - \int_{-\infty}^{\infty} p(x) \log p(x) dx$$

Here, $p(x)$ is the probability density function (PDF) of the continuous variable X .

The integral calculates the expected value of the information content $-\log p(x)$ over the entire range of X . This continuous entropy measure reflects the "spread" or "unpredictability" of the data, where higher entropy indicates greater uncertainty.

Step 1.2: Defining Entropy in the SAX-Encoded Discrete Distribution

SAX-encoding involves segmenting the continuous time-series data into discrete symbols. This results in a discrete distribution where each symbol has an associated probability $p_{i,j}$, representing the likelihood of the j -th symbol in the i -th variable (dimension) across the dataset. The entropy H_{SAX} for the SAX-encoded distribution is calculated using the discrete entropy formula:

$$H_{\text{SAX}} = - \sum_{i=1}^d \sum_{j=1}^{m_{\text{SAX},i}} p_{i,j} \log p_{i,j}$$

In this formula:

- d represents the number of variables in the multivariate time-series.
- $m_{\text{SAX},i}$ represents the number of discrete symbols for variable i .
- $p_{i,j}$ is the probability of observing symbol j in variable i .

This discrete entropy captures the information content after the time-series data has been simplified into SAX symbols, reflecting the reduced "spread" in the SAX representation.

Step 1.3: Discretisation Error in Probability Distributions

When continuous values are mapped to discrete symbols in SAX encoding, a small error (or "discretisation error") ε is introduced because the continuous data is "binned" into a finite set of symbols. This error ε can be quantified by measuring the deviation of the SAX-discretised probability distribution $p_{\text{SAX}}(x)$ from the original probability distribution $p(x)$. We define ε as the maximum difference between the original and SAX-discretised probability masses for any symbol x_i :

$$\varepsilon = \sup_i |p(x_i) - p_{\text{SAX}}(x_i)|$$

This error term ε indicates the maximum "misalignment" between the continuous and discrete distributions introduced by SAX encoding. Smaller ε values correspond to better preservation of the original distribution.

Step 1.4: Convergence in Distribution with Increased SAX Resolution

SAX-encoding transforms a continuous time-series into a sequence of discrete symbols. To quantify information loss, we analyse how the probability distribution of the encoded symbols, denoted as $p_{\text{SAX}}(x)$, relates to the original distribution $p(x)$.

1.4.1 Defining $p_{\text{SAX}}(x)$. Given a continuous random variable X with probability density function (PDF) $p(x)$, SAX discretises X into m_{SAX} bins, each corresponding to a symbol. The probability mass function (PMF) of the SAX-encoded data is defined as:

$$p_{\text{SAX}}(x_j) = \int_{B_j} p(x) dx \quad (5.21)$$

where B_j represents the j^{th} bin in the discretisation scheme, and x_j is the representative symbol for that bin. This definition ensures that $p_{\text{SAX}}(x)$ is a valid probability distribution derived from the continuous density function.

1.4.2 Convergence of $p_{\text{SAX}}(x)$ to $p(x)$. As the resolution increases ($m_{\text{SAX}} \rightarrow \infty$), the bin width Δx decreases, meaning each bin B_j becomes smaller. Under standard assumptions (e.g., smoothness of $p(x)$), we apply the mean-value theorem for integrals, which states that for sufficiently small bins:

$$p_{\text{SAX}}(x_j) = p(\xi_j) \Delta x \quad (5.22)$$

for some $\xi_j \in B_j$. As $\Delta x \rightarrow 0$, the right-hand side approaches $p(x_j)$, leading to:

$$\lim_{m_{\text{SAX}} \rightarrow \infty} p_{\text{SAX}}(x) = p(x). \quad (5.23)$$

This proves that the discretised distribution converges to the original continuous distribution in the

limit.

1.4.3 Entropy Convergence. Entropy is defined for a discrete distribution as:

$$H_{\text{SAX}} = - \sum_{j=1}^{m_{\text{SAX}}} p_{\text{SAX}}(x_j) \log_2 p_{\text{SAX}}(x_j). \quad (5.24)$$

Using the asymptotic expansion of entropy under discretisation [56], the entropy difference is bounded by the discretisation error ε :

$$\Delta H = |H(X) - H_{\text{SAX}}| \leq \varepsilon \log(1/\varepsilon), \quad (5.25)$$

where:

$$\varepsilon = \sup_j |p(x_j) - p_{\text{SAX}}(x_j)|. \quad (5.26)$$

Since we showed $p_{\text{SAX}}(x) \rightarrow p(x)$ as $m_{\text{SAX}} \rightarrow \infty$, it follows that $\varepsilon \rightarrow 0$, implying:

$$\lim_{m_{\text{SAX}} \rightarrow \infty} H_{\text{SAX}} = H(X). \quad (5.27)$$

This establishes that SAX-encoding preserves entropy asymptotically with minimal information loss as resolution increases.

2. Empirical Validation

To further support the analytical approximation of probability distribution, we performed empirical observations across our collected dataset by generating time-series, applying the SAX-encoding, and computing ΔH . Figure 5.2 plotting a subset of ΔH against increasing SAX symbol count to empirically show that ΔH approaches zero as the resolution improves for the TP-Link lightbulb.

The empirical results show that with appropriate window size, segmentation size, and alphabet size, the maximum difference in entropy between the original time-series and the SAX-encoding (ΔH) is $\approx 1.3406387318906961 \times 10^{-06}$, indicating that SAX-encoding effectively preserves the entropy of the original data.

5.5 Results

In this section, we describe the experimental setup and data collection process, followed by a comprehensive analysis of the evaluation results for iDetector. We detail the optimal detection settings and rates for known and novel malicious events, compare iDetector with the only publicly available state-of-the-art IoT malicious traffic framework, present an overhead analysis, and analyse iDetector’s robustness against adversarial models.

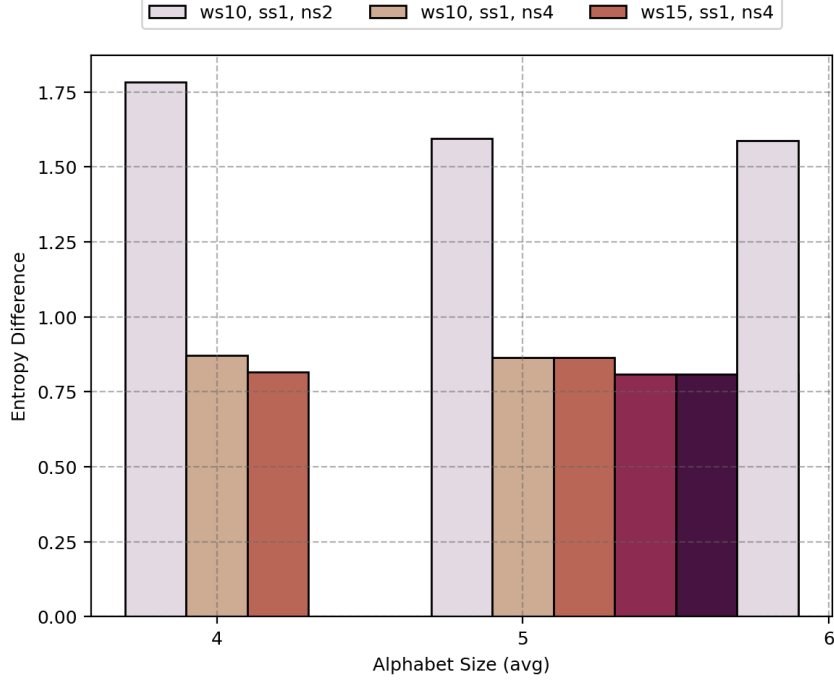


Figure 5.2: Empirical entropy difference ΔH for different SAX parameters in TP-Link Smart Plug.

Dataset Name	Dataset Descriptions	No. of Devices	Attacks Descriptions	No. of Victims	Size of Benign	Size of Malicious	Ratio of Malicious
HCRL[113]	Collected by Hacking and Countermeasure Research Lab in Korea Uni., between May and June 2019 mainly to evaluate IoT network intrusion dataset	1	Attacks from port scanning, host discovery OS/Version Detection, spoofing, and Flooding	1	8.4MB	1.46GB	60.97%
CICIoT2023[59]	Collected by Canadian Institute for Cybersecurity between June and October of 2023, mainly for benchmarking for Large-Scale Attacks in IoT.	105	Attacks from DDoS, DoS, Recon, Web-based, brute force, spoofing, and Mirai	62	6.5GB	1.01TB	21.97%
UNSW[96]	Collected by UNSW between June and October of 2018, mainly to evaluate detection solution against volumetric attack.	33	Attacks Volumetric attacks ARP spoofing, (TCP SYN flooding, Fraggle (UDP flooding), and Ping of Death. Reflective attacks include SNMP, SSDP, TCP SYN, and Smurf.	10	96GB	8.66GB	5.48%
Yourthings[189]	Multipurpose datasets, created by Georgia Tech Institutions, used in evaluating several IoT based solutions and analysis.	75	Benign traffic only	/	67GB	/	/
HorusEye[70]	Collected by Peng Cheng Laboratory, one month period, using 8 cameras, 7 sensors, and one smart gateway, mainly to evaluate the HorusEye framework	14	Botnet infection, including Airda, Bashlite and Mirai; data exfiltration, including keylogging and data theft; scanning attacks, including service scan and OS scan; DDoS attack, including HTTP/TCP/UDP DDoS	1	13.9GB	43.21MB	0.65%
iDetector	Collected from testbed mimic smart home devices were in active and idle modes	15	DDoS attack including TCP/UDP DDoS, SYNC flood, scanning including service scan and OS scan, ARP spoofing, and Mirai	3	10.06GB	3.25GB	44.95%

Table 5.3: Testbed and dataset configuration and statistics used to evaluate iDetector. We use "/" when the dataset is benign only traffic.

5.5.1 Experimental Setup

In this section, we describe the experimental setup, including the testing environment, implementation of iDetector phases, and performance metrics.

Environment. We deployed iDetector on Raspberry Pi 4 Model B running Linux-OS 10 (buster). The Raspberry Pi has 8 GB RAM and a 64-bit quad-core Cortex-A72 processor. We implemented all iDetector phases in Python 3.9. We ran iDetector as a background process and automatically performed detection on a one-minute basis.

Performance Metrics. The distribution of benign and malicious flows in our datasets is unequal (*i.e.*, imbalanced dataset); for example, iDetector dataset benign flows are 55.05% while malicious

flows are 44.95% (see Table 5.3). Having an imbalanced dataset leads to an imbalanced classification problem [221]. Maximising accuracy is not a suitable performance metric for evaluating an imbalanced classification problem [221]. However, metrics such as True Positive Rate (TPR) and False Positive Rate (FPR) are more suitable for evaluating such an imbalanced classification problem [221]. We quantified the true positive rate, false positive rate, F1-score (the harmonic mean of precision and recall), and Area Under the Precision-Recall Curve (AUCPR).

5.5.2 Data Collection and Labelling

In this section, we detail the data collection process and outline the pipeline used to partition the dataset for training and testing. We use data generated in our IoT testbed and public datasets, with 243 types of IoT devices and four attacks. Table 5.3 summarises these datasets’ duration, purpose, number of victims, devices, and attack instances.

Normal dataset- Ours. We collected traffic from our testbeds consisting of 15 IoT and non-IoT devices. Our testbed provides a controlled environment that mimics real-world home deployments. By instrumenting everyday tasks typical of home users and providing diverse attack scenarios, the testbed allows for systematic control over different variables, enhancing the reliability of the results. Unlike real-world datasets [104], they may contain uncontrolled and latent factors that impact the evaluation of IoT malicious detection. The controlled environment of the testbeds eliminates confounding variables that allow us to set an upper bound on malicious traffic IoT identification performance using pristine conditions. This approach improves the accuracy of the evaluation and ensures the reproducibility and repeatability of experiments, which are essential for empirical research. Lastly, using a controlled testbed addresses ethical considerations by avoiding potential privacy issues associated with real-world data collection.

Normal datasets- [59, 96, 189, 113]. In addition to our dataset, we used publicly available datasets. Many datasets that cover a wide range of IoT devices are public and available to use [96, 31, 186, 196, 104]. However, we cannot use all these datasets to evaluate iDetector because (1) IoT Inspector dataset does not reflect trends in smart home IoT behaviour, as users start and end the traffic collecting process [104], and (2) for Aposemat IoT-23 datasets [186, 94], they lack sufficient information to be reused under iDetector, such as information to generate baseline behavioural profile. We were able to use HCRL [113], CICIOT2023 [59], UNSW [96], Horus-Eye [70] and yourthings [189] datasets. We selected the longest continuous PCAP files capturing network traffic for each dataset. We labelled these traffic traces using a dataset and device model to create a bidirectional flow.

Attack dataset- Ours. We replicated cyberattacks on our testbed. We implemented all malicious activities using Python or bash scripts. We deployed the malicious activities on a Raspberry Pi 4 Model B with Kali Linux-OS (2020.4). We configured the Raspberry Pi (acting as a home gateway) to reroute all unfiltered outgoing traffic to a specific port. This rerouting is done to prevent the spreading of malicious traffic on the public Internet. After each attack experiment, we reset the network setup and devices. We infect three of the IoT devices and reproduce OS scan, service scan, TCP DDoS, and UDP DDoS attacks for testing. Table 5.3 describes replicated attack instances, the total size of flows generated, and the percentage of malicious flows. We labelled

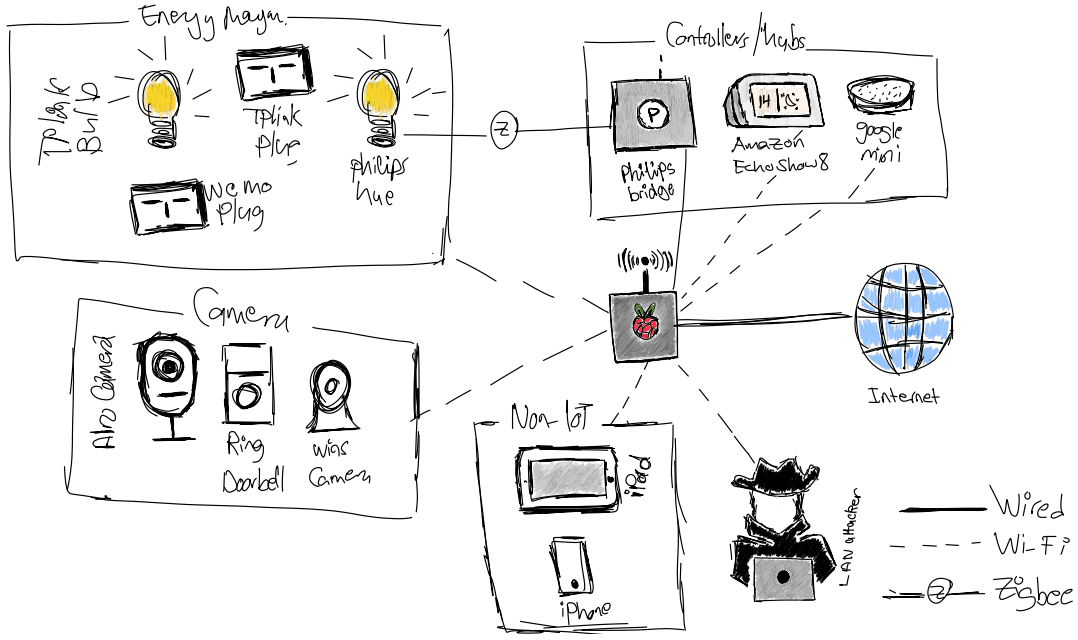


Figure 5.3: **Experimental setup to examine malicious events in our testbed.**

these traffic traces by year, device mode of operation, and device model to create a bidirectional flow and malicious activities.

Attack dataset- [59, 96, 113]. We use public attack datasets [59, 96, 113]. Botnet infection, including Mirai; (2) data exfiltration, including keylogging and data theft; (3) scanning attacks, including service scan and OS scan; (4) distributed denial of service attack (DDoS), including HTTP DDoS, TCP DDoS, and UDP DDoS. Table 5.3 describes replicated attack instances, the total size of flows generated, and the percentage of malicious flows. We labelled these traffic traces using a dataset and device model to create a bidirectional flow.

Dataset Partitioning. We divided the collected dataset into the following: (1) **Baselines Set:** we require malicious-free traffic to establish our baselines, to be used to calibrate the results. We performed baseline generation manually using passive traffic. We utilised three sets of 21 days spanned over different device modes of operation. We assumed that 21 days is sufficient for most of the other pattern behaviours of IoT devices in an environment [154, 126]. We use this baseline traffic to establish three baselines for each IoT device: (a) Setup: this baseline represents traffic from the device once set up (companion device, *e.g.*, smartphone app) for the first time or when the device is reset or rebooted. (b) Idle: this baseline represents traffic generated by IoT device when there is no explicit user-to-device interaction and no explicit device-to-device interaction. We excluded the device setup traffic. (c) Active: this baseline represents traffic generated by the IoT device when there is an explicit user-to-device interaction and an explicit device-to-device interaction. Again, we excluded the device setup traffic. We divided the traffic into equal time windows, applied encoding to each window, and compared the similarities across all windows over the 21 days. We chose the window (1-second) that caused the least false positives for each device and mode of operation when applying the similarity analysis (see full details in Section 5.5.3). (2) **Training and Validating Sets:** we used this subset to train and validate the detection time

window, segmentation for traffic encoding, and setting thresholds for similarity detection. The percentage of malicious samples in this set was 25%. (3) **Testing Set:** we used this subset of the dataset to evaluate the final performance of the solution after it has been trained and validated. We used this subset using the optimal time window, optimal segmentation, and optimal threshold chosen from the training and validating set. The percentage of malicious samples in this set was 75%. We ensured that the test set contained new and unknown malicious traffic to provide an unbiased evaluation of the solution's ability to detect new/unseen samples.

5.5.3 Detection Results

We designed the evaluation to answer the following questions:

1. Can iDetector maintain a high detection rate for known (seen) and unknown (unseen) malicious traffic? (Section 5.5.3)
2. How does iDetector's detection rate compare to other relevant detection frameworks? (Section 5.5.3)
3. Does iDetector offer lower implementation overheads, making deploying more practical? (Section 5.5.3)
4. How does iDetector function under an adversarial model? (Section 5.5.3)

Optimal Detection Settings

We aim to minimise information loss and false positive rates. We conducted an automated depth search to optimise several parameters: window size, window slide, segmentation size, and alphabet size. We perform the automation search across traffic data collected for 21 days for baseline generation. Automated baseline generation is beyond the scope of this paper, but vendors could provide an 'initial baseline', which they generate in the lab with the knowledge of the default firmware. Then, consumer set-up could include a short training period, a.k.a. iPhone face recognition. We determined the set of parameters through an automated search, which resulted in a minimal entropy difference. We found that the optimal settings were a window size of 5 minutes, a sliding window of 1 minute, four segments in SAX-encoding, and an average alphabet size of 6. These settings resulted in a minimal entropy difference of $1.3406387318906961 \times 10^{-06}$. From this automated search, we determined the frequency at which detection should repeat automatically, corresponding to the length of the pcap files obtained during the monitoring module and used in the encoding module to encode the traffic. The frequency of automated detection was per minute buffering to feed a 5-minute time window. We performed an automated search for an optimal baseline afterwards. Initially, we hypothesised that the baseline could be any time window slot from the baseline subset. However, during experiments, we found that there is an optimal time window slot that causes the most minor false negatives. Interestingly, the optimal time window slot was various across devices. Some device's baseline was the first time window, whereas others were the last time window. At the same time, the rest time window is in the middle.

Dataset	Attack	TPR	FPR	F1-Score	AUCPR
Ours	Service scan	0.96	0.04	0.97	0.95
	OS scan	0.96	0.04	0.97	0.95
	TCP DDoS	0.91	0.10	0.94	0.92
	UDP DDoS	0.91	0.10	0.94	0.92
	Spoofing	0.92	0.10	0.95	0.93
UNSW [96] CIC2023 [59] HCRL[113]	Service scan	0.93	0.19	0.94	0.95
	OS scan	0.93	0.19	0.94	0.95
	HTTP DDoS	0.82	0.01	0.90	0.82
	TCP DDoS	0.90	0.17	0.94	0.90
	UDP DDoS	0.90	0.17	0.94	0.90
	Spoofing	0.85	0.03	0.90	0.87
	Mirai	0.93	0.19	0.94	0.95

Table 5.4: Detection rate of iDetector for different malicious activities conducted by our experiments and public datasets.

Framework	Metric	UNSW[96]	CIC2023[59]	HCRL[113]	Yourthings[189]	HorusEye[70]	iDetector
iDetector	TPR	0.82156	0.97013	0.88000	0.99998	0.99932	0.96273
	FPR	0.00069	0.33333	0.04878	0.00001	0.00001	0.03853
	F1-Score	0.90200	0.98455	0.89795	0.99999	0.99966	0.97623
	AUPRC	0.82287	0.97972	0.91679	0.99998	0.99901	0.95957
HorusEye	TPR	0.66121	0.50000	0.50000	/	0.73000	0.50000
	FPR	0.39213	0.50000	0.50000	/	0.00000	0.50000
	F1-Score	0.81767	0.00000	0.00000	/	0.87000	0.00000
	AUPRC	0.87384	0.69592	0.50367	/	0.96400	0.50367

Table 5.5: Comparison analysis of iDetector and HorusEye across different datasets detection rate performance.

Detection Rate

We split datasets per typical machine learning pipelines. The training and validation dataset percentage was 25%, while the percentage of the testing dataset was 75%. We split datasets under detection with a higher percentage for testing as these experiments aim to test the false positive rate of iDetector with new malicious. For devices in our datasets, we assumed three baselines (setup, idle, and active), while for IoT devices in the public dataset, we assumed one baseline (active) due to a lack of comprehensive information about device setup and experiments. See Section 5.5.2 for dataset partitioning details.

We performed known malicious events on our testbed and quantified any malicious event falsely classified as benign. This detection performed online lasted for seven weeks and involved over 15 devices, and the victims' devices included an Amazon Echo Show, a TP-Link Lightbulb, and a TP-Link smart plug. We replicated Services scan, OS scanning, TCP SYN-Flood, UDP Flood and ARP spoofing. In our dataset, iDetector shows a high detection rate across most attack types, with TPR for Service scan and OS scan attacks, each achieving 0.96. This detection rate suggests that scanning attack types are well-represented and effectively identified within our dataset. Regarding false positives (FPR), our dataset shows rates for Service scan and OS scan (FPR 0.04),

highlighting the high detection rate for these attacks. These false alarm cases happened with devices such as the Amazon Echo Show and Google Home Mini, as we observed more than one idle representation for each device. We leave an investigation of the minimum number of baselines for each device for future work. The detection rates for TCP DDoS, UDP DDoS, and Spoofing are 0.91 and 0.92, respectively, indicating a drop in detection rate. TCP DDoS, UDP DDoS, and Spoofing exhibit an FPR of 0.10, indicating an increased likelihood of false alarms. These false alarm cases happened because we used the two baselines (setup and idle baselines) and excluded the active baseline during the detection process for the TP-Link Lightbulb

We performed Unknown malicious detection on public datasets (UNSW [96], CIC2023 [59], and HCRL[113]). By newly malicious activities, we refer to any threats that were not developed, observed, or accounted for in the experimental setup and timeframe. Table 5.5 shows differing patterns in detection efficiency across attack types. The TPRs for Service scan, OS scan, and Mirai are the highest at 0.93, showing strong detection performance for these attacks. However, detection of HyperText Transfer Protocol (HTTP) DDoS falls behind with a TPR of 0.82, suggesting a challenge in identifying this attack within the public datasets. Investigating those false alarm cases, we found that the baseline of normal traffic lacks representativeness (one baseline) due to limited information about the experimental setup in UNSW [96] and CIC2023 [59] datasets, which led us to approximate the timing of normal behaviour. This limitation introduces potential inaccuracies in distinguishing between normal and malicious traffic. The FPR values for Service scan, OS scan, and Mirai attacks are elevated at 0.19, which indicates a higher incidence of false alarms for these types when using public datasets. These false alarms happened with dataset samples related to HCRL datasets. In the HCRL dataset, the duration of normal traffic samples is significantly shorter than that of attack traffic, reducing the representativeness of normal behaviours. This imbalance could lead the model to rely more heavily on attack patterns, ultimately affecting its ability to detect attacks when the normal baseline is insufficiently comprehensive accurately. HTTP DDoS shows an FPR of 0.01, indicating high detection with minimal false alarms for this attack type.

Comparison with Related Approaches

We present a quantitative comparison between iDetector and HorusEye, the only publicly available state-of-the-art IoT malicious traffic detection framework based on LSTM autoencoders [70]. While prior works often compare their approaches internally with Kitsune [163]—a framework designed initially for non-IoT malicious traffic detection—these comparisons have consistently highlighted Kitsune’s inability to detect IoT malicious traffic effectively [70]. Moreover, Kitsune, which utilises an LSTM-based model, failed to converge during our benchmarking process. Beyond these performance limitations, the current state-of-the-art solutions are ML-based and exhibit theoretical assumptions limiting their practicality. Specifically, none of the solutions considers the impact of different modes of operation in IoT devices and assumes that baseline behaviours are static and lack temporal variation. This assumption overlooks the diverse and evolving nature of IoT device behaviours, which is critical for real-world malicious traffic detection.

A recent study by Olszewski *et al.* [181] has also detailed general reproducibility challenges in

machine learning solutions within the security domain. These challenges include a lack of proper documentation, such as missing hyperparameter settings, unclear algorithmic steps, or insufficient descriptions of experimental setups. Further, code quality and availability issues—such as inaccessible, incomplete, or non-functional code—often impede replication efforts. Finally, environment-specific constraints exacerbate these reproducibility issues, including hardware configurations, software libraries, or versioning discrepancies. These factors highlight the need for more robust and transparent practices in evaluating and benchmarking IoT malicious traffic detection frameworks.

We deployed HorusEye on a Raspberry Pi 4 model B, acting as a gateway with the same experimental setup and datasets used to evaluate iDetector. Table 5.5 presents a detailed comparison between two frameworks, iDetector and HorusEye, across multiple datasets: UNSW, CIC2023, HCRL, Yourthings, HorusEye, and iDetector. We reproduced HorusEye classification using their datasets. We could not reconcile the results of the original paper. We investigated the code published [240]. The code calculates an AUROC with an unusual approach [239]. HorusEye uses RMSE values (rmse_list) as if they were classifier scores. This approach is unconventional because RMSE is typically a continuous error metric, not a probability or confidence score [11]. RMSE is not concerned with classifying samples as true positives or false positives but rather with how close predicted values are to actual values on average. During the HorusEye evaluation, four days from UNSW were used randomly as the training dataset and the next two days as the testing dataset. In our reproduction, we trained the model on the first seven days and tested it the following days. We fixed the TPR and FPR calculation method. We get different results once we rerun the code with the same dataset.

From Table 5.5, we can see that the maximum TPR achieved by HorusEye is 0.73 on their dataset. HorusEye fails to classify traffic in the CIC2023, HCRL and iDetector data. A TPR of 0.5 is the same random selection instead of classification. iDetector excels in minimising false positives and maximising true positives. The lowest FPR achieved by iDetector is 0.00001 on the Yourthings dataset. The lowest TPR for iDetector is 0.82, achieved on the UNSW dataset, while the highest true positive rate is 0.99, achieved on the Yourthings dataset. Excluding the Yourthings dataset (attack-free), the lowest TPR achieved by iDetector is 0.82 on the UNSW dataset, and the highest TPR is 0.97 on the CIC2023 dataset. The lowest FPR achieved by iDetector, excluding Yourthings, is 0.00069 on the UNSW dataset. HorusEye, unlike iDetector, could not perform classification in cases where traffic is attack-free, such as in the yourthings dataset. Overall, HorusEye, while showing some strengths, generally under-performs compared to iDetector. These results make iDetector the preferred choice in environments where low false positives and high true positives are needed.

Overhead Analysis

Table 5.6 compares the performance of different datasets using several metrics: the number of devices, disk usage (MB), CPU usage (%), and latency (s).

Disk Usage. In the similarity calculation module, we need disk storage to save devices' Baseline Encoded Features (BNF). We used a Raspberry Pi with 8GB RAM and 64GB disk space. We save

Dataset Name	Metric	No. Devices	Disk Usage (MB)	CPU Usage (%)	Latency (s)
HCRL[113]	MAX	1	0.0184	6.26	82.14
	MIN				0.599
	AVG				15.29
CICIoT2023[59]	MAX	43	0.0203	6.56	566.21
	MIN		0.0002	6.19	0.0580
	AVG		0.0065	6.27	15.287
UNSW[96]	MAX	33	2.1596	7.63	709.11
	MIN		0.0005	5.14	0.0003
	AVG		0.1535	6.16	238.65
Yourthings[189]	MAX	75	0.0239	6.25	912.56
	MIN		0.0001	6.20	2.2200
	AVG		0.0046	6.25	151.56
HorusEye[70]	MAX	14	0.0145	6.25	633.06
	MIN		0.0001	6.24	13.663
	AVG		0.0063	6.25	93.628
<i>iDetector</i>	MAX	15	0.0261	6.25	10647.3
	MIN		0.0001	5.61	0.03260
	AVG		0.0054	6.23	1234.44

Table 5.6: **Overhead Analysis of proposed approach across different datasets.**

none of the compression files used in CDM calculations. The HCRL dataset exhibits a minimal disk usage of 0.0184MB, reflecting a scenario with a single device. In contrast, the CICIoT2023 dataset, which involves 43 devices, displays a more complex range with a maximum disk usage of 0.0203MB and a minimum of 0.0002MB, averaging around 0.0065MB. The UNSW dataset, with 33 devices, shows a substantially higher maximum disk usage at 2.1596MB, although its average usage is much lower at 0.1535MB. The Yourthings dataset, with 75 devices, has a relatively low average disk usage of 0.0046MB, with a maximum of 0.0239MB. Similarly, HorusEye, with 14 devices, maintains a low average disk usage of 0.0063MB, with a maximum of 0.0145MB. Finally, the iDetector dataset, with 15 devices, presents a maximum disk usage of 0.0261MB and an average of 0.0054MB. In general, if the total number of devices is m , with a total number of baseline n , then we can find the total disk usage needed using the following equation:

$$Total\ Disk\ Usage = \sum_m \sum_n size(BNF) \quad (5.28)$$

CPU & Memory Consumption. After generating baselines for offline and online detection in two cases, we analysed CPU consumption running the proposed approach as a background process that may add to the deployed hardware resources. CPU consumption remains relatively consistent for offline and online detection across all datasets, with minimal variation. The average CPU consumption is around 6 – 7% for all datasets, with the UNSW dataset showing a slightly higher minimum and maximum CPU consumption (between 5.14% and 7.63%). There is no memory consumption as the CDM uses already saved baselines and does not save compression files.

Latency. A complete cycle of detection happens in the background with no manual intervention. Baseline establishment, however, requires manual effort (Section 5.6.2). After generating baselines in two cases for offline and online detection, we analysed latency (time lapsed) for the complete detection cycle. We performed online detection for our testbed and replicated three classes of attacks: Information gathering, DDoS and botnet. We used four public datasets for offline evaluation, *i.e.*, HCRL, CICIOT2023, UNSW, and yourthings datasets. iDetector exhibits the highest average latency (1234.44s) with a maximum latency of 10647.34s, which indicates a substantial delay in processing compared to other datasets. Yourthings also shows a high average latency of 151.56s, much lower than iDetector. The UNSW dataset also has a relatively high latency, averaging 238.65s. While CICIOT2023 shows the lowest average latency (15.29s), HCRL follows closely (15.29s).

Robustness of *iDetector*

We now analyse the security properties of iDetector when functioning under an adversarial model. The adversary’s overarching goal is to generate flows of a target malicious class without being detected by iDetector. We consider three categories of adversaries considering their knowledge about iDetector and the modules employed in the detection phase.

No Knowledge about *iDetector* nor compression algorithm. In the weakest threat model, the adversary knows nothing about the presence of iDetector in the network infrastructure nor the details of the compression algorithm used by the detection phase leveraging our system’s capabilities. In such a case, as shown in the sections above, the detection phase using the encoded and baseline modules can accurately identify various forms of malicious traffic.

***iDetector*-aware adversary.** In the second case, we consider an adversary that is aware of the deployment of iDetector in the network infrastructure but who is unaware of the particular detection models being used to filter the network for specific classes of traffic. In this case, the adversary may attempt to launch DoS attacks. An adversary may attempt to compromise the availability of our iDetector. For example, An adversary may try to mount a DoS attack based on transmitting packets with random IP addresses, forcing iDetector to keep track of multiple dummy flows and waste the shift slide window. To mitigate such a threat, iDetector can temporarily prevent retrieving baselines when it detects unusual bursts of traffic or reconfigure segmentation encoding on the fly to store larger (yet less accurate) flow signatures to increase the number of measured flows.

***iDetector* and compression algorithm adversary.** The third adversary we consider an adversary that is aware of the deployment of compression algorithm used by iDetector. Apart from an adversary’s attempts to evade or compromise the availability of our system, such an adversary aims to leverage adversarial ML techniques. i) Training time, where an adversary aims at manipulating the training set used by the iDetector model by inserting specific samples that alter the encoded baselines of the detection. ii) Test-time, where an adversary aims to evade detection by crafting traffic samples so that these fool the detection during its similarity collocation. Or they are replacing the dictionary used in the compression algorithm in a way that fools the detection during its similarity collocation. In both cases, we assume all device baselines will be saved in the Trusted Execution Environment (TEE). While adversaries may employ sophisticated methods to attempt

to compromise a TEE, robust security measures and designs aim to provide high protection against such attacks.

5.6 Discussion

Here, we will detail the key takeaways and lessons from our results while also highlighting the limitations of our evaluation.

5.6.1 Takeaways and Lessons

As with any security framework evaluation, a number of questions have yet to be answered. Some of these include:

How can we maintain minimal false positive and false negative rates? False positives and negatives in detection can happen when outdated baselines or traffic encoding features fail to capture differences (as seen in Section 5.5.3). As cyberattacks and device behaviours constantly evolve, crucial [59], unresolved questions arise: How often should baselines and detection features be updated?

How often do we need to update devices' baselines? Outdated baselines can lead to more false positives and negatives, raising further questions: What minimum traffic is required to create baselines? What factors, like user interactions, time changes, or manufacturer updates, should be considered when updating them? Hence, improving baseline mechanisms is crucial for future work.

How often do we need to update features used in detection? False positives and negatives can rise when traffic encoding features fail to capture differences. This raises crucial questions for future work: What features can be extracted to meet design requirements, and how can we create an automated framework that accurately represents a device's normal behaviour across different dimensions?

How usable and scalable is the approach? We envision deploying the proposed approach on edge and gateway devices, which would require firmware upgrades or could be included in future devices. Coordination across these devices over the Internet is necessary. Our evaluation shows that the approach has minimal computational and resource demands, regardless of the number of IoT devices under continuous detection. We expect no loss in detection performance or additional processing overhead from this constant detection.

5.6.2 Limitations

Here, we discuss the limitations of iDetector, acknowledging factors that may have influenced the results and suggesting areas for improvement in future research:

iDetector can be used for Wi-Fi-connected devices or route traffic via a Wi-Fi hub or IP gateway. It could also be adapted for other short-range protocols such as Bluetooth [169], Z-Wave [256], or other IEEE 802.15.4-compliant radio frequency standards, as long as it can monitor device packets

during Internet data transmission and distinguish between plain text headers and encrypted payloads. It is crucial to consider each protocol's nature of traffic and adjust the approach accordingly.

We envision deploying iDetector on edge devices such as home gateways or ISP routers, requiring minimal technical knowledge from users but physical intervention from network operators. Creating device baselines takes at least a day per device, which could be more practical for users who frequently add or replace IoT devices. Future work will focus on automating this process.

SAX-encoding has limitations in handling time-series data due to its sensitivity to shifts along the time axis, which can lead to inconsistent and unreliable representations. SAX-encoding also relies on Piecewise Aggregate Approximation (PAA) for dimensionality reduction, which can result in the loss of crucial information, particularly in data with high-frequency components. While extensions such as Adaptive SAX (aSAX) and Weighted SAX (wSAX) attempt to address these issues, Generative AI models such as GANs and VAEs offer promising alternatives. Future work will explore whether these models can create more robust and adaptable representations that overcome SAX-encoding current limitations.

5.7 Conclusion

In this chapter, we present a malicious traffic detection solution utilising Symbolic Aggregate Approximation (SAX) and compression-based dissimilarity operating in unsupervised mode. The model verified the 80% validated traffic from iHybrid and identified any malicious activity, improving real-time detection of malicious activity and addressing evolving malicious events at the network's edge. With this model, we concluded that robust device identification can enhance IoT anomaly detection, but such device identification can not work alone. A robust IoT device identification must be used with a detection model. By this we mark the conclusion of this Thesis.

Chapter 6

Conclusion

In Chapter 2, we identified a critical gap: the role of robust and accurate device identification as a prerequisite for improving IoT anomaly detection. In Chapter 3, we examined how machine learning-based IoT device identification models degrade over time and provided recommendations for improvement. Building on these recommendations, in Chapter 4 we introduced IoT device fingerprints based on modes of operation and developed the hybrid traffic identification model, iHybrid, which combined these fingerprints with behavioural profiles (MUDs), Access Control Lists (ACLs), and functional destination blocklisting to validate incoming traffic. In Chapter 4, we showed that iHybrid successfully validated up to 80% of incoming traffic. However, we are uncertain whether 80% of the traffic is truly free of malicious activity or if any verified traffic is mimicked legitimate behaviours. To address this, in Chapter 5 we presented an anomaly detection model to investigate further that verified traffic and identify any malicious activity. With this model, we concluded that robust device identification could enhance IoT anomaly detection, but we must use such device identification with a detection model. By this we mark the conclusion of our Thesis. In this chapter, we summarise the thesis achievements, discuss the security applications of our contributions, highlight the limitations encountered, and outline potential directions for future work.

6.1 Summary of Thesis Achievements

Building upon the literature review for anomaly detection in IoT environments presented in Chapter 2, we identified a critical gap: the role of robust and accurate device identification as a prerequisite for improving IoT anomaly detection. This gap motivated the rigorous evaluations of IoT device identification methods in Chapter 3, to understand how we can enhance identification accuracy. Building on these insights, in Chapter 4 we introduce a novel IoT device identification fingerprinting approach, which we combined with other identification models to validate incoming traffic from IoT devices. Finally, in Chapter 5, we develop a detection model capable of distinguishing normal traffic that mimics legitimate behaviour. By addressing these interconnected aspects, this thesis advances IoT security significantly. The main achievements of this research are outlined below:

- **Comprehensive Evaluation of Machine Learning Models for IoT Device Identification.** Our objective was to critically evaluate machine learning-based solutions for IoT device identification, focusing on their effectiveness and reliability in identifying both devices and their associated traffic. In Chapter 3, we achieved this by evaluating these models using highly curated datasets and representative features, uncovering their strengths and limitations. This evaluation provided actionable insights into addressing issues such as randomness (*e.g.*, spoofing) and high false alarm rates, paving the way for improved performance in operational deployments.
- **Design and Development of a Hybrid Model for IoT Traffic Identification.** Our objective was to develop a robust solution for IoT traffic identification that could overcome the limitations of standalone identification models, particularly their susceptibility to false alarms. In Chapter 4, we achieved this by designing a hybrid model that integrates device fingerprints, behavioural profiles (*e.g.*, Manufacturer Usage Description), and functional blocklists. This hybrid approach ensures robust and efficient traffic analysis, enabling the model to handle identification challenges more effectively in real-world settings.
- **Introduction of an Advanced Model for IoT Malicious Traffic Detection.** Our objective was to create a detection model capable of identifying malicious IoT traffic in real-time, even in the presence of evolving attack patterns and mimicry by attackers. In Chapter 5, we achieved this by leveraging Symbolic Aggregate Approximation (SAX) and compression-based dissimilarity in an unsupervised framework. This solution dynamically validates network traffic, significantly enhancing anomaly detection at the network's edge and effectively addressing predefined and previously unseen malicious events.

These achievements highlight the advancements made in IoT security through this research, laying a foundation for practical implementations. In the next section, we explore how we can apply these contributions to real-world security applications.

6.2 Security Applications

The contributions of this thesis have practical applications in various security contexts, offering enhanced solutions for IoT device identification and anomaly detection that can be integrated into real-world solutions to strengthen IoT security frameworks. Here are some of these applications:

- **IoT Device Management Solutions.** We can utilise the insights gained from evaluating machine learning models for device identification to enhance IoT device management solutions. This enhancement includes improving device onboarding, monitoring, and categorisation, which can streamline network management and reduce administrative overhead.
- **Home Gateway Security Solutions.** We can implement the hybrid model for IoT device traffic identification on home gateways in residential security solutions to improve monitoring and threat detection. These solutions safeguard smart home environments from potential attacks and unauthorised access.

- **IoT Enforcement Solutions.** The developed anomaly detection techniques, including the hybrid model and iDetector, can be integrated into enforcement solutions to ensure more robust and real-time protection against emerging IoT threats. These solutions can be deployed across various environments, such as smart offices, industrial IoT environments, and critical infrastructure, to enforce stricter security measures and maintain operational integrity.
- **Edge Device Security Solutions.** iDetector’s real-time anomaly detection capabilities can be applied to edge devices in IoT environment, ensuring that security measures are in place for devices operating at the network’s periphery. This is crucial for protecting edge computing environments where immediate threat detection and response are necessary.

While these applications demonstrate the practical value of the proposed contributions, they also reveal limitations within the broader scope of this thesis, which we must carefully consider to inform future advancements. In the next section, we discuss these limitations.

6.3 Limitations

Throughout each chapter, we identified and discussed specific limitations relevant to the focus of each chapter. Like any research, this thesis encountered several limitations associated with its broader scope:

- **Threat Modelling, Adversarial Machine Learning, Privacy, and Security**
 - **Threat Model.** The threat model for IoT device identification must consider adaptive adversarial tactics, including adversarial drift and reinforcement learning-based evasion strategies, which can degrade model performance over time. For our malicious traffic classification, it is crucial to expand the threat model to include the device features, baselines, and dictionaries used in compression. This broader scope ensures the model remains resilient against attacks targeting these specific components.
 - **Adversarial Machine Learning.** Adversarial machine learning presents a major challenge, as attackers can manipulate input features, baselines, and compression dictionaries to evade detection or mislead classification. In the case of IoT device identification, adversaries may target both the features and the baselines used in models such as iDetector and iHybrid, causing adversarial drift and exploiting vulnerabilities in the compression process. Ensuring resilience against adversarial perturbations, poisoning attacks, and evasion techniques is crucial for maintaining the reliability of ML-based IoT device identification and malicious classification solutions. This includes addressing attacks that manipulate the dictionary used in compression, which can affect the model’s performance and ability to classify devices accurately. Developing robust countermeasures, such as adversarial training and uncertainty-aware models, becomes essential to ensure the system can defend against adaptive adversaries and maintain its effectiveness over time.
 - **Privacy.** The use of sensitive or personal data in training models, particularly for IoT

device identification and traffic classification, raises significant privacy concerns. As the model relies on device features and other sensitive attributes, there is a risk of exposing private information if the data is not adequately protected. Ensuring compliance with data protection regulations, such as GDPR or other relevant frameworks, while handling and processing data is essential but challenging. Integrating privacy-preserving techniques, such as anonymisation or feature obfuscation, must be considered to mitigate risks. Additionally, as adversarial attacks could target privacy features or the data itself, it is crucial to adopt robust privacy measures without compromising the model's effectiveness. Ensuring that privacy measures align with security objectives is a delicate balance that requires careful consideration during model development.

- **Security.** Implementing new security solutions, particularly those involving IoT device identification and malicious classification, may introduce unforeseen vulnerabilities or attack vectors. For example, adversarial machine learning techniques could exploit weaknesses in the models' features, baselines, and compression dictionaries, potentially degrading their performance or leading to false positives/negatives. Rigorous testing and validation are necessary to identify and mitigate potential security risks associated with the new models, ensuring that countermeasures such as adversarial training effectively maintain model robustness. Ongoing evaluation must consider the impact of adversarial tactics such as adversarial drift, reinforcement learning-based evasion strategies, and poisoning attacks on the long-term security of the model.
- **Data Availability and Quality:** Although many public IoT datasets are available, they often need more detailed and diverse information for effective anomaly detection, limiting their reuse for our research. We created our testbed to generate more representative data to overcome this challenge. Despite this effort, we faced constraints due to the university's ethical hacking code of conduct, which restricted the range and types of cyberattacks we could simulate. Moreover, we encountered issues with data imbalance, as our datasets had a disproportionate number of normal versus anomalous instances. This imbalance hindered our model's ability to detect rare but critical anomalies, affecting our models' overall accuracy and generalisability.
- **Complexity of IoT Environments:**
 - IoT environments encompass various devices with varied functionalities and communication patterns. Developing a one-size-fits-all detection solution that effectively covers all types can be challenging.
 - IoT devices exhibit diverse and evolving behaviour over time. Solutions trained on historical data may struggle to adapt to new or changing device behaviour patterns.
- **Scalability:** As the number of IoT devices and the volume of data increase, maintaining the performance and efficiency of anomaly detection models can become challenging. Scaling these models to manage large-scale deployments while ensuring real-time performance may

necessitate additional validation and evaluation processes.

- **Real-Time Processing:** Real-time anomaly detection demands rapid processing and analysis of data. In some instances, we encountered high latency or delays in detection, which can affect the effectiveness of security measures, particularly in time-sensitive applications. Nonetheless, we adhere to the principle of "Better late than never," believing that detecting anomalies late is preferable to not at all.
- **Integration with Existing Solutions:** As we use various behavioural identification methods, integrating our anomaly detection models with existing IoT infrastructure and security systems may present technical challenges (*e.g.*, MUDgee). Compatibility issues could potentially affect the smooth deployment and operation of the proposed solutions.
- **Generalisation Across Environments:** Despite testing and developing our framework in various lab environments, we recognise that it remains within a controlled lab context. We need to generalise it to other environments better or use cases, such as actual smart homes. We should carefully evaluate the applicability of our proposed solutions across different IoT deployments and settings.

These limitations provide valuable insights that will guide future work. The following section highlights areas for improvement and potential directions for further research to address these challenges.

6.4 Future Work

Building on the limitations discussed in the previous section, in this section, we outline potential directions for future work where further advancements can address existing challenges and expand upon the contributions of this thesis:

- **Enhancing Robustness Against Adversarial Evasion in ML-based IoT Solutions.** Future research should explore a broader range of adversarial machine learning evasion techniques to enhance the robustness of iDetector and iHybrid. In addition to adversarial perturbations and poisoning attacks, methods such as generative adversarial networks (GANs) for crafting evasive traffic, reinforcement learning-based evasion strategies, and adaptive adversarial attacks that exploit temporal variations in IoT behaviour must be considered. Developing countermeasures against these techniques requires integrating adversarial training with dynamic feature augmentation, enabling models to recognise and adapt to evolving attack patterns. Moreover, leveraging ensemble learning and uncertainty-aware models could mitigate the impact of adversarial drift. Exploring decentralised and federated learning approaches may also strengthen defences by reducing the risk of targeted model manipulation. Future research should further investigate explainable AI (XAI) methods to provide deeper insights into adversarial impact, allowing for real-time detection and adaptation to adversarially manipulated inputs.
- **Integration and Scalability.** Future research should focus on integrating the proposed

anomaly detection models into larger and more diverse IoT environments to evaluate their scalability and effectiveness across different types of networks and device configurations. This evaluation will help refine the models and ensure their applicability in various contexts.

- **Adaptive and Self-Learning Solutions.** Developing adaptive and self-learning anomaly detection solutions that can continuously improve their performance based on evolving threat landscapes and changing device behaviours is a key area for future work. These adaptive and self-learning involve incorporating advanced machine-learning techniques and real-time learning algorithms.
- **Enhanced Real-Time Analytics.** Exploring ways to enhance further real-time analytics capabilities to handle high volumes of data and complex network environments can improve the efficiency and accuracy of anomaly detection. This exploration includes optimising data processing and reducing latency in threat detection.
- **User-Centric Security Solutions.** Investigating user-centric approaches for IoT security, such as personalised threat detection based on individual user behaviour and preferences, can provide more tailored and adequate security measures.
- **Generative AI and Large Language Models (LLMs).** Future research should explore the application of Generative AI and LLMs to learn and model the behavioural patterns of IoT devices more effectively. These technologies can analyse complex and diverse device behaviours to develop more accurate representations of normal activity and better identify anomalies. By leveraging the pattern recognition capabilities of LLMs, researchers can enhance the detection of subtle deviations from expected behaviour, improving overall security and device management.
- **Standardisation.** Future work should also focus on collaborating with industry stakeholders to develop standards (*e.g.*, IETF) and best practices for implementing anomaly detection in IoT solutions. This collaboration can promote wider adoption of the proposed solutions and ensure consistency in security measures.
- **Comprehensive Evaluation.** Conducting comprehensive evaluations of the proposed solutions in real-world settings, including pilot deployments and field studies, will help validate their effectiveness and identify areas for improvement.

6.5 Final Thoughts on Anomaly Detection in IoT Environments

We conclude this Thesis with some final thoughts on anomaly detection in the broader context of IoT Environments. The ever-expanding deployment of IoT devices in consumer environments introduces a vast attack surface as millions of connected devices become increasingly vulnerable to exploitation. The motivation behind this Thesis stemmed from the urgent need to address the security challenges posed by diverse and evolving IoT behaviours. Through systematic exploration, we have demonstrated that accurate and robust device identification is fundamental to enhancing anomaly detection in IoT environments.

This Thesis established that existing machine learning-based identification solutions, though promising, suffer from limitations such as high false alarm rates and sensitivity to randomness, including spoofing. By introducing the hybrid identification model, iHybrid, we bridged this gap by integrating device fingerprints, behavioural profiles, and destination blocklists. This solution improved validating network traffic, enhancing robustness and scalability in operational contexts. However, the potential for mimicked legitimate behaviours necessitated a further layer of anomaly detection. To this end, we developed an unsupervised malicious traffic detection model leveraging Symbolic Aggregate Approximation (SAX) and compression-based dissimilarity, enabling the detection of nuanced and evolving threats at the network edge. This Thesis shows the importance of accurate device identification for robust anomaly detection, highlighting that more than one approach can be required in different environments. Combining robust identification with anomaly detection, we presented a comprehensive framework for mitigating the risks associated with IoT-based attacks. Our contributions advance the state of the art in IoT security and pave the way for practical deployments in smart home and office environments.

While this Thesis has made significant advances, limitations such as the reliance on curated datasets and the evolving nature of IoT threats remain. Future work could explore more adaptive, real-time models capable of self-updating in response to new attack patterns alongside deeper investigations into lightweight anomaly detection techniques tailored to resource-constrained IoT devices. Ultimately, this Thesis reaffirms the critical importance of securing IoT networks against emerging threats and sets the foundation for future innovation in anomaly detection and device identification.

Bibliography

- [1] 2Smart. IoT in the Consumer Electronics Market: Trends for 2024. <https://2smart.com>. Accessed: 2024-10-16. 2024.
- [2] Abbas Acar et al. "Peek-a-Boo: I see your smart home activities, even encrypted!" In: *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2020, pp. 207–218.
- [3] Ahmadreza Ahmadi, Arash Shams, and Rasool Jalili. *CICFlowMeter*. <https://github.com/ahlashkari/CICFlowMeter>. 2022.
- [4] Dilawer Ahmed. *Analyzing the Feasibility and Generalizability of Fingerprinting Internet of Things Devices*. <https://github.com/dilawer11/iot-device-fingerprinting>. [Online; accessed 01.02.2023]. 2022.
- [5] Dilawer Ahmed, Anupam Das, and Fareed Zaffar. "Analyzing the feasibility and generalizability of fingerprinting Internet of Things devices". In: *Proceedings on Privacy Enhancing Technologies* 2022.2 (2022).
- [6] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. "A survey of network anomaly detection techniques". In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31.
- [7] Mosabber Uddin Ahmed and Danilo P Mandic. "Multivariate multiscale entropy analysis". In: *IEEE signal processing letters* 19.2 (2011), pp. 91–94.
- [8] Mosabber Uddin Ahmed and Danilo P Mandic. "Multivariate multiscale entropy: A tool for complexity analysis of multichannel data". In: *Physical Review E* 84.6 (2011), p. 061918.
- [9] Mohammed M Alani and Ernesto Damiani. "XRecon: An explainable IoT reconnaissance attack detection system based on ensemble learning". In: *Sensors* 23.11 (2023), p. 5298.
- [10] Fahad Alotaibi and Sergio Maffei. "Mateen: Adaptive Ensemble Learning for Network Anomaly Detection". In: *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*. 2024, pp. 215–234.
- [11] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.
- [12] Omar Alrawi et al. "Sok: Security evaluation of home-based iot deployments". In: *2019 IEEE symposium on security and privacy (sp)*. IEEE. 2019, pp. 1362–1380.
- [13] Nesrine Ammar, Ludovic Noirie, and Sébastien Tixeuil. "Autonomous identification of iot device types based on a supervised classification". In: *ICC 2020-2020 IEEE international conference on communications (ICC)*. IEEE. 2020, pp. 1–6.
- [14] Ampere Analysis. *Subscribers to ad-supported tiers have surpassed 100m in the US*. Accessed: 2024-10-30. 2024. URL: <https://www.ampereanalysis.com/press/release/dl/subscribers-to-ad-supported-tiers-have-surpassed>.
- [15] Eirini Anthi et al. "A supervised intrusion detection system for smart home IoT devices". In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 9042–9053.
- [16] Manos Antonakakis et al. "Understanding the mirai botnet". In: *26th USENIX security symposium (USENIX Security 17)*. 2017, pp. 1093–1110.
- [17] Noah Apthorpe, Dillon Reisman, and Nick Feamster. "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic". In: *arXiv preprint arXiv:1705.06805* (2017).
- [18] Noah Apthorpe et al. "Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic". In: *arXiv preprint arXiv:1708.05044* (2017).

- [19] Daniel Arp et al. “Dos and don’ts of machine learning in computer security”. In: 31st USENIX Security Symposium (USENIX Security 22). 2022, pp. 3971–3988.
- [20] ASVIN. More Cyber Attacks on IoT Devices. Accessed: 16-Dec-2024. 2024. URL: <https://asvin.io/more-cyber-attacks-on-iot-devices/>.
- [21] Luigi Atzori, Antonio Iera, and Giacomo Morabito. “The Internet of Things: A Survey”. In: Computer Networks 54.15 (2010), pp. 2787–2805. DOI: [10.1016/j.comnet.2010.05.010](https://doi.org/10.1016/j.comnet.2010.05.010). URL: <https://doi.org/10.1016/j.comnet.2010.05.010>.
- [22] Tom Auld, Andrew W Moore, and Stephen F Gull. “Bayesian neural networks for internet traffic classification”. In: IEEE Transactions on neural networks 18.1 (2007), pp. 223–239.
- [23] Yichiet Aun, Yen-Min Khaw, and Ming-Lee Gan. “A holistic IoT device classification approach through spatial & temporal behaviors modelling”. In: Telecommunication Systems 79.4 (2022), pp. 515–528.
- [24] Eray Balkanli, Jander Alves, and A Nur Zincir-Heywood. “Supervised learning to detect DDoS attacks”. In: 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS). IEEE. 2014, pp. 1–8.
- [25] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. “Towards periodicity based anomaly detection in SCADA networks”. In: International Conference on Emerging Technologies & Factory Automation. IEEE. 2012, pp. 1–4.
- [26] Vic Barnett and Toby Lewis. “Outliers in statistical data”. In: Wiley Series in Probability and Statistics. (1984).
- [27] Diogo Barradas et al. “FlowLens: Enabling Efficient Flow Classification for ML-based Network Security Applications”. In: Proceedings of the 28th Network and Distributed System Security Symposium. 2021.
- [28] Vadim Bashurov and Paul Safonov. “Anomaly detection in network traffic using entropy-based methods: application to various types of cyberattacks.” In: Issues in Information Systems 24.4 (2023).
- [29] Roman Beltiukov et al. “In search of netunicorn: A data-collection platform to develop generalizable ml models for network security problems”. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. 2023, pp. 2217–2231.
- [30] Randeep Bhatia et al. “Unsupervised machine learning for network-centric anomaly detection in iot”. In: Proceedings of the 3rd acm conext workshop on big data. 2019, pp. 42–48.
- [31] Vishwajeet Bhosale, Lorenzo De Carli, and Indrakshi Ray. “Detection of Anomalous User Activity for Home IoT Devices”. In: International Conference on Internet of Things, Big Data and Security (IoTBDS) (2021).
- [32] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. “Network anomaly detection: methods, systems and tools”. In: Ieee communications surveys & tutorials 16.1 (2013), pp. 303–336.
- [33] Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. “NADO: network anomaly detection using outlier approach”. In: Proceedings of the 2011 International Conference on Communication, Computing & Security. 2011, pp. 531–536.
- [34] Bitdefender. 2023 IoT Security Landscape Report. Tech. rep. Accessed: 2024-08-16. 2023. URL: <https://www.bitdefender.com/files/News/CaseStudies/study/429/2023-IoT-Security-Landscape-Report.pdf>.
- [35] Steven L. Blake et al. An Architecture for Differentiated Services. RFC 2475. Dec. 1998. DOI: [10.17487/RFC2475](https://www.rfc-editor.org/info/rfc2475). URL: <https://www.rfc-editor.org/info/rfc2475>.
- [36] Sergey Bratus et al. “Active behavioral fingerprinting of wireless devices”. In: Proceedings of the first ACM conference on Wireless network security. 2008, pp. 56–61.
- [37] Dominik Breitenbacher et al. “HADES-IoT: A practical host-based anomaly detection system for IoT devices”. In: Proceedings of the 2019 ACM Asia conference on computer and communications security. 2019, pp. 479–484.
- [38] Vladimir Brik et al. “Wireless device identification with radiometric signatures”. In: Proceedings of the 14th ACM international conference on Mobile computing and networking. 2008, pp. 116–127.
- [39] Olivier Brun et al. “Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments”. In: Security in Computer and Information Sciences: First International ISCIS Security Workshop 2018. Springer International Publishing. 2018, pp. 79–89.

- [40] Arthur Callado et al. "A survey on internet traffic identification". In: *IEEE communications surveys & tutorials* 11.3 (2009), pp. 37–52.
- [41] Liangyue Cao, Alistair Mees, and Kevin Judd. "Dynamics from multivariate time series". In: *Physica D: Nonlinear Phenomena* 121.1-2 (1998), pp. 75–88.
- [42] Z Berkay Celik, Gang Tan, and Patrick D McDaniel. "Iotguard: Dynamic enforcement of security and safety policy in commodity IoT." In: *NDSS*. 2019.
- [43] Philip K. Chan, Matthew V. Mahoney, and Muhammad H. Arshad. *A Machine Learning Approach to Anomaly Detection*. Tech. rep. CS-2003-06. Melbourne, FL: Department of Computer Sciences, Florida Institute of Technology, 2003. URL: https://repository.fit.edu/ces_faculty/194/.
- [44] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.
- [45] Vasilis Chatzigiannakis et al. "Data fusion algorithms for network anomaly detection: classification and evaluation". In: *International Conference on Networking and Services (ICNS'07)*. IEEE. 2007, pp. 50–50.
- [46] Parminder Chhabra et al. "Distributed spatial anomaly detection". In: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE. 2008, pp. 1705–1713.
- [47] Haotian Chi et al. "Pfirewall: Semantics-aware customizable data flow control for home automation systems". In: *arXiv preprint arXiv:1910.07987* (2019).
- [48] Rajarshi Roy Chowdhury, Azam Che Idris, and Pg Emeroylariffion Abas. "A deep learning approach for classifying network connected IoT devices using communication traffic characteristics". In: *Journal of Network and Systems Management* 31.1 (2023), p. 26.
- [49] CIRALabs. *CIRA secure-IoT-Home-Gateway*. URL: <https://github.com/CIRALabs/Secure-IoT-Home-Gateway>. (accessed: 31.08.2020).
- [50] Cisco. *Cisco 2018 Annual Cybersecurity Report*. Technical Report. (accessed: 24.08.2020).
- [51] Cisco Systems. *Configuring NetFlow-Lite Software Configuration Guide, Release 15.0*. 2011.
- [52] Benoît Claise. *Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information*. RFC 5101. Available: <https://www.rfc-editor.org/rfc/rfc5101>. 2008.
- [53] Cloudflare. *DNS Load Balancing: What Is It and How Does It Work?* Accessed: 2024-10-30. 2020. URL: <https://www.cloudflare.com/learning/dns/dns-load-balancing/>.
- [54] *Consumer IoT Market Trends and Key Players*. <https://www.mordorintelligence.com>. Accessed: 2024-10-16. 2023.
- [55] Andrew A Cook, Göksel Mısırlı, and Zhong Fan. "Anomaly detection for IoT time-series data: A survey". In: *IEEE Internet of Things Journal* 7.7 (2019), pp. 6481–6494.
- [56] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.
- [57] Emma Crockett. *85 Top IoT Devices*. <https://www.datamation.com/mobile/85-top-iot-devices/>. Accessed: June 30, 2024. 2023.
- [58] Ivan Cvitić et al. "Ensemble machine learning approach for classification of IoT devices in smart home". In: *International Journal of Machine Learning and Cybernetics* 12.11 (2021), pp. 3179–3202.
- [59] Canadian Institute for Cybersecurity. *Canadian Institute for Cybersecurity datasets*. "<https://www.unb.ca/cic/datasets/ids-2017.html>". [Online; accessed 12.04.2021].
- [60] Alexander D'Amour et al. "Underspecification presents challenges for credibility in modern machine learning". In: *Journal of Machine Learning Research* 23.226 (2022), pp. 1–61.
- [61] Sajjad Dadkhah et al. "Towards the development of a realistic multidimensional IoT profiling dataset". In: *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*. IEEE. 2022, pp. 1–11.
- [62] Data Alliance. "IoT Security: Key Internet of Things Trend for 2023". In: (2023). Accessed: 2024-08-16. URL: <https://www.data-alliance.net/iot-security-key-internet-of-things-trend-for-2023/>.
- [63] DataProt. *Internet of Things Statistics for 2024*. Accessed: 16-Dec-2024. 2024. URL: <https://dataprot.net/statistics/iot-statistics/>.
- [64] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 233–240.

- [65] Jyoti Deogirikar and Amarsinh Vidhate. "Security attacks in IoT: A survey". In: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). IEEE. 2017, pp. 32–37.
- [66] Loh Chin Choong Desmond et al. "Identifying unique devices through wireless fingerprinting". In: Proceedings of the first ACM conference on Wireless network security. 2008, pp. 46–55.
- [67] Hailun Ding et al. "{AIRTAG}: Towards Automated Attack Investigation by Unsupervised Learning with Log Texts". In: 32nd USENIX Security Symposium (USENIX Security 23). 2023, pp. 373–390.
- [68] Pedro Domingos. "A few useful things to know about machine learning". In: Communications of the ACM 55.10 (2012), pp. 78–87.
- [69] Shuaike Dong et al. "Your smart home can't keep a secret: Towards automated fingerprinting of iot traffic". In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. 2020, pp. 47–59.
- [70] Yutao Dong et al. "{HorusEye}: A Realtime {IoT} Malicious Traffic Detection Framework using Programmable Switches". In: 32nd USENIX Security Symposium (USENIX Security 23). 2023, pp. 571–588.
- [71] Rohan Doshi, Noah Apthorpe, and Nick Feamster. "Machine learning ddos detection for consumer internet of things devices". In: 2018 IEEE Security and Privacy Workshops (SPW). IEEE. 2018, pp. 29–35.
- [72] Min Du et al. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning". In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 2017, pp. 1285–1298.
- [73] Chenxin Duan et al. "PINBALL: Universal and Robust Signature Extraction for Smart Home Devices". In: 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE. 2021, pp. 1–9.
- [74] Easylist. Easylist: <https://easylist.to/>. [Online; accessed 13.03.2025].
- [75] Eleazar Eskin. "Anomaly detection over noisy data using learned probability distributions". In: (2000).
- [76] European Union. (General Data Protection Regulation). Accessed: 2024-10-30. 2016. URL: <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.
- [77] Linna Fan et al. "AutoIoT: Automatically Updated IoT Device Identification With Semi-Supervised Learning". In: IEEE Transactions on Mobile Computing (2022).
- [78] Johann Faouzi and Hicham Janati. "pyts: A Python Package for Time Series Classification." In: J. Mach. Learn. Res. 21 (2020), pp. 46–1.
- [79] Xuan Feng et al. "Acquisitional rule-based engine for discovering {Internet-of-Things} devices". In: 27th USENIX security symposium (USENIX Security 18). 2018, pp. 327–341.
- [80] Ralph Foorthuis. "On the Nature and Types of Anomalies: A Review". In: arXiv preprint arXiv:2007.15634 (2020).
- [81] Jason Franklin et al. "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting." In: USENIX Security Symposium. Vol. 3. 2006. 2006, pp. 16–89.
- [82] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. "{HAWatcher}:{Semantics-Aware} anomaly detection for appified smart homes". In: 30th USENIX Security Symposium (USENIX Security 21). 2021, pp. 4223–4240.
- [83] Chuanpu Fu, Qi Li, and Ke Xu. "Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis". In: arXiv preprint arXiv:2301.13686 (2023).
- [84] Chuanpu Fu et al. "Realtime robust malicious traffic detection via frequency domain analysis". In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021, pp. 3431–3446.
- [85] Shekhar R Gaddam, Vir V Phoha, and Kiran S Balagani. "K-Means+ ID3: A novel method for supervised anomaly detection by cascading K-Means clustering and ID3 decision tree learning methods". In: IEEE transactions on knowledge and data engineering 19.3 (2007), pp. 345–354.
- [86] Sahil Garg et al. "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks". In: IEEE Transactions on Network and Service Management 16.3 (2019), pp. 924–935.
- [87] N. D. Geethakumari, D. A. S. Kumar, and A. V. Ramani. "A Survey on Internet of Things Architectures and Security". In: International Journal of Computer Applications 177.30 (2021), pp. 10–16. DOI: [10.5120/ijca2021921724](https://doi.org/10.5120/ijca2021921724). URL: <https://doi.org/10.5120/ijca2021921724>.
- [88] Ryan M Gerdes et al. "Device Identification via Analog Signal Fingerprinting: A Matched Filter Approach". In: ().

- [89] Federico Giannoni, Marco Mancini, and Federico Marinelli. “Anomaly detection models for IoT time series data”. In: *arXiv preprint arXiv:1812.00890* (2018).
- [90] Raspberry Pi group. *Setting up a Routed Wireless Access Point*. <https://www.raspberrypi.com/documentation>. [Online; accessed 25.08.2021].
- [91] Guofei Gu et al. “Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection”. In: (2008).
- [92] Haixuan Guo, Shuhan Yuan, and Xintao Wu. “Logbert: Log anomaly detection via bert”. In: *2021 international joint conference on neural networks (IJCNN)*. IEEE. 2021, pp. 1–8.
- [93] Isabelle Guyon and André Elisseeff. “An introduction to variable and feature selection”. In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.
- [94] Ibbad Hafeez et al. “IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge”. In: *IEEE Transactions on Network and Service Management* 17.1 (2020), pp. 45–59.
- [95] Ayyoob Hamza et al. “Clear as MUD: Generating, validating and applying IoT behavioral profiles”. In: *Proceedings of the 2018 Workshop on IoT Security and Privacy*. 2018, pp. 8–14.
- [96] Ayyoob Hamza et al. “Detecting volumetric attacks on iot devices via sdn-based monitoring of mud activity”. In: *Proceedings of the 2019 ACM Symposium on SDN Research*. 2019, pp. 36–48.
- [97] Ayyoob Hamza et al. “Verifying and monitoring iots network behavior using mud profiles”. In: *IEEE Transactions on Dependable and Secure Computing* 19.1 (2020), pp. 1–18.
- [98] Dongqi Han et al. “Deepaid: Interpreting and improving deep learning-based anomaly detection in security applications”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 3197–3217.
- [99] Douglas M Hawkins. *Identification of outliers*. Vol. 11. Springer, 1980.
- [100] LLC Pi-hole. *Pi-hole A black hole for Internet advertisements*. <https://github.com/pi-hole/pi-hole>. 2015. [Online; accessed 25.08.2021].
- [101] Jordan Holland et al. “nprint: A standard data representation for network traffic analysis”. In: *arXiv preprint arXiv:2008.02695* (2020).
- [102] Kevin Hsieh et al. “{NetVigil}: Robust and {Low-Cost} Anomaly Detection for {East-West} Data Center Security”. In: *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 2024, pp. 1771–1789.
- [103] Tianrui Hu, Daniel J Dubois, and David Choffnes. “Behaviot: Measuring smart home iot behavior using network-inferred behavior models”. In: *Proceedings of the 2023 ACM on Internet Measurement Conference*. 2023, pp. 421–436.
- [104] Danny Yuxing Huang et al. “Tot inspector: Crowdsourcing labeled network traffic from smart home devices at scale”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4.2 (2020), pp. 1–21.
- [105] Mohamed Idhammad, Karim Afdel, and Mustapha Belouch. “Semi-supervised machine learning approach for DDoS detection”. In: *Applied Intelligence* 48 (2018), pp. 3193–3208.
- [106] IETF. *Manufacturer Usage Description Specification*. <https://tools.ietf.org/id/draft-ietf-opsawg-mud-22.html>. [Online; accessed 01.09.2020].
- [107] Matthew Iji and Rohit Gurung. *IoT Connections Forecast to 2030*. 2023. URL: <https://data.gsmainelligence.com/research/research-2023/iot-connections-forecast-to-2030>.
- [108] Christopher M Inacio and Brian Trammell. “{YAF}: Yet another flowmeter”. In: *24th large installation system administration conference (LISA 10)*. 2010.
- [109] Internet Systems Consortium. *BIND: A Versatile, Classic, Complete Name Server Software*. <https://www.isc.org/bind/>. Accessed: 2024-08-06. 2024. URL: <https://www.isc.org/bind/>.
- [110] Masaru Irino. *softflowd: NetFlow probe for IPv4/v6, software implementation*. Accessed: 2024-08-23. 2024. URL: <https://github.com/irino/softflowd>.
- [111] Arthur S Jacobs et al. “Ai/ml for network security: The emperor has no clothes”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2022, pp. 1537–1551.

- [112] Changhun Jung et al. "A scalable and dynamic acl system for in-network defense". In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022, pp. 1679–1693.
- [113] Hyunjae Kang et al. "IoT network intrusion dataset". In: IEEE Dataport 10 (2019), q70p–q449.
- [114] Eamonn Keogh et al. "Compression-based data mining of sequential data". In: Data Mining and Knowledge Discovery 14.1 (2007), pp. 99–129.
- [115] Paul K Kerr, John Rollins, and Catherine A Theohary. The stuxnet computer worm. Congressional Research Service Washington, DC, 2010.
- [116] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. "A survey of feature selection and feature extraction techniques in machine learning". In: 2014 science and information conference. IEEE. 2014, pp. 372–378.
- [117] Dilawer Khan. IoT Device Fingerprinting. Accessed: 2024-10-30. 2024. URL: <https://github.com/dilawer11/iot-device-fingerprinting/blob/main/src/iotpackage/ModelTraining.py#L933>.
- [118] Minhaj Ahmad Khan and Khaled Salah. "IoT security: Review, blockchain solutions, and open challenges". In: Future generation computer systems 82 (2018), pp. 395–411.
- [119] Wael Khreich et al. "Adaptive ROC-based ensembles of HMMs applied to anomaly detection". In: Pattern Recognition 45.1 (2012), pp. 208–230.
- [120] Hyunchul Kim et al. "Internet traffic classification demystified: myths, caveats, and the best practices". In: Proceedings of the 2008 ACM CoNEXT conference. 2008, pp. 1–12.
- [121] Min-Gyu Kim and Hwankuk Kim. "Anomaly Detection in Imbalanced Encrypted Traffic with Few Packet Metadata-Based Feature Extraction." In: CMES-Computer Modeling in Engineering & Sciences 141.1 (2024).
- [122] Tadayoshi Kohno, Andre Broido, and K.C. Claffy. "Remote Physical Device Fingerprinting". In: IEEE Transactions on Dependable and Secure Computing 2.2 (2005), pp. 93–108. DOI: [10.1109/TDSC.2005.26](https://doi.org/10.1109/TDSC.2005.26). URL: <https://doi.org/10.1109/TDSC.2005.26>.
- [123] Roman Kolcun et al. "Revisiting iot device identification". In: arXiv preprint arXiv:2107.07818 (2021).
- [124] Constantinos Kolias et al. "DDoS in the IoT: Mirai and other botnets". In: Computer 50.7 (2017), pp. 80–84.
- [125] Brian Krebs. "Dyn DDoS Attack Reflects Challenges in Securing IoT Devices.". 2016. URL: <https://krebsonsecurity.com/2016/10/dyn-ddos-attack-reflects-challenges-in-securing-iot-devices/>.
- [126] Deepak Kumar et al. "All Things Considered: An Analysis of {IoT} Devices on Home Networks". In: 28th USENIX security symposium (USENIX Security 19). 2019, pp. 1169–1185.
- [127] Aditya Kusupati et al. "Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network". In: Advances in Neural Information Processing Systems. 2018, pp. 9017–9028.
- [128] Markus Laner et al. "Traffic models for machine type communications". In: ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems. VDE. 2013, pp. 1–5.
- [129] Yukyung Lee, Jina Kim, and Pilsung Kang. "Lanobert: System log anomaly detection based on bert masked language model". In: Applied Soft Computing 146 (2023), p. 110689.
- [130] Ming Li et al. "The similarity metric". In: IEEE transactions on Information Theory 50.12 (2004), pp. 3250–3264.
- [131] Ruoyu Li et al. "ADRIoT: An Edge-assisted Anomaly Detection Framework against IoT-based Network Attacks". In: IEEE Internet of Things Journal (2021).
- [132] Zhong Li, Yuxuan Zhu, and Matthijs Van Leeuwen. "A survey on explainable anomaly detection". In: ACM Transactions on Knowledge Discovery from Data 18.1 (2023), pp. 1–54.
- [133] Xiaoyu Liang and Taieb Znati. "A long short-term memory enabled framework for DDoS detection". In: 2019 IEEE global communications conference (GLOBECOM). IEEE. 2019, pp. 1–6.
- [134] Jessica Lin et al. "A symbolic representation of time series, with implications for streaming algorithms". In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. 2003, pp. 2–11.
- [135] Jessica Lin et al. "Experiencing SAX: a novel symbolic representation of time series". In: Data Mining and knowledge discovery 15.2 (2007), pp. 107–144.

- [136] Shuyu Lin et al. “Anomaly detection for time series using vae-lstm hybrid model”. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Ieee. 2020, pp. 4322–4326.
- [137] Xiangyu Liu, Yi Han, and Yanhui Du. “IoT Device Identification Using Directional Packet Length Sequences and 1D-CNN”. In: Sensors 22.21 (2022), p. 8337.
- [138] Xiaofei Liu et al. “A Comprehensive Survey on IoT Security Issues and Solutions”. In: Journal of Communications and Networks 19.6 (2017), pp. 600–616. DOI: [10.1109/JCN.2017.000087](https://doi.org/10.1109/JCN.2017.000087). URL: <https://doi.org/10.1109/JCN.2017.000087>.
- [139] Zaoxing Liu et al. “Jaqen: A {High-Performance}{Switch-Native} approach for detecting and mitigating volumetric {DDoS} attacks with programmable switches”. In: 30th USENIX Security Symposium (USENIX Security 21). 2021, pp. 3829–3846.
- [140] Manuel Lopez-Martin et al. “Network traffic classifier with convolutional and recurrent neural networks for Internet of Things”. In: IEEE access 5 (2017), pp. 18042–18050.
- [141] Jie Lu et al. “Learning under Concept Drift: A Review”. In: IEEE Trans on Knowledge and Data Eng 31.12 (2019), pp. 2346–2363. DOI: [10.1109/TKDE.2018.2876857](https://doi.org/10.1109/TKDE.2018.2876857).
- [142] Wei Lu and Ali A Ghorbani. “Network anomaly detection based on wavelet analysis”. In: EURASIP Journal on Advances in Signal processing 2009 (2008), pp. 1–16.
- [143] Scott M Lundberg and Su-In Lee. “A unified approach to interpreting model predictions”. In: Advances in neural information processing systems 30 (2017).
- [144] Eman Maali. practical-paper. GitHub repository. 2024. URL: <https://github.com/emanmaali/practical-paper>.
- [145] David JC MacKay. Information theory, inference and learning algorithms. Cambridge university press, 2003.
- [146] Salma Abdel Magid, Francesco Petrini, and Behnam Dezfouli. “Image classification on IoT edge devices: profiling and modeling”. In: Cluster Computing (2019), pp. 1–19.
- [147] Jamshid Mahdavi and Jamshid Mahdavi. sFlow: A Method for Monitoring Traffic in Switched and Routed Net. RFC 3176. 2001. DOI: [10.17487/RFC3176](https://doi.org/10.17487/RFC3176). URL: <https://rfc-editor.org/rfc/rfc3176.txt>.
- [148] Matthew V Mahoney and Philip K Chan. “Learning rules for anomaly detection of hostile network traffic”. In: Third IEEE International Conference on Data Mining. IEEE. 2003, pp. 601–604.
- [149] Manisha Malik, Maitreyee Dutta, et al. “Feature engineering and machine learning framework for DDoS attack detection in the standardized internet of things”. In: IEEE Internet of Things Journal (2023).
- [150] Anna Maria Mandalari et al. “Blocking without Breaking: Identification and Mitigation of Non-Essential IoT Traffic”. In: Privacy Enhancing Technologies Symposium (PETS) (2021).
- [151] Anna Maria Mandalari et al. “Towards automatic identification and blocking of non-critical iot traffic destinations”. In: Workshop on Technology and Consumer Protection (ConPro) (2020).
- [152] Samuel Marchal et al. “Audi: Toward autonomous iot device-type identification using periodic communication”. In: IEEE Journal on Selected Areas in Communications 37.6 (2019), pp. 1402–1412.
- [153] Frank J Massey Jr. “The Kolmogorov-Smirnov test for goodness of fit”. In: Journal of the American statistical Association 46.253 (1951), pp. 68–78.
- [154] M Hammad Mazhar and Zubair Shafiq. “Characterizing smart home iot traffic in the wild”. In: 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI). IEEE. 2020, pp. 203–215.
- [155] Christopher D McDermott, Farzan Majdani, and Andrei V Petrovski. “Botnet detection in the internet of things using deep learning approaches”. In: 2018 international joint conference on neural networks (IJCNN). IEEE. 2018, pp. 1–8.
- [156] Yair Meidan et al. “A novel approach for detecting vulnerable IoT devices connected behind a home NAT”. In: Computers & Security 97 (2020), p. 101968.
- [157] Yair Meidan et al. “Detection of unauthorized IoT devices using machine learning techniques”. In: arXiv preprint arXiv:1709.04647 (2017).
- [158] Yair Meidan et al. “N-baiot—network-based detection of iot botnet attacks using deep autoencoders”. In: IEEE Pervasive Computing 17.3 (2018), pp. 12–22.

- [159] Markus Miettinen et al. "Iot sentinel: Automated device-type identification for security enforcement in iot". In: 2017 IEEE 37th international conference on distributed computing systems (ICDCS). IEEE. 2017, pp. 2177–2184.
- [160] David L. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. Request for Comments: 1305. Accessed: 2024-10-26. 1992. URL: <https://www.rfc-editor.org/rfc/rfc1305>.
- [161] Céline Minh et al. "An explainable-by-design ensemble learning system to detect unknown network attacks". In: 2023 19th International Conference on Network and Service Management (CNSM). IEEE. 2023, pp. 1–9.
- [162] Seyed Mohammad Mehdi Mirnajafizadeh et al. "Enhancing Network Attack Detection with Distributed and {In-Network} Data Collection System". In: 33rd USENIX Security Symposium (USENIX Security 24). 2024, pp. 5161–5178.
- [163] Yisroel Mirsky et al. "Kitsune: an ensemble of autoencoders for online network intrusion detection". In: The Network and Distributed System Security Symposium (NDSS) (2018).
- [164] Sankalp Mittal et al. "iGuard: Efficient Isolation Forest Design for Malicious Traffic Detection in Programmable Switches". In: International Conference on emerging Networking EXperiments and Technologies. 2024, pp. 55–64.
- [165] P. Mockapetris. Domain Names: Implementation and Specification. RFC 1035, November 1987. 1987. URL: <https://tools.ietf.org/html/rfc1035>.
- [166] Yasser Mohammad and Toyoaki Nishida. "Robust learning from demonstrations using multidimensional SAX". In: 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014). IEEE. 2014, pp. 64–71.
- [167] Reza Mohammadi, Reza Javidan, and Mauro Conti. "Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks". In: IEEE Transactions on Network and Service Management 14.2 (2017), pp. 487–497.
- [168] Anand Mudgerikar, Puneet Sharma, and Elisa Bertino. "Edge-based intrusion detection for IoT devices". In: ACM Transactions on Management Information Systems (TMIS) 11.4 (2020), pp. 1–21.
- [169] Nathan J Muller. Bluetooth demystified. McGraw-Hill Professional, 2000.
- [170] Andreas C Müller and Sarah Guido. Introduction to machine learning with Python: a guide for data scientists. " O'Reilly Media, Inc.", 2016.
- [171] Steven J Murdoch. "Hot or not: Revealing hidden services by their clock skew". In: Proceedings of the 13th ACM conference on Computer and communications security. 2006, pp. 27–36.
- [172] Mohammad Najafimehr, Sajjad Zarifzadeh, and Seyedakbar Mostafavi. "A hybrid machine learning approach for detecting unprecedented DDoS attacks". In: The Journal of Supercomputing 78.6 (2022), pp. 8106–8136.
- [173] Netscout. IoT Devices Are Typically Attacked Within Five Minutes of Connecting to the Internet. Accessed: 16-Dec-2024. 2024. URL: <https://www.netscout.com/threatreport>.
- [174] Thien Duc Nguyen et al. "DfIoT: A federated self-learning anomaly detection system for IoT". In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE. 2019, pp. 756–767.
- [175] Thuy TT Nguyen and Grenville Armitage. "A survey of techniques for internet traffic classification using machine learning". In: IEEE communications surveys & tutorials 10.4 (2008), pp. 56–76.
- [176] Nga Nguyen Thi, Van Loi Cao, and Nhien-An Le-Khac. "One-class collective anomaly detection based on LSTM-RNNs". In: Transactions on Large-Scale Data-and Knowledge-Centered Systems. (2017), pp. 73–85.
- [177] Keith Noto, Carly Brodley, and Donna Slonim. "Anomaly detection using an ensemble of feature models". In: 2010 IEEE International Conference on Data Mining. IEEE. 2010, pp. 953–958.
- [178] Number of Connected IoT Devices. <https://iot-analytics.com/number-connected-iot-devices>. Accessed: 2024-05-01. 2024.
- [179] TJ OConnor et al. "HomeSnitch: Behavior transparency and control for smart home IoT devices". In: Proceedings of the 12th conference on security and privacy in wireless and mobile networks. 2019, pp. 128–138.
- [180] Norihiro Okui et al. "Identification of an iot device model in the home domain using IPFIX records". In: 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE. 2022, pp. 583–592.

- [181] Daniel Olszewski et al. ““ Get in Researchers; We’re Measuring Reproducibility”: A Reproducibility Study of Machine Learning Papers in Tier 1 Security Conferences”. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. 2023, pp. 3433–3459.
- [182] Anders Orrevad. M2M traffic characteristics: When machines participate in communication. 2009.
- [183] Jorge Ortiz, Catherine Crawford, and Franck Le. “DeviceMien: network device behavior modeling for identifying unknown IoT devices”. In: Proceedings of the International Conference on Internet of Things Design and Implementation. 2019, pp. 106–117.
- [184] Jitendra Padhye et al. “Modeling TCP throughput: A simple model and its empirical validation”. In: Proceedings of the ACM SIGCOMM’98. 1998, pp. 303–314.
- [185] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. “Challenges in deploying machine learning: a survey of case studies”. In: ACM Computing Surveys 55.6 (2022), pp. 1–29.
- [186] Erquiag Parmisano Garcia. A labeled dataset with malicious and benign IoT network traffic. <https://www.stratosphereips.org/datasets-iot23>. [Online; accessed 12.04.2021].
- [187] Vern Paxson. “Bro: a system for detecting network intruders in real-time”. In: Computer networks 31.23-24 (1999), pp. 2435–2463.
- [188] Roberto Perdisci, Guofei Gu, and Wenke Lee. “Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems”. In: Sixth International Conference on Data Mining (ICDM’06). IEEE. 2006, pp. 488–498.
- [189] Roberto Perdisci et al. “Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis”. In: 2020 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE. 2020, pp. 474–489.
- [190] Roberto Perdisci et al. “McPAD: A multiple classifier system for accurate payload-based anomaly detection”. In: Computer networks 53.6 (2009), pp. 864–881.
- [191] Steven M Pincus. “Approximate entropy as a measure of system complexity.” In: Proceedings of the national academy of sciences 88.6 (1991), pp. 2297–2301.
- [192] Antônio J Pinheiro et al. “Identifying IoT devices and events based on packet length from encrypted traffic”. In: Computer Communications 144 (2019), pp. 8–17.
- [193] Piyush. Internet of Things (IoT). Accessed: 2024-12-15. 2023. URL: <https://www.studyiq.com/articles/internet-of-things-iot/>.
- [194] Sasanka Potluri, Christian Diedrich, and Girish Kumar Reddy Sangala. “Identifying false data injection attacks in industrial control systems using artificial neural networks”. In: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE. 2017, pp. 1–8.
- [195] Sivaramakrishnan Ramanathan, Jelena Mirkovic, and Minlan Yu. “Blag: Improving the accuracy of black-lists”. In: The Network and Distributed System Security Symposium (NDSS). 2020.
- [196] Jingjing Ren et al. “Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach”. In: Proc. of the Internet Measurement Conference (IMC). 2019.
- [197] Counterpoint Research. Global IoT Connections Forecast 2023. 2023. URL: <https://www.counterpointresearch.com/global-iot-connections-forecast-2023/>.
- [198] Francesco Restuccia, Salvatore D’oro, and Tommaso Melodia. “Securing the internet of things in the age of machine learning and software-defined networking”. In: IEEE Internet of Things Journal 5.6 (2018), pp. 4829–4842.
- [199] E. Lear, R. Droms, and D. Romascanu. Manufacturer Usage Description Specification. RFC 8520 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, Mar. 2019. DOI: [10.17487/RFC8520](https://doi.org/10.17487/RFC8520). URL: <https://www.rfc-editor.org/rfc/rfc8520.txt>.
- [200] Joshua S Richman and J Randall Moorman. “Physiological time-series analysis using approximate entropy and sample entropy”. In: American journal of physiology-heart and circulatory physiology 278.6 (2000), H2039–H2049.
- [201] Tanaphon Roempluk and Olarik Surinta. “A machine learning approach for detecting distributed denial of service attacks”. In: (ECTI DAMT-NCON). IEEE. 2019, pp. 146–149.

- [202] Martin Roesch et al. "Snort: Lightweight intrusion detection for networks." In: Lisa. Vol. 99. 1. 1999, pp. 229–238.
- [203] Said Jawad Saidi et al. "A haystack full of needles: Scalable detection of iot devices in the wild". In: Proceedings of the ACM Internet Measurement Conference. 2020, pp. 87–100.
- [204] Makhduma Saiyed and Irfan Al Anbagi. "Entropy and divergence-based DDoS attack detection system in IoT networks". In: 2023 19th WiMob. IEEE. 2023, pp. 224–230.
- [205] Makhduma F Saiyed and Irfan Al-Anbagi. "Deep Ensemble Learning with Pruning for DDoS Attack Detection in IoT Networks". In: IEEE Transactions on Machine Learning in Communications and Networking (2024).
- [206] Ola Salman et al. "A machine learning based framework for IoT device identification and abnormal traffic detection". In: Transactions on Emerging Telecommunications Technologies 33.3 (2022), e3743.
- [207] David Salomon. Data compression: the complete reference. Springer Science & Business Media, 2004.
- [208] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. "Anomaly detection in time series: a comprehensive evaluation". In: Proceedings of the VLDB Endowment 15.9 (2022), pp. 1779–1797.
- [209] Peter Schneider and Konstantin Böttinger. "High-performance unsupervised anomaly detection for cyber-physical system networks". In: Proceedings of the 2018 workshop on cyber-physical systems security and privacy. 2018, pp. 1–12.
- [210] Franka Schuster and Hartmut König. "No Need for Details: Effective Anomaly Detection for Process Control Traffic in Absence of Protocol and Attack Knowledge". In: Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses. 2024, pp. 278–297.
- [211] Bruno Henrique Schwengber et al. "Learning from network data changes for unsupervised botnet detection". In: IEEE Transactions on Network and Service Management 19.1 (2021), pp. 601–613.
- [212] Jonas Herskind Sejr and Anna Schneider-Kamp. "Explainable outlier detection: What, for Whom and Why?" In: Machine Learning with Applications 6 (2021), p. 100172.
- [213] Ramasubramanian Sekar et al. "Specification-based anomaly detection: a new approach for detecting network intrusions". In: Proceedings of the 9th ACM conference on Computer and communications security. 2002, pp. 265–274.
- [214] Vyas Sekar et al. "cSamp: A System for Network-Wide Flow Monitoring". In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI). San Francisco, CA, 2008, pp. 233–246. URL: <https://www.cs.cmu.edu/~dga/papers/csamp-nsdi2008.pdf>.
- [215] sFlow.org. sFlow Version 5 Specification. https://sflow.org/sflow_version_5.txt. Accessed: April 23, 2024. 2012.
- [216] Claude Elwood Shannon. "A mathematical theory of communication". In: The Bell system technical journal 27.3 (1948), pp. 379–423.
- [217] Chin-Shiuh Shieh et al. "Detection of unknown ddos attacks with deep learning and gaussian mixture model". In: Applied Sciences 11.11 (2021), p. 5213.
- [218] Federico Simmross-Wattenberg et al. "Anomaly detection in network traffic based on statistical inference and α -stable modeling". In: IEEE Transactions on Dependable and Secure Computing 8.4 (2011), pp. 494–509.
- [219] Arunan Sivanathan et al. "Characterizing and classifying IoT traffic in smart cities and campuses". In: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE. 2017, pp. 559–564.
- [220] Arunan Sivanathan et al. "Classifying IoT devices in smart environments using network traffic characteristics". In: IEEE Transactions on Mobile Computing 18.8 (2018), pp. 1745–1759.
- [221] Robin Sommer and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection". In: 2010 IEEE symposium on security and privacy. IEEE. 2010, pp. 305–316.
- [222] Xiuyao Song et al. "Conditional anomaly detection". In: IEEE Transactions on knowledge and Data Engineering 19.5 (2007), pp. 631–645.

- [223] Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. “Open for hire: Attack trends and misconfiguration pitfalls of iot devices”. In: Proceedings of the 21st ACM Internet Measurement Conference. 2021, pp. 195–215.
- [224] S Sriram et al. “Network flow based IoT botnet attack detection using deep learning”. In: IEEE INFOCOM 2020-IEEE conference on computer communications workshops (INFOCOM WKSHPS). IEEE. 2020, pp. 189–194.
- [225] State of California. California Consumer Privacy Act of 2018 (CCPA). Accessed: 2024-10-30. 2018. URL: https://leginfo.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375.
- [226] StudyIQ. Internet of Things (IoT), Characteristics, Types and Applications. Accessed: 2024-12-15. 2023. URL: <https://www.studyiq.com/articles/internet-of-things-iot-characteristics-types-and-applications/>.
- [227] Vincent F Taylor et al. “Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic”. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE. 2016, pp. 439–454.
- [228] Vijayanand Thangavelu et al. “DEFT: A distributed IoT fingerprinting technique”. In: IEEE Internet of Things Journal 6.1 (2018), pp. 940–952.
- [229] Jay Thom et al. “Smart recon: Network traffic fingerprinting for IoT device identification”. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). IEEE. 2022, pp. 0072–0079.
- [230] Threatpost. “IoT Devices at Risk: Malware Incidents Surge from 813 Million to 2.9 Billion in a Year”. In: Threatpost (2020). Accessed: 2024-08-16. URL: <https://threatpost.com/iot-disasters-2019/151730/>.
- [231] Threatpost. “Top 10 IoT Disasters of 2019”. In: Threatpost (2019). Accessed: 2024-08-16. URL: <https://threatpost.com/iot-disasters-2019/151730/>.
- [232] Christina Ting et al. “Compression analytics for classification and anomaly detection within network communication”. In: IEEE Transactions on Information Forensics and Security 14.5 (2018), pp. 1366–1376.
- [233] Hanghang Tong et al. “Anomaly internet network traffic detection by kernel principle component classifier”. In: Advances in Neural Networks–ISNN 2005: Second International Symposium on Neural Networks. Springer. 2005, pp. 476–481.
- [234] Rahmadi Trimananda et al. “Packet-level signatures for smart home devices”. In: Network and Distributed Systems Security (NDSS) Symposium. Vol. 2020. 2020.
- [235] Rahmadi Trimananda et al. “PingPong: Packet-level signatures for smart home device events”. In: arXiv preprint arXiv:1907.11797 (2019).
- [236] tshark: Network Protocol Analyzer. <https://www.wireshark.org/docs/man-pages/tshark.html>. 2025.
- [237] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: Journal of machine learning research 9.11 (2008).
- [238] Thijs Van Ede et al. “Deepcase: Semi-supervised contextual analysis of security events”. In: 2022 IEEE Symposium on Security and Privacy (SP). IEEE. 2022, pp. 522–539.
- [239] Victor Kd. HorusEye: control_plane.py, line 727. Accessed: 2024-11-11. 2024. URL: https://github.com/vicTorKd/HorusEye/blob/main/control_plane.py#L727.
- [240] VictorKd. HorusEye. Accessed: 2024-11-11. 2024. URL: <https://github.com/vicTorKd/HorusEye>.
- [241] Cynthia Wagner et al. “Machine learning approach for ip-flow record anomaly detection”. In: NETWORKING 2011: 10th International IFIP TC 6 Networking Conference. Springer. 2011, pp. 28–39.
- [242] Juan Wang et al. “IoT-praetor: Undesired behaviors detection for IoT devices”. In: IEEE Internet of Things Journal 8.2 (2020), pp. 927–940.
- [243] Peter Wayner. Compression algorithms for real programmers. Morgan Kaufmann, 2000.
- [244] Wiki. Dynamic time warping. https://en.wikipedia.org/wiki/Dynamic_time_warping. [Online; accessed 27-Dec-2021].
- [245] Wiki. Mean absolute percentage error. https://en.wikipedia.org/wiki/Mean_absolute_percentage_error. [Online; accessed 27-Dec-2021].
- [246] Wiki. Pearson correlation coefficient. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient. [Online; accessed 27-Dec-2021].

- [247] Wikipedia contributors. Euclidean distance. https://en.wikipedia.org/wiki/Euclidean_distance. [Online; accessed 27-Dec-2021].
- [248] Wikipedia contributors. Euclidean distance — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Euclidean_distance. [Online; accessed 16-September-2024]. 2023.
- [249] Liang Xiao et al. “IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?” In: *IEEE Signal Processing Magazine* 35.5 (2018), pp. 41–49.
- [250] Xiao Xie et al. “Network traffic classification based on error-correcting output codes and nn ensemble”. In: *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 3. IEEE. 2009, pp. 475–479.
- [251] Ruoyu Yan and Chao Shao. “Hierarchical method for anomaly detection and attack identification in high-speed network”. In: *Information Technology Journal* 11.9 (2012), p. 1243.
- [252] Fan Yang et al. “{PROGRAPHER}: An Anomaly Detection System based on Provenance Graph Embedding”. In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 4355–4372.
- [253] Kai Yang, Qiang Li, and Limin Sun. “Towards automatic fingerprinting of IoT devices in the cyberspace”. In: *Computer networks* 148 (2019), pp. 318–327.
- [254] Kun Yang et al. “Ddos attacks detection with autoencoder”. In: *network operations and management symposium*. IEEE. 2020, pp. 1–9.
- [255] Yuchen Yang et al. “A survey on security and privacy issues in Internet-of-Things”. In: *IEEE Internet of things Journal* 4.5 (2017), pp. 1250–1258.
- [256] Muneer Bani Yassein, Wail Mardini, and Ashwaq Khalil. “Smart homes automation using Z-wave protocol”. In: *2016 International Conference on Engineering & MIS (ICEMIS)*. IEEE. 2016, pp. 1–6.
- [257] Ting-Fang Yen et al. “Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks”. In: *Proceedings of the 29th annual computer security applications conference*. 2013, pp. 199–208.
- [258] Jason Yosinski et al. “Understanding neural networks through deep visualization”. In: *arXiv preprint arXiv:1506.06579* (2015).
- [259] Lingjing Yu et al. “You are what you broadcast: Identification of mobile and {IoT} devices from (public){WiFi}”. In: *29th USENIX security symposium (USENIX security 20)*. 2020, pp. 55–72.
- [260] Ming Yu. “A nonparametric adaptive CUSUM method and its application in network anomaly detection”. In: *Int. J. Adv. Comput. Technol* 4.1 (2012), pp. 280–288.
- [261] Xiaodong Zang et al. “Encrypted malicious traffic detection based on natural language processing and deep learning”. In: *Computer Networks* (2024), p. 110598.
- [262] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *Computer Vision–ECCV 2014*. Springer. 2014, pp. 818–833.
- [263] Menghao Zhang et al. “Poseidon: Mitigating volumetric ddos attacks with programmable switches”. In: *the 27th Network and Distributed System Security Symposium (NDSS 2020)*. 2020.
- [264] Shigeng Zhang et al. “Accurate IoT Device Identification based on A Few Network Traffic”. In: *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*. IEEE. 2023, pp. 01–10.
- [265] Wei Zhang et al. “Homonit: Monitoring smart home apps from encrypted traffic”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 1074–1088.
- [266] Jianjin Zhao et al. “Efficient IoT Device Identification via Network Behavior Analysis Based on Time Series Dictionary”. In: *IEEE Internet of Things Journal* (2023).
- [267] Ying Zhong et al. “HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning”. In: *Computer Networks* 169 (2020), p. 107049.
- [268] Wengang Zhou et al. “Internet traffic classification using feed-forward neural network”. In: *2011 International conference on computational problem-solving (ICCP)*. IEEE. 2011, pp. 641–646.
- [269] Bo Zong et al. “Deep autoencoding gaussian mixture model for unsupervised anomaly detection”. In: *International conference on learning representations*. 2018.