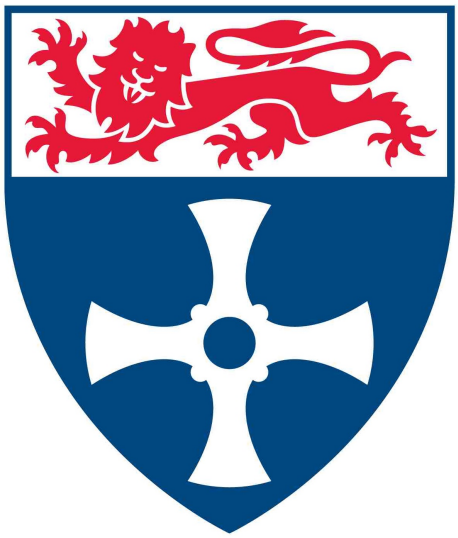
A Methodology for the Quantitative Evaluation of Attacks and Mitigations in IoT Systems



Luca Arnaboldi

School of Computing

Newcastle University

This dissertation is submitted for the degree of

Doctor of Philosophy

Secure and Resilient Systems Group December 2020

I dedicate this thesis to my sibling, for their friendship, wit and incredible strength.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 80,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 100 figures.

Luca Arnaboldi

December 2020

Acknowledgements

I thank my fellow labmates and colleagues for the stimulating discussions, with an extra special mention to Ricardo and his *"pearls"* of wisdom that have helped me get a deeper understanding of academia, Roberto for our many talks and conversations, our work together has been some of the most gratifying work I have done, and Milad for his friendship and support in the last months of this thesis.

I am especially grateful to Dr. Morisset for his continuous support and guidance throughout my study. I also thank my industrial partner ARM, for their contribution to my PhD, and Dr. Griesmayer for his supervision.

I would like to thank my family: my parents and siblings, for supporting and encouraging me throughout my years of study. Last but not least, I thank my partner Désirée, who encouraged and inspired me to pursue an academic career, and who was always by my side throughout all the hardships it involved.

Abstract

As we move towards a more distributed and unsupervised internet, namely through the Internet of Things (IoT), the avenues of attack multiply. To compound these issues, whilst attacks are developing, the current security of devices is much lower than for traditional systems.

In this thesis I propose a new methodology for white box behaviour intrusion detection in constrained systems. I leverage the characteristics of these types of systems, namely their: heterogeneity, distributed nature, and constrained capabilities; to devise a pipeline, that given a specification of a IoT scenario can generate an actionable intrusion detection system to protect it.

I identify key IoT scenarios for which more traditional black box approaches would not suffice, and devise means to bypass these limitations. The contributions include; 1) A survey of intrusion detection for IoT; 2) A modelling technique to observe interactions in IoT deployments; 3) A modelling approach that focuses on the observation of specific attacks on possible configurations of IoT devices; Combining these components: a specification of the system as per contribution 1 and a attack specification as per contribution 2, we can deploy a bespoke behaviour based IDS for the specified system. This one of a kind approach allows for the quick and efficient generation of attack detection from the onset, positioning this approach as particularly suitable to dynamic and constrained IoT environments.

Table of contents

List of figures

List of tables

Glossary

Terms	Definitions
Internet of Things (IoT)	The term *IoT* is defined by International Telecommunication Union (ITU) as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving inter operable information and communication technologies". This definition covers just about every modern internet deployment, we differentiate to cover a subset of these systems in line with most current literature [72, 71, 182], focusing on constrained IoT deployments. In these systems there is an assumption of heterogeneity, constrained capabilities of devices and deficits in connectivity
Intrusion Detection Systems (IDS)	Monitors network and/or system traffic for suspicious activity. Once potential threats are identified, intrusion detection software sends an alert. An IDS comes in one of two forms; host-based intrusion detection system (HIDS) or a network-based intrusion detection system (NIDS). When these approaches are combined, they are collectively referred to as collaborative IDS (CIDS). The remainder of this thesis focusses on NIDS, hitherto referred to as IDS.
Misuse IDS	Are based on a set of predefined rules, dicating events that should not occur in a system.
Behaviour IDS	Examines the behaviour of the specific system and based on this either detect anomalies or uses signatures of specific *bad* and *good* behaviour.

Hybrid IDS	Combines specific techniques, often a mix of misuse and behaviour, to increase the overall detection. Benefits from the advantages of both types of detection.
Black-Box IDS	This approach assumes that a classifier trained on large datasets of network data can then be deployed on a system and can classify oncoming traffic as either good or malicious.
White-Box IDS	Enhances black-box techniques with domain knowledge. This requires careful analysis and understanding of the system.
Actionable ML	In the context of machine learning and more specifically intrusion detection, actionability refers to the ability to act upon a prediction. This may directly relate to understanding the reason behind the prediction, in turn facilitating informed decisions on how to act on it.
Behaviour Traces	Corresponds to the series of network messages transmitted by a device. However, in a model these correspond to a finite sequence of actions outputted by the system.

Glossary: This glossary comprises essential definitions used throughout the thesis. Similar definitions may be reintroduced to contextualise the discussion.

Chapter 1

Introduction

Existing literature cites various challenges with IoT security, naming the dynamism, heterogeneity and constraints of these devices as key factors on why it is insecure [145, 182]. It is generally accepted that due to the large amounts of possible combinations of IoT architectures it is difficult for a single solution to be devised [108]. The constraint of low computational power and memory as well as the potential cost required, complicates the implementation of a full stack security solution on the device itself [89, 83, 108]. It is therefore desirable to investigate approaches which could offload the security effort away from the devices, in the form of IDS techniques. However, implementing an IDS within the IoT faces multiple challenges [61]. If each technique is broken down to each possible method of detection, no single approach is sufficient and each have their downsides as discussed in Sec. 1.1. The remainder of this chapter is structured as follows: Sec. 1.1, discusses issues with IDSs in the IoT, including limitations of previous approaches. Sec. 1.2, introduces the research question and objectives of the thesis, Sec. 1.3, highlights key methodological challenges to addressing our aim, Sec. 1.4, explains the methodology used to address challenges, discusses how the thesis addressed the research question/objectives as well as a discussion of contributions, implications and resultant publications.

1.1 Problem Formulation

Intrusion Detection is a well studied area, with early work starting in the early 90s [80, 183, 159]. The field is perennially evolving, and as the scope of the internet changes, and attacks get more varied, the detection mechanisms evolve alongside them. Standard approaches used to train IDSs include using a database of known attacks (*misuse detection*) and testing systems to create a "benchmark" behaviour (*behaviour based detection*) to find attack patterns [120]. However, despite dozens of proposed approaches and publications in the area, leading figures in the field have extensively discussed downfalls in current detection techniques [161, 63, 64, 61], especially (but not only) in the context of the IoT. To fully understand the extent of the problem, we breakdown the hierarchy of IDS techniques in Fig. 1.1 and provide an overview of their effectiveness.

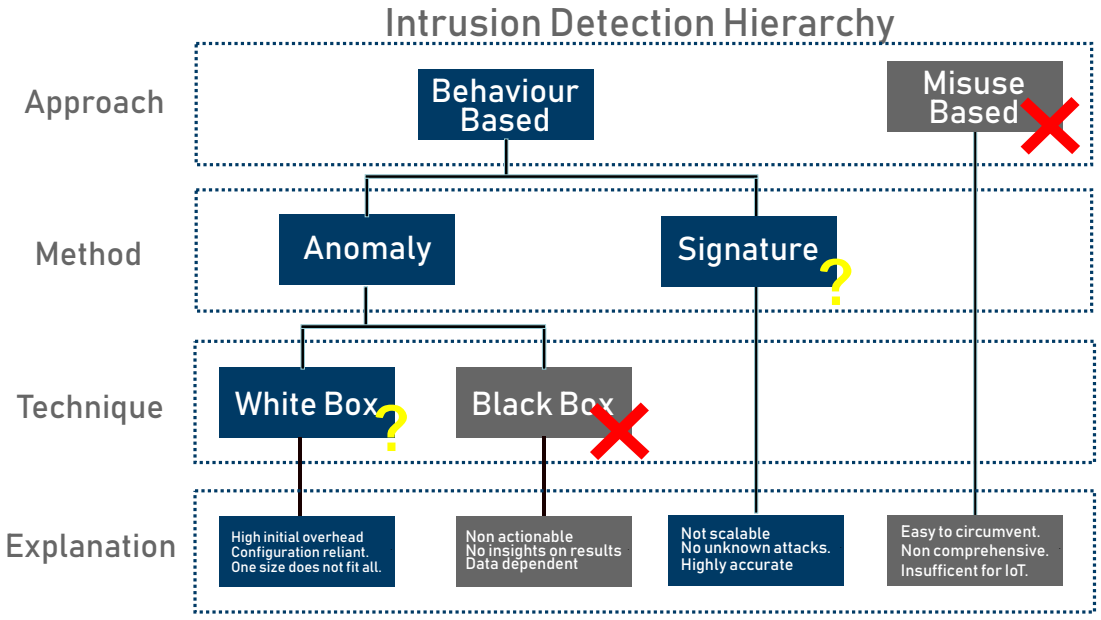


Fig. 1.1 Categorisation and limitations of IDS strategies for IoT, adapted from [64]

Misuse detection is largely regarded as difficult to implement as it can be time consuming to enforce, requires large databases of data, has high overhead (especially for IoT devices) and may still be easily circumvented by attack obfuscations [120, 64]. Due to the nature of their design, having to specify the attacks before hand, they are unable to detect unknown/new

attacks. That said, the majority of modern commercial IDS systems are misuse based. They have the advantage that they are highly actionable, meaning that if an attack is detected, the nature of the attack is readily identified and mitigations are easier to devise. They can also scale to a variety of systems as they are attack-dependent, rather than setup-dependent.

In the case of behaviour based IDS, it can be challenging to establish a benchmark behaviour in dynamic IoT systems as devices may constantly shift, new devices might join and behaviours can change [72]. This complicates the implementation of black box anomaly detection. To compound these issues, whilst some black box systems may well be highly accurate at detecting attacks, they are highly non-actionable. In other words, given a black box approach, such as a deep neural network, if an anomaly is found, it is difficult to reason about; as one cannot know what to do about it, or whether the anomaly is even relevant. Sommer and Paxson [161], provide an extensive discussion on this topic. Conversely, with specific domain knowledge of the system it is possible for behaviour detection to be very powerful, as it can distinguish attacks without reference of an attack database. With a good representation of the system (white box), that explains what is going on between its components, it is much easier to understand *how* things go wrong, improving on the actionability of behaviour based detection.

In the IoT certain scenarios make it unfeasible to adopt a traditional behaviour based black box IDS approach. These specific IoT deployments are dynamic, constrained, and safety critical. This specific mix of circumstances makes it crucial for security measures to be in place, but prevent standard data collection techniques (such as penetration testing), to train and deploy the IDS.

Example Scenario 1 : Earthquake Safety Sensors

Scenario adapted from IETF Draft on Disadvantaged networks [60]. Assume that IoT devices are deployed to monitor a large geographical area following a significant earthquake with threat to human life. These IoT devices may be small constrained sensors collecting information from their environment to aid search and rescue efforts. Each of these IoT devices will collect data to find survivors, alert first-responders of potential building collapses

and monitor the environment for safety conditions and further shocks. In this scenario there is a central server which will receive and process the data, as well as act as an IDS. The central server will be mostly static, or slow moving; it could be deployed in a nearby building, or transported in vehicles if there is no central location. Furthermore, first-responders would most commonly rely on smartphones or tablets, to interact with the devices. Due to the mobility of the first responders and the large areas over which the sensors are deployed, there would only be intermittent connectivity to both the central server and the sensors. In such a scenario it is essential that key information is delivered, as the denial of one device could mean that first-responders are not alerted to a building collapse or are unable to locate a survivor.

Example Scenario 2 : COVID-19 Temporary Field Hospitals

Due to the global coronavirus disease (COVID-19) pandemic, on 3 April 2020, the government of the United Kingdom announced the construction of the first NHS Nightingale hospital [1]. These temporary facilities were designed as state-of-the-art e-hospitals with internet connectivity and means to quickly expand if needed. The temporary field hospital could house 5000 beds, each integrated with sensors and Internet connectivity. They are to be quickly deployed with the intent to be immediately used for emergency treatment, limiting time for cyber security evaluation to almost zero. This one of a kind IoT scenario, where the effects of an attack could be human lives creates a unique security problem.

There is no current silver bullet solution to the complex problem that is intrusion detection in the IoT. Chapter 3 presents a novel and extensive survey of techniques used in the extant literature, and demonstrates that most techniques are inadequate or currently not capable of detecting certain attacks as well as being unsuitable for certain IoT setups. Although there is a lot of promise in the usage of techniques that may take advantage of the domain knowledge of the IoT deployment, the issue of system constrains in the context of IoT still remains, making the ability to use this white box knowledge an open challenge [64].

[1] "Coronavirus: Nightingale Hospital opens at London's ExCel centre". BBC News. 3 April 2020

1.2 Research Question

Intrusion Detection is a challenging field with many open questions, and in the context of the IoT there are several restrictions that narrow down options even further. Specifically, there is an increasing need to develop means to use knowledge based detection in a non static manner, as the most promising and applicable technique; albeit, with the downside that they are difficult to adapt to the constrains of the IoT and there are still limitations in what attacks may be captured. I therefore investigate: *"Can we devise a new model-based enhancement to knowledge based intrusion detection techniques to expand on the current deployment options and improve our attack detection capabilities in constrained IoT systems?"*

1.2.1 Aim

The overall aim is to provide a new, well-tested methodology for the deployment of knowledge based IDSs for constrained IoT systems - through the use of white box formally defined system models.

1.3 Research Challenges

Devising a methodology for implementing a model based behaviour for IoT networks is not without challenges:

- **Challenge 1.** There is no existing gold-standard criteria in which to evaluate and compare IDS solutions.
- **Challenge 2.** Modelling constrained and dynamic IoT scenarios to evaluate their security is non trivial and to the best of our knowledge, an accepted methodology does not currently exist.
- **Challenge 3.** A model is an abstraction of the system, and often does not directly correspond to captured system behaviour.

It has been widely argued that current techniques will not succeed for IoT systems, and so the literature proposes solutions that cater to the scenarios specific to the IoT. However, following our extensive review presented in Chap. 3, it is evident that the evaluation of solutions in a uniform manner, remains an open challenge **[Challenge 1]**.

Due to rapid advances in sensor technologies, it is now possible to deploy them in many types of new scenarios. One such deployment involves the positioning of devices in constrained IoT systems, which is often the case for emergency and/or safety critical situations. These new characteristics make it difficult to conduct accurate quantitative assessments and to model realistic scenarios **[Challenge 2]**.

Making use of modelling approaches to enhance the understanding of attacks on a system prior to physical system implementation poses a unique challenge. There is currently no accepted methodology of how to model these types of systems to capture the desired behaviours and apply them in this manner. Further adjustments are required to relate to real system dynamics and to reason about an attack, extrapolating from a model to a real system **[Challenge 3].**

1.4 Proposed Solution

In scenarios such as these where system specifics might be known, but network datasets are unavailable, methods that incorporate this knowledge need to be devised. To reason about these systems it is beneficial to abstract them into the core characteristics relevant to each specific scenario and to formalise the problem using models. Models offer the opportunity to assess various configurations and to evaluate systems for security with ease. An advantage is that the simplistic nature of IoT devices which characterizes them as particularly hard to secure, also plays to their favour when abstracting their behaviour in formal models. Since the behaviours to capture are simpler, more accurate characterisations can be modelled to observe the security of the systems. This can account for system behaviour in the IDS detection, mitigating the limitations that static knowledge-based approaches may face.

A formal model is relatively easier to scale and adapt compared to a real system - a formal model can observe events and model attacks, and it can simulate different scenarios. We create two new modelling techniques to describe an IoT system; one that can capture attacks on the system, and one to capture the interactions between devices. We specifically focus on protocols used in the system to describe the interactions. Both models leverage an extra layer of battery drain as a key factor in their analysis. Based on these intuitions, we seek out to develop new attack detection techniques based on formal models of IoT systems.

Our proposed model presents the following foreseeable advantages. Using our approach, we examine attacker behaviour through the use of a Markov chain, placing less emphasis on individual packets. Our stochastic Markovian chains may resemble very complex attacker signatures, mimicking multi-step attacks and decreasing the likelihood that these can be circumvented. By using modelling instead of system data for intrusion detection, we can have many more training sets, at a much faster rate.. Importantly, we are also able to capture system behaviour that is not typically available in standard datasets, such as battery usage. This offers the benefit of not only the detection of typical behaviour, but also the extension to the detection of good behaviour. Maximizing on this good behaviour, the anomaly detection can be used to identify harmful behaviour in a way that conventional approaches cannot.

We unite our modelling approaches to generate a Lightweight IoT System Specification under Attack (LISSA). A LISSA is composed of two elements; , 1) Specification of the system using our *"gentlemen devices"* and *"rude devices"* approach [16], ; and 2) a attack specification through use of stochastic processes, as per our attack specification model *Lightweight IoT System under attack (LISA)* [17, 18]. By combining these components, LISSA is able to deploy a bespoke IDS for the specified system. By extrapolating the network traffic to a set of traces of behaviour, they can be compared to the model traces to find anomalies or known signatures of attacks. In summary, the thesis presents the following novel contributions:

- **Contribution 1.** In response to Challenge. 1, we propose a systematic assessment of proposed IDS solutions for the IoT and provide an evaluation of current limitations and current state of the art, presented in Chap 3.

- **Contribution 2.** In answer to Challenge. 2, We propose a modelling approach that focuses on the observation of specific attacks on possible configurations of IoT devices. This enables a user to have accurate representations of the system prior to deployment, to observe system behaviours and run security evaluation.

- **Contribution 3.** In answer to Challenge. 2 and 3, we propose an IDS that can be trained and deployed prior to setting up the system. This is achieved by making use of the model behaviours (as per Contribution 2) as the basis of the detection, instead of necessitating collection of the real system data.

1.4.1 Thesis Structure

Chapter 2 provides relevant background literature, as well as the conceptual reasons underpinning and informing this thesis. In each chapter, the scope is summarised in respect to the aim. Each chapter contains its own related work pertinent to the specific topic, has its own evaluation for each section and a final chapter conclusion is also provided. In Chapter 6 an evaluation and discussion of the approach is presented, as well as potential future extensions.

- **Chapter 2:** Background and Related Work. This chapter highlights the concepts and work that provide the foundations for this research.

- **Chapter 3:** A Review of Intrusion Detection Systems in the IoT - A survey & Qualitative Analysis. A structured survey of the current state of intrusion detection in the IoT. Over 50 IDSs are reviewed, representing the largest survey on this topic at the time of writing, to the best of our knowledge. Beyond the survey, we also provide an assessment methodology and qualitatively analyse the pros and cons of all approaches. We then provide an analysis of the current state of attack detection in the IoT. This chapter addresses Contribution 1.

- **Chapter 4:** A formal modelling technique for the evaluation of IoT System Interactions. In this chapter, we present the novel *gentleman/rude device* approach to modelling interactions between IoT devices and their behaviour under attack as per

Contribution 2. This technique is the foundation for our detection and targets addressing means to model adaptable and flexible IoT scenarios.

- **Chapter 5:** A formal modelling technique for the quantitative assessment of attacks on IoT Systems. This chapter presents LISA, our novel modelling approach for the observation and assessment of attacks on IoT systems. The sections describe how a LISA can be modelled and demonstrates how analysis can be performed to quantify the impact of attacks and evaluate their effectiveness. The attack behaviour in the model is then used for the signature components of the final IDS. This comprises Contribution 2 and 3.

- **Chapter 6:** Conclusion & Reflections. This final contribution discusses our modelling approaches to form the intrusion detection system. The thesis concludes with reflections about the research question and evaluates whether the aim was addressed. We then discuss future expansions for this work.

1.4.2 Publications

Some chapters in this thesis are formed from my publications. Unless indicated, all papers are solely the original contribution of my own work, under the supervisory and editorial support of my supervisor, Charles Morisset. In the case where work part of a collaboration with colleagues, this is explicitly stated in the chapter, and a discussion of my own contributions is presented. Some of the work done in these papers is extended and adapted for the specific scenario of intrusion detection. These are my sole contributions, and is not contained within publications at the time of writing. Publications are subsumed within the following contributions: Contribution 1 is composed of a survey presented in Chap. 3. Arnaboldi & Morisset (2017) and Arnaboldi et al. (2019) address Contribution 2, Arnaboldi & Morisset (2018-1) and Arnaboldi & Morisset (2018-2) comprise Contribution 3, Arnaboldi & Tschofenig (2019) and Arnaboldi & Metere (2019) inform the motivations behind our work, but not used in any of the thesis chapters. A full list of publications and (work in progress) potential future applications of this thesis are presented below:

Publications used for thesis

- Arnaboldi, L., Czekster, R. M., Morisset, C., & Metere, R. (2020). Modelling Load-Changing Attacks in Cyber-Physical Systems. Electronic Notes in Theoretical Computer Science, 353, 39-60.
- Luca Arnaboldi and Charles Morisset, "Generating Synthetic Data for Real World Detection of DoS attacks in the IoT", Federation of International Conferences on Software Technologies: Applications and Foundations Springer, Cham, Toulouse, France, 2018,.
- Luca Arnaboldi and Charles Morisset, "LISA Predicting the Impact of DoS Attacks on Real-World Low Power IoT Systems", Foundations of Computer Security (FCS), Oxford, UK, 2018
- Luca Arnaboldi and Charles Morisset, "Quantative Analysis of Denial Of Service Attacks and Client Puzzles in IoT Systems", Security and Trust Management Workshop (STM), Oslo, Norway, 2017

Further publications in preparation, as an outcome of this research or outside its scope

- Luca Arnaboldi and Charles Morisset, "A White Box Anomaly Detection System For Disadvantaged Networks", *Work in Progress* - Venue To Be Decided.
- Luca Arnaboldi and Charles Morisset, "A Review of Intrusion Detection Systems and Their Evaluation in the IoT: A survey & Qualitative Analysis", *Work in Progress* - Venue To Be Decided.
- Roberto Metere and Luca Arnaboldi, "MetaCP: Cryptographic Protocol Design Tool for Formal Verification", *Work in Progress* - Venue To Be Decided.
- Artur Sokolovsky and Luca Arnaboldi, "Machine Learning Classification of Price Extrema Based on Market Microstructure Features: A Case Study of S&P500 E-mini Futures", *In Submission* - Journal of Algorithmic Finance.

- Luca Arnaboldi and Roberto Metere, “Towards a Data Centric Approach for the Design and Verification of Cryptographic Protocols”, Proceedings of ACM Computer and Communications Security (CCS) - Poster Session, London, UK, 2019
- Luca Arnaboldi and Hannes Tschofenig, “A Formal Model for Delegated Authorization of IoT Devices Using ACE-OAuth”, Fourth Annual OAuth Workshop, Stuttgart, Germany, 2019

Chapter 2

Background

2.1 Chapter Introduction

This chapter presents pertinent background knowledge in relation to each contribution of the thesis. We begin by discussing the different approaches for IDS usage in IoT, discussing current research in knowledge based detection and justifying the need for further extensions in this area. Sec. 2.2 follows on describing current pitfalls in IoT security and explain the motivations of using an IDS as a defence mechanism for such systems in Sec. 2.3, I then introduce the notion of probabilistic modelling as a means to understand IoT systems and act upon malicious behaviours in Sec. 2.4.

2.2 Intrusion Detection Systems for IoT

Implementing an IDS within an IoT network faces several challenges. In dynamic IoT systems, devices may constantly shift, new devices might join and behaviours might change [72]. Simultaneously, protocols can vary from one network to another, which may necessitate data collection to be bespoke to an individual system [73]. Finally, some system changes can require data (or part of the data) to be collected from scratch (e.g. interactive smart homes where devices can change frequently). An in depth review of Intrusion Detection in

the IoT and its limitation is presented in a Survey in Chapter 3. For brevity. here I focus the discussion on behaviour based detection, to justify and contextualise our research aim.

2.2.1 Behaviour Based Detection

Behaviour based techniques seek to establish a baseline of good behaviour of the system, or bad behaviour in case of signature. False positives are common with behaviour based detection, and can often incur higher operational costs than false negatives [64], e.g. a false positive may require a dedicated analysis of several hours of work to identify threats, protect assets and perform incident response procedures, whilst a false negative may just be a scanning software or non consequential access attempt. Often as an outcome of the latter, actionability is a also a big topic of interest in IDS research. Spotting unusual behaviour is one thing, however if there is no other information it becomes very difficult to plan a response, making informative IDSs a requirement. The way we train these system takes two general approaches, Black Box and White Box. The first assuming no domain knowledge and the latter based on known details of the system to aid the behaviour analysis.

Black Box Detection

A black box approach assumes that a classifier trained on large datasets of network data can then be deployed on a system and classify oncoming traffic as either good or malicious. To an outsider to the field, one would assume that an IDS would be the perfect fit for a machine learning approach [161]. However, with further insight into this we can see why it so difficult to apply in practice. In the real world, classifying network data is not an easy task. As networks vary vastly, the training data from even quite a similar system might be of poor quality for another, and in reality this similarity is almost never the case. There currently is not many available datasets, with several academic papers still choosing 20+ year old datasets as the case studies for evaluation [1]. Finally due to the critical nature of these

[1] KDD Cup 1999, was a dataset developed for IDS competitions and evaluation by DARPA, and is still widely used for evaluation, usually to benchmark against previous approaches. Available: http://kdd.ics.uci.edu/databases/kddcup99/

systems, there is a very high cost associated with errors, which is exacerbated by the fact the using this approach it is difficult to understand the cause of the error.

White Box Detection

As the consequence of the limitations identified in previous discussion, research has shifted towards the enhancement of these same techniques with domain knowledge [161]. This approach, defined as White Box detection requires careful analysis and understanding of the system. This may take several different approaches, and there is no unique way to do so [63]. In practice a scope of the system needs to be designed, as different systems will have different types of requirements. A threat analysis might be conducted, this need not encompass clear attack rules such as misuse approaches, but rather an understanding of what threats one wishes to defend against e.g. DoS. It is also suggested to perform risk assessments, to reduce false positives will significantly improve performance [161]. Given a white box approach, and a well thought out system assessment, it becomes much more feasible to act upon a threat, and whilst no detection system is perfect, it is preferable to have one where in anomaly can be acted upon, even if it were to reduce accuracy slightly.

Whilst this approach has many benefits, it is still not simple to achieve. There currently is no one methodology to gain this white box knowledge of a system, and even less guidance on how to apply it to an IDS in practice [64]. Sommer and Paxson [161], provide some recommendations and discuss ways to achieve this, however very little work has implemented it or extended upon their guidelines. The problem consequently shifts from the study of attack detection to the successful description of a system (which may also be shifting). This area is an open research problem and we attempt to propose new methodologies to specify the way systems behave so that we can improve on the current way white box detection is done - through formal modelling and verification.

2.3 IoT System Security

Literature classify these kinds of systems differently, whilst some work focuses on the deployment structures and communication characteristics [71], other works focuses on the characteristics of the devices themselves [182]. An IoT device might spend long periods of time offline and not connected to the internet, is often powered by battery, and its computing power is greatly restricted [72]. These new challenges are something current technologies and various protocols have not been built to to deal with [182], which may cause for there to be security flaws in the current implementations.

As discussed in the survey by Sicari et al. [71], traditional security countermeasures often cannot be directly applied to IoT technologies due to the different standards and communication stacks involved. Consequently, researchers need to develop new security solutions bespoke to these systems. This issue spans across almost all areas of security, not just communications, as the problem is deeply rooted into how these systems are designed [182, 71]. In our work we show that a solution which might be perfectly functional for a traditional infrastructure, may actually cause harm in certain IoT deployments [16]. This establishes that ways to evaluate these systems as a useful tool to have.

As infrastructures shift towards more distributed systems, the requirements change, whilst initial computing design was constrained by costs of transistors [33, 57], constrains have now shifted towards, power and computing speed [182, 33]. The scope and purpose of these systems likewise has evolved, in the IoT there no longer needs to be a single multi purpose workstation performing complex analysis, instead several basic devices are meant to work in unison towards a common objective. By design, and to reduce complexity, these devices are often very simple, performing basic sensing operations and communicating them onward. Whilst this design allows them to perform their purpose, it often comes at the cost of being unable to do other desirable key tasks, most importantly security [145]. Often built for large systems and relying on hard to compute operations, such as cryptographic functions, many current security techniques are not well suited for these new systems.

2.3.1 Evaluatable IoT Systems for Intrusion Detection

Understanding the impact of attacks is core to answering our research question. Through system understanding we can better understand why attacks are raised. As we cannot expect solutions and attacks to apply in the same way to these kinds of systems [16], we need to find new ways to observe this. Whilst non comprehensive, we focus our research on the security implications of the following characteristics: 1) Devices have limited battery life, and need to limit actions to conserve energy; 2) IoT systems often span across large areas, both geographically and across scopes, extenuating the difficulty to monitor traffic and deploy defence mechanisms; 3) Computational restrains limiting the ability to perform security tasks; and 4) Usage of short range communication technologies making routing of messages and deployment of nodes a complex problem; In order to effectively evaluate the impact of these properties we make use of formal models.

2.4 Probabilistic System Modelling

Formal modelling can be used to abstract a system or subsystem to test its integrity or to evaluate disruptive behaviours. When looking at the IoT due to the sheer number of devices modelling a system with any degree of accuracy is difficult [48], we consequently investigate different and new approaches to make informed security decisions balancing the right level of abstraction. Our goal was to design an approach that was flexible enough to capture various different scenarios, but also non restrictive so that it allowed the specification of evaluation properties of interest.

Several modelling techniques exist including: Queueing Networks (QN) [164], Stochastic Petri Nets (SPN) [126], Performance Evaluation Process Algebra (PEPA) [81], Superposed Generalized Stochastic Petri Nets (SGSPN) [59], Stochastic Automata Networks (SAN) [143], and Markov Chains [133], to name a few. We finally converged to the usage of Markov chains for the following reasons:

1. Markov chains are usually considered due to simplicity in terms of modelling primitives - only states and transitions decorated with rates (CTMC) or probabilities (DTMC), fitting our desired properties fully.

2. This decisions was partially supported by the existence of mature and well researched tooling, namely the PRISM Model Checker [101]. PRISM allows to models a wide range of situations allowing for formalisations using probabilistic automata, MDP, on top of CTMC and DTMC.

3. PRISM provides a modular approach and modellers may scale modules as needed using a technique known as reactive modules [7]. This allows the designer to take the necessary precautions to tackle state space explosion problems common in Markovian approaches.

4. In addition the tool allows for a rich property specification language able to capture a wide variety of scenarios [100].

A discussion of the used formalism as well as PRISM is continued on the next section, more detailed explanations of these models can be found in Baier et al. [26].

2.4.1 Markov Chains

A markov chain in its core is a state transition system, augmented with probabilities, possessing the so called *Markov* property. This property defines that: if the current state is known, then the future states of the systems are independent of its past states. i.e. the current state of the models contains all the necessary information required to influence the future state of the system. When referring to Markov chains, with discrete time and countable states we refer to Discrete Times Markov Chains (DTMC). In this context, discrete time views values of variables as occurring at distinct and separate points in time.

Formally a DTMC D is a tuple (S, s_{init}, P, L):

- S is the set of states

- $s_{init} \in S$ is the initial state of the system
- $\mathbf{P} : S \times S \to [0,1]$ is the *transition probability matrix* where $\sum_{s' \in S} \mathbf{P}(s,s') = 1$ for all $s \in S$
- $\mathbf{L} : S \to 2^{\alpha}$ is a function labelling states with atomic propositions taken from a predefined alphabet α

A state in the system represents the combination of variable values, often delimited figuratively by a circle with a label. Consequently, S delimits every possible combination of possible values the variables in the system may take, with S_{init} denoting their initial values. The transition probability matrix denotes the probability for any given state to transition to another available state, where for each state, the sum of probabilities for each possible transition will sum to one, delimited graphically as arrows between states. And finally the labelling function $\mathbf{L}$, is used to observe specific scenarios within the model, it assigns a label to a set of states which fit a specific boolean condition. This may be very useful to e.g. find failure states in the system.

Running Example (DTMC) - Knuth and Yao Six Sided Dice with Coin

Knuth & Yao [94] propose the mathematical problem of modelling the probability distribution of repeatedly flipping a fair (six-sided) die using a fair coin. To solve this problem, we can make use of a concept called rejection sampling, and basic binary mathematics. The coin is flipped three times, each time resulting in either heads (0) or tails (1), concatenating the three bits you get a range of numbers from 0 to 7. Eliminating state 0 (000) and 7 (111) by re-rolling leads to the required options for our playing dice. For the remaining six combinations, we assign 1-6 individually and terminate. Following the notation presented previously our DTMC is formalised in Fig. 2.1 and visualised in Fig. 2.2.

Continuous Time Markov Chains

A Continuous Time Markov Chain (CTMC), extends the DTMC over the state space S with a function $r : S \to \mathbb{R}_{>0}$, assigning to each state s the rate of a negative exponential distribution,

initial state

$$\overbrace{\begin{pmatrix}0,0,0 & 0\end{pmatrix}}$$

states

$$\overbrace{\begin{pmatrix}
s0:(0,0,0) & 0\\
s1:(0,0,1) & 0\\
s2:(0,1,0) & 0\\
s3:(0,1,1) & 0\\
s4:(1,0,0) & 0\\
s5:(1,0,1) & 0\\
s6:(1,1,0) & 0\\
s7:(1,1,1) & 1\\
s8:(1,1,1) & 2\\
s9:(1,1,1) & 3\\
s10:(1,1,1) & 4\\
s11:(1,1,1) & 5\\
s12:(1,1,1) & 6
\end{pmatrix}}$$

transition matrix

$$\overbrace{\begin{pmatrix}
s & s' & P(s,s')\\
s0 & s1 & 0.5\\
s0 & s2 & 0.5\\
s1 & s3 & 0.5\\
s1 & s4 & 0.5\\
s2 & s5 & 0.5\\
s2 & s6 & 0.5\\
s3 & s1 & 0.5\\
s3 & s7 & 0.5\\
s4 & s8 & 0.5\\
s4 & s9 & 0.5\\
s5 & s10 & 0.5\\
s5 & s11 & 0.5\\
s6 & s2 & 0.5\\
s6 & s12 & 0.5\\
s7 & s7 & 1\\
\vdots & \vdots & \vdots\\
s12 & s12 & 1
\end{pmatrix}}$$

Labelling

$$\overbrace{\begin{matrix}
L(s7) = \\
L(s8) = \\
L(s9) = \\
L(s10) = \\
L(s11) = \\
L(s12) =
\end{matrix}}$$

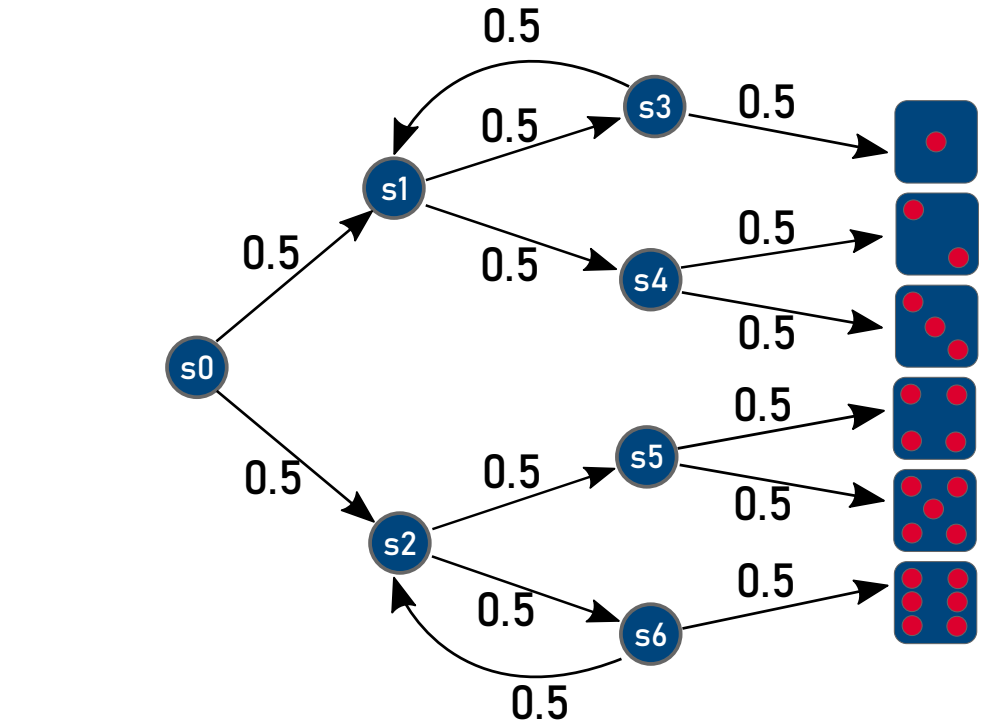


Fig. 2.2 Example DTMC - Graphical Representation

governing the residence time spent in s. So, the likelihood to reside within s for a maximum time unit d is $1 - e^{-r(s)\cdot d}$, so the average stay within any state s is $1/r(s)$. To link the current definitions with the previous notation, we no longer have $\mathbf{P}$, which instead becomes $\mathbf{R}$ by doing $R(s,s') = P(s,s') \cdot r(s)$, to dictate the transition rate from s to s'. To interpret how this works in practice, upon entering the state s, the time of stay is dictated by the exponential distribution over the rate $r(s)$, and upon exiting s, the probability to transition to state s' is dictated by $P(s,s')$. The main difference between these two formalisms, is that within a CTMC, time stayed at a state needs to be factored in. This leads to situations in which there exist multiple s' with $\mathbf{R}(S,s') > 0$. This is called a race condition, at which whichever action is triggered first dictates the next state. Given a time interval $[0,t]$, the probability to move onto state s' and determine which is the likely winner of the race is:

$$\frac{\mathbf{R}(s,s')}{r(s)} \cdot \left(1 - e^{-r(s)\cdot t}\right)$$

This particular paradigm becomes particularly powerful in situations when timings are critical to the operations of the system.

Markov Decision Process

The final extension to a MC which we will discuss is a Markov Decision Process (MDP), which extends the MC with non-determinism. The main difference is that whilst a DTMC will transition to a new state s' according to a probability distribution $D_s = \mathbf{P}(s,s')$, a state in an MDP may have several different distributions. Upon reaching a state s in an MDP, a distribution $\omega \in D(s)$ is selected and the next state is selected with probability $\omega(s')$. One limitation however is that it is not possible to have an unknown distribution, the system must be defined for all of the actions, with a random choice of distributions.

In this formalism paths can be chosen independent of probabilities to select the most optimum scenario. Based on these strategies, we can resolve the non-determinism in a MDP in various different ways resulting in several different DTMCs. A strategy takes multiple finite paths starting from the initial states, and assigns rewards to actions, based on a property

you are trying to optimize for, e.g. time. After computing every possible path it will choose the one with the highest reward and compute the probability distribution for the path, to form the DTMC. This is achieved using the Bellman equation [29].

$$V(x) = \max_{a \in \Gamma(x)} \{F(x,a) + V(T(a,x)) \cdot \beta\}$$

As can be seen this a recursive function acting on state x until the final state is reached. It aims to find the optimal value V, which is the best possible value of the objective as a function of the state x. We denote the available actions a in the current state x by $a \in \Gamma(x)$. To denote the current reward associated to an action we use notation $F(x,a)$. And finally we recursively call the equation on the next transition state denoted by $T(a,x)$, to which discount factor β is applied in the case of diminishing reward. This will recursively progress through the transitions to compute the values of the paths in the system.

2.4.2 PRISM Model Checker

PRISM is a probabilistic model checker [101], a tool for formal modelling and analysis of systems that exhibit random or probabilistic behaviour. It has been used to analyse systems from many different application domains, including communication and multimedia protocols, randomised distributed algorithms, security protocols, biological systems and many others, available at [2]. It is also an open source, multi platform tool which has been in active development for over 20 years, making it well tested and proven as a solid option. PRISM models are written using a simple state based highlevel language, from which the tool automatically constructs either a DTMC, CTMC or MDP. The prism language makes use of a concept of reactive modules, this techniques devises small MCs for each modules and links them together by formulas and compositions. The basic components of the language are, 1) Modules, which are the modular components, composed in parallel, which capture the logic of the model; 2) Variables, composed of *finite* integer ranges either local or global;

[2] http://www.prismmodelchecker.org/

and 3) Guarded Commands, which dictate the behaviour of each module i.e. the changes in state which can occur, labelled by probabilities. These take the form of:

$$\overbrace{[\text{update}]}^{\text{action}}\ \overbrace{x=0}^{\text{guard}} \rightarrow\ \overbrace{P_x:}^{\text{probability}}\ \overbrace{(x'=1)\ \&\ (s'=s+1)}^{\text{update}}\ \overbrace{+}^{\text{and}}\ \overbrace{P_y:}^{\text{probability}}\ \overbrace{(y'=0)}^{\text{update}}$$

Synchronisation is achieved across modules by interlinking specific actions triggering the updates in both conditions and double guarding the transition.

Running Example (PRISM Language) - Knuth and Yao Six Sided Dice with Coin

In the PRISM language the described DTMC is specified as shown in Fig. 2.3. As can be observed the prism language matches closely with the described formalisation. A variable s represents the possible values of the states (binary 000 to 111) with a terminating state when $s = 7$, and the variable d represents the state of the die. If one wishes they can easily follow the guarded equations to construct the full state space in Fig. 2.1.

```
dtmc
module die
        // local state
        s : [0..7] init 0;
        // value of the die
        d : [0..6] init 0;

        [] s=0 -> 0.5 : (s'=1) + 0.5 : (s'=2);
        [] s=1 -> 0.5 : (s'=3) + 0.5 : (s'=4);
        [] s=2 -> 0.5 : (s'=5) + 0.5 : (s'=6);
        [] s=3 -> 0.5 : (s'=1) + 0.5 : (s'=7) & (d'=1);
        [] s=4 -> 0.5 : (s'=7) & (d'=2) + 0.5 : (s'=7) & (d'=3);
        [] s=5 -> 0.5 : (s'=7) & (d'=4) + 0.5 : (s'=7) & (d'=5);
        [] s=6 -> 0.5 : (s'=2) + 0.5 : (s'=7) & (d'=6);
        [] s=7 -> (s'=7);
endmodule
```

Fig. 2.3 PRISM Model for six sided dice with fair coin, single module with guarded state transitions, labelled with probabilities.

2.4.3 Advantages of Modelling Approaches

Probabilistic model checking has been successfully applied to a variety of security scenarios [134, 22, 27]. It allows for the observation of attack impacts, worse case scenarios, as well as performance analysis of several different mitigations. Complex scenarios can be observed and used to make informed decisions about a system. Previous work has been able to observe attack and defence scenarios to find optimal resolutions in stochastic systems [22]. This extends previous work on attack defence trees and formally verifies the trees using game theory. This approach exemplifies the usefulness of such tools to reason about complex scenarios. Our analyses to evaluate IoT systems security and impact of attacks follows a quite different approach [16, 17], however we see that a similar methodology can scale to a variety of different scenarios.

PRISM in particular has been shown to be effective for reasoning about network and protocol attacks (which might be covered by a NIDS). Basagiannis et al. [27], work on a way of modelling DoS attacks for the quantification of DoS security threats. Their approach looks at protocol specific costs for participants associated to a DoS attack and potential mitigations. Whilst not centred in the context of the IoT they show a very interesting evaluation. The properties the authors observed varied from our approach of looking at distributed IoT systems as they focused on the single protocol participants, they didn't observe the snowball effect across the system that we introduced in our work [16]. Another work uses PRISM to model side channel attacks on protocols [134]. Side channels are attacks that convey information about the behaviour of a hardware or software system implementation beyond what was intended by its design - most commonly via their use of resources such as time or power [134]. The work models battery power of systems showing potential worse case behaviours and information leaks under these attacks. This approach allows to reason about resource usage in protocol verification, something that we identified as a key feature of importance in evaluating an IoT system.

Various papers have used probabilistic modelling and PRISM to observe scenarios analogous to our own. The initial contribution of this thesis focuses on the modelling of interactions of devices and the second on the modelling of attacks on these systems. These

techniques are the foundation for our attack detection technique. Based on the work done in this area we gain confidence in the usage of these same techniques for fulfilling our aim.

Chapter 3

Intrusion Detection Systems in the IoT

3.1 Chapter Summary

The procedure of implementing an IDS for Internet of Things (IoT) networks is not without challenges due to the variability of these systems and specifically the difficulty in accessing data. The specifics of these very constrained devices render the design of an IDS capable of dealing with the varied attacks a very challenging problem and a very active research subject. In the current state of literature, a number of approaches have been proposed to improve the efficiency of intrusion detection, catering to some of these limitations, such as resource constraints and mobility. In this chapter, we review works on IDS specifically for these kinds of devices from 2008 to 2018, collecting a total of 51 different IDS papers. We summarise the current themes of the field, summarise the techniques employed to train and deploy the IDSs and provide a qualitative evaluations of these approaches. We do not only present a review of what the best practice currently is, we propose means to evaluate and test the different IDSs in a unified methodology. While these works provide valuable insights for sub-parts of these constraints, we discuss the limitations of these solutions as a whole, in particular what kinds of attacks these approaches struggle to detect and the setup limitations that are unique to this kind of system. We find that although several paper claim novelty of their approach little inter paper comparisons have been made, there is a dire need for sharing of datasets and almost no shared code repositories. Raising the need for a thorough comparative evaluation.

3.2 Chapter Introduction

An IDS is in essence a monitor placed on a device and/or network that analyses incoming messages, to detect attacks and/or unwanted traffic, and when paired with an intrusion prevention system can be used to stop attacks before they affect the system. They are trained using system behaviour data or from existing attack databases, and use these patterns to make the detection. They are widely deployed in a variety of systems and can often be considered as a first line of defence against intruders. Deployment strategies differ; however, they can broadly be classified into Host Based (HIDS) and Network Based (NIDS). A HIDS monitors activities on the device itself such as system calls or shell commands to discover unauthorised behaviours or accesses. They are normally very fine grained traces of behaviour on a single device. A NIDS on the other hand looks at the network data to determine the likelihood of intrusion. This approach is more flexible when looking at large systems of devices and will have a smaller overhead on the devices themselves; however, it has less granularity. To optimise detection it is sometime desirable to mix the approaches and have both implemented on a system. This technique is often referred to as collaborative intrusion detection (CID). Whilst this approach theoretically provides a wider coverage it also needs a very structured architecture [176].

Standard approaches used to train IDSs include using a database of known attacks (*misuse detection*) and testing systems to create a “benchmark" behaviour and flag any anomaly as a potential attack (*behaviour based detection*) [120]. However it has become more and more common to mix up the approaches to make up for their respective detriments, namely the inability for misuse detection to detect unknown attacks and the high false positive rate of anomaly detection. This is generally referred to as an Hybrid IDS, which can be either a combination of these two techniques or a combination of sub categories such as rule based together with anomaly. In combination with the different kinds of deployment options described it is possible to create various different IDS frameworks to suit different scenarios.

One such scenario that this paper focuses on is the Internet of Things (IoT). This might describe anything from a smart television to a building or factory to potentially a whole city of interconnected devices monitoring various aspects; traffic levels, temperature, number

of people. These systems aim to improve various aspects of our lives [85]; however, with these improvements, several issues arise in terms of security and privacy, as made evident by various surveys [83, 89, 182].

A lot of challenges that are cited in the literature arise as an outcome of the restraints of these devices. The devices themselves are often very simple and are built to perform a specific task, with little room for variation Consequently, they are unable to be adapted to be implemented with security solutions and rely externally for protection. In such scenarios it is beneficial to implement an IDS as it can perform the much needed security without needing to change the device setup. It is of particular concern to ensure that an IoT system is secure as this emergent paradigm relies heavily on data collection and, as recent attacks show, this may lead to serious privacy concerns [141]. Whilst data is a key attack objective, the devices themselves make for desirable targets as their limited capabilities make it easy to take down whole networks e.g. through battery drainage or through gain of illicit access to the devices themselves, in turn allowing to employ them as a botnets. In these very heterogeneous and dynamic systems it becomes very difficult to find a single IDS solution and several approaches will be discussed.

There are various different approaches proposed to secure IoT systems by means of IDSs, and several other surveys have discussed them extensively [39, 188, 61]; however, whilst a summation of work is done, no work has been done towards evaluating the different approaches. We conducted a systematic review selecting papers from the year of 2008 to 2018. The papers were selected if they fit the criteria we identified in Sec. 3.4. The criteria was selected as the core metrics that made designing an IDS for the IoT different than that for a more traditional system. Our results have found that although several papers are proposing new approaches to deal with the new attacks pertinent to the IoT, very few cross evaluate with previously proposed approaches. This lack of cross evaluation is quite worrying as there are several approaches all solving the same problem, without clear guidance on why one cannot use the existing techniques. In this survey we attempt to understand why this is the case an systematically evaluate the pros and cons of the different approaches.

A unified evaluation criteria allows for informed decision making, and gives users a clearer sense of what the abilities of an IDS are. IDS Solutions for the IoT take many forms and are often bespoke to a specific scenario, so it is very difficult for a potential implementer to make a decision of whether it would be suitable for their needs. Furthermore, due to the highly dynamic nature of IoT environments the adaptability of these systems becomes an important effectiveness metric and the results need to fit their scenario. To compare the effectiveness of an IDS several metrics can be used e.g. F Score, false positive rate, mean error etc; however, results can be scenario dependent or be reliant on setup specifics. It is desirable, on the other hand, to evaluate each IDS under the same circumstances of deployment and against the same attack range to get an accurate comparison.

Whilst the current evaluations might be effective for some scenarios no single paper in the literature produces an extensive comparison or is evaluated in different IoT scenarios. In this chapter we propose a flexible assessment testbed that can mimic the characteristics of an IoT deployment under a range of attacks and with different system setups. This effort will aim to produce transparent reproducible results that can compare the different ways to deploy intrusion detection systems in IoT scenarios. Out of the fifty one surveyed works only three provided source code artefacts in their papers. Although we attempted to contact all authors of the works to compare the various approaches, we were only able to obtain one further artefact. This discovery is a very worrying trend, as new techniques keep getting proposed it becomes unclear as to the strengths and weaknesses of each approach. Without open access to previous work it becomes impossible to build on top of previous efforts making it harder to innovate the field and leading to repetition of efforts and lack of advancement.

The contributions of this chapter are the following; The contributions of this chapter are the following;

1. A comprehensive survey of the current state of the art in IoT intrusion detection consisting of 51 papers;

2. A break down of different approaches and designs currently proposed;

3. A summary of the current literature assessing pros and cons of the different approaches.

4. Proposal of a testing environment for deployment of security solutions on virtualised IoT systems.

The remainder of the paper is broken down as follows::

- Section 3.3 provides a list of related works surveying IDSs in the IoT;
- Section 3.4 provides a discussion on adjustments and specific requirements to deploy and IDS in the context of IoT;
- Section 3.5 provides a breakdown of deployment architectures for IoT systems;
- Section 3.6 provides a breakdown of techniques and algorithms used to train and deploy IDSs as well as their limitations and advantages;
- Section 3.7 discusses current evaluation criteria and methods, the way IDS are currently evaluated and a discussion on their effectiveness.
- Section 3.8 summarises our literature review of 51 proposed IDS tools for IoT, categorises them and breaks down each approach;
- Section 3.10 describes our design methodology for the IoT Testbed proposed for evaluation;
- Section 3.9 concludes and discusses current trends, challenges and the current gaps in the literature;

3.3 Related Work

Despite the maturity of the field of IDS research, the current IDS solutions are inadequate for wide usage in IoT deployments. To address this difficulty researchers have proposed new means to do intrusion detection that can adapt to the IoT constraints and circumvent them. In this survey we investigate the current state of intrusion detection, present the difficulties in adapting different IDS training/detection techniques to the IoT and comprehensively summarise what the state of the art in IDS in the IoT is.

The discussion of how to protect IoT systems by means of IDSs is a very hot topic for research and several other works have reviewed the literature of IDSs in IoT or similar paradigms [39, 188, 61]. Whilst a summation of the work in the area exists, little to no work has attempted to answer the questions of how to quantify an IDSs effectiveness in the scope of an IoT System.

Prior to the IoT, another form of constrained network environment was wireless sensor networks (WSN). A WSN would be considered as a sub category of a IoT system, with very specific configurations and several of the same key security concepts and limitations. Butun et al. [39], provide one of the early works in surveying IDSs for Wireless Sensor Networks. The authors also progress to discuss MANETS or Mobile Ad hoc networks, another constrained network configuration that is more dynamic than WSNs. The survey talks about some of the most common approaches used in the detection of attacks on these systems. Unlike a lot of IoT configurations, WSNs are often homogeneous, this allows for a lot of statistical approaches to be used which are often not able to capture attacks in the IoT. The authors provide an example of this using forwarding percentage of packets by a node, and discuss the effectiveness of these approaches. The authors discuss a lot of the techniques used to detect attacks in WSNs putting specific focus on the advantages of using each of the approaches. Unlike the IoT these systems don't have downsides of multi protocols, and interconnect multi network systems and therefore there is much less focus on how the IDS is deployed and how the different detection agents may interact. The evaluation of the approaches revolves around their applicability to scenarios and no formal comparison of the approaches is provided. This nonetheless is an extensive and very good analysis around the techniques that can be used in MANETS and WSN and how they work.

Departing from WSNs, Zarpelão et al. [188], propose one of the first efforts classifying IDS research for the IoT. Their aim is to identify leading trends, open issues, and future research possibilities. Their proposed classification is structured around the: detection method, IDS placement strategy and security threat. The authors discuss the IDS from the point of view of how the IDS is deployed, how the detection is done and also what attacks are detected. The survey takes a form of a literature summary where for each paper a

summary of these three concepts is provided. The authors also discuss the issue of how these approaches are validated. The authors find that many different approaches are used, and no single strategy is used. This is mentioned as a core issues as it makes the comparison of the different approaches difficult, not to mention some approaches didn't provide any validation at all.

The most recent survey for IDSs in the IoT, to the best of our knowledge, is the work of Elrawy et al. [61]. In their work the authors extensively discuss the structure and architecture of the IoT, potential threats, and different applications. A summary is provided of techniques used in different papers, with a summary of their advantages and disadvantages. The structure provided is informative and does a good job of introducing the different concepts prior to discussing the IDSs. The authors then provide a short summary of each paper and summarise the techniques they use and deployment strategy in a table. the survey also discusses the authors evaluation results, the intuition of this is very good however, as mentioned by the authors, due to the different ways of evaluation is is impossible to draw comparisons from these results.

Across these surveys there are many shared themes. As different advances are done in network architectures several new approaches are developed to defend them showing the need for new analysis and surveys. They recurring trend is also that there is a lack of evaluation of these approaches, so none of the surveys are able to compare the effectiveness of the different IDS. Each of these reviews have chosen around 20 different IDS papers, and our more extensive review has collected over 50, and yet we have no easy way to compare the different approaches. This makes it difficult to establish open problems, choose the best approach for a scenario and to assess the validity of the proposed literature. This raises the need for ways to compare different IDSs something which we address in Section 3.4.

3.4 An overview of Intrusion Detection for the IoT

As the IoT takes over several key aspects of our lives, including homes, factories and even healthcare, so it becomes very important to keep them secure. However, these systems vary

greatly and function differently from traditional internet systems, so new solutions need to be devised to ensure their safety. From a security perspective it is important to understand the implications of the IoT on how solutions are designed. We know for certain that some specifics of IoT make it impossible to implement certain solutions, e.g. resource constrains such as limited battery life, low bandwidth, small processing capabilities, and memory constraints make computationally intensive security protocols impossible to implement. However it is less certain how these impact the ability for an IDS to be implemented successfully. We identify four main concerns to consider:

1. *New kinds of attacks* which are often not pertinent to standard systems. Consequently arising the need for the IDS detection to be expanded. Taking these observations in mind current solutions need to be adjusted to be able to detect these new avenues of intrusion. In traditional networks, the system administrator deploys IDS agents in nodes with higher computing capacity. Whilst in the context of IoT networks which are usually composed of nodes with resource constraints, this may not be an option. We summarise a list of new vulnerability vectors in the IoT that would impact the ability for an IDS to detect intrusions as:

    i. *Node compromise attacks*, devices in IoT systems are often vulnerable to physical takeover, allowing for malicious behaviour from previously benign members of the network;

    ii. *Communication errors*, due to deployment in disadvantaged environments, IoT is very prone to network errors which can be a source of irregular traffic;

    iii. *Battery drainage attacks*, the constrained resources may lead to selfish device behaviours, due to device self preservation constant polling may lead to battery drainage so devices will stay offline for large periods of time, leading to difficulty in behavioural patterning and harder network diagnostics;

    iv. *Routing attacks*, the IoT is often deployed as an open network through WiFi or similar technologies, this opens up the network to external connections as well as new attacks to do with routing, such as sinkhole attacks;

v. *Compromised communication*, as a consequence of low bandwidth it is difficult for devices to reach far away destinations,so the network configuration becomes very important, therefore, if a node is identified as a key connector between two of the devices and is consequently compromised, communication in the network is broken;

2. *Placement* of the IDS itself represents a unique challenge. In traditional networks end systems are directly connected to specific nodes (e.g., wireless access points, switches, and routers) that are responsible for forwarding the packets to the destination [188]. In the IoT on the other hand the network may have

3. *Multi hop routing* and even be partially or fully *disconnected*, so the ability to survey the full traffic may be infeasible.

4. *New communication protocols*, most modern IDSs are built to work with traditional web protocols (HTTP, TCP, REST architecture etc.), due to the constraints of these devices the IoT often operates on completely new protocols (6LowPAN, CoAP, ZigBee etc.), making traditional IDSs simple unable to comprehend the protocols. To compound this even further there is an issue of *multi protocol systems* that are very common and require even more adaptation.

These core issues, are the main principles in consideration when creating an IDS for IoT systems.

3.5 Types of Intrusion Detection Systems in IoT Context

The IoT does not reside in a bubble, it is very much integrated with the various internet infrastructures such as cloud, fog and edge computing; therefore, these need to be considered as key components of an IDS design strategy. When talking about IoT it becomes difficult to generalise in the same way traditional IDS classifications have done; one cannot, for example, easily say that the IDS is placed centrally or distributed, as where it's placed as

well as how it is placed, is a core factor of importance. All these options have different security and performance implications. If a solution claims to cover the whole of the IoT such as the solution proposed by Raza et al. [148] then the solutions needs to have a strategy for working in each of these scenarios, it is not simply enough to show how it solves just one of the restrictions (e.g. new protocols). The location of an IDS also largely depends on what type of IDS it is. IDS types are commonly split into Host based (HIDS) and Network Based (NIDS); however, a further categorisations can be made when applied to the context of IoT Systems namely, Collaborative IDSs (CIDS) which involve the collaborations both a Network based and Host Based IDS; each of which could potentially be deployed across various networks/sub-networks. Each of these approaches have different uses and scenarios.

3.5.1 Network Intrusion Detection for IoT

A NIDS is an IDS that monitors network traffic to detect remote attacks i.e. attacks carried out over a network connection. These kinds of attacks work at the higher level of the stack often targeting vulnerabilities in protocols; however, their effect can have high impact on the device themselves. One of the most common network attack is a DoS attack, a DoS attack aims to target the availability of a device or network and due to the constraints of IoT devices they can be particularly effective. Traditional safety measures that can be deployed to protect against these kinds of attacks i.e. security protocols, often cannot be ran on these devices due to their constraints [186].

One of the core differences between most IoT systems and standard infrastructure is the protocols at play. In the IoT there has been a shift from more traditional protocol to new technologies to cater to the constraints of devices. Examples of this is the movement from IP to UDP, IPv4 to IPv6 and even more specialised protocols such as RPL which is specific for constrained networks. With the complete change in the underlying infrastructure the current NIDS techniques may simply just not function on these kinds of systems. To tackle these challenges various new techniques are developed specific to these new protocols.

Within the IoT we also have a raise in popularity of what are often referred to as unsupervised systems, these systems are deployed using the concept of machine to machine

(M2M) interactions only, and therefore will operate largely unsupervised by humans. As an outcome of these scenarios a compromised node will often go unnoticed, allowing for an attacker to lunch attacks from within the network. These kinds of attacks cannot be blocked by an external facing firewall and therefore the deployment of an internal IDS is essential [186, 112, 92]. Not only is the issue with internal attacks and lack of authentication of devices, there are also new attacks raising in popularity due to the infrastructure of these systems. One of the components of unsupervised systems is *dynamic routing*, this means that routes are constructed based on shifts in the system, signal strengths and resource optimisations. As an outcome of this it becomes beneficial for an attacker to disrupt this process and route packets in either a less than optimal way or a way that benefits his interests e.g. through a compromised node. These routing attacks have raised hugely in popularity in WSN and IoT systems some of the most famous being, Wormhole attacks [1] and Sybil attacks [2], both of which rely heavily on the system being unsupervised and self adaptive. These kinds of attacks have spawned various research papers specifically catered to detecting them, as we discuss in Sec. 3.6.

Changes to the infrastructure do not only have *negative* effects on intrusion detection. The restrictive nature of these systems may come as an advantage from an attack detection perspective. Due to the limited behaviours of the components of these systems new techniques that would not be suitable in standard networks are being developed and explored. Work has developed using automatons [68, 125] for behaviour modelling as the limited behaviours allows to pospone the state explosion issue that traditionally prevented it. Automata based approaches allow for system analysis as well as behaviour modelling making for more actionable IDSs. The set of simple behaviours has also allowed for game theory based approaches [155], to find the optimal behaviour of a simple system. And the raise of popularity of clustering approaches [56, 86, 112], which rely on grouping similar behaving devices in a cluster to improve detection of misbehaving nodes.

Deployment strategies also play a big role in how intrusion detection is done, this does not only have to consider maximum coverage but resource constraints as well. The vast majority

[1] https://www.sciencedirect.com/topics/computer-science/wormhole-attack
[2] https://www.sciencedirect.com/topics/computer-science/sybil-attack

of IDS systems for the IoT implement distributed IDSs. This involves sharing of information between sub-networks, which adds complexity if there are contrasting information and around trust in reporting. This also is reflected in the amount of data that needs to be processed by an IDS, in some cases with large systems it simple becomes infeasible. A relatively new area of research has been focusing on reducing this overhead using dimensionality reduction techniques and smart data processing [112], to more effectively deal with alerts. As each component in the system needs to report data in a distributed manner, authors have considered trust based scheme to ensure the quality of reported data is maintained [186, 43, 92]. All these changes in infrastructure, protocols and the types of system at play are key factors that need to be considered when designing a NIDS for IoT systems.

3.5.2 Host Intrusion Detection for IoT

A host based IDS refers to an IDS that monitors the activities on the device (i.e., the host) where it is deployed, to detect local attacks i.e. attacks executed by users of the targeted system or attacks directly impacting the device operations. These types of IDS may monitor operations at a lower level than the network based IDS and they have access to much more details of the impact on the system itself. Whilst a network based IDS might only see packets travelling through the system, the HIDS also have access to system logs and can monitor metrics of the device behaviour. One of the recurring difficulties around converting IDS to work on IoT systems is around the low computational power of the devices. This is particularly relevant in the scope of HIDS, as these limitations greatly restrict how many operations can be done on the device itself, and the data overhead it can handle.

Devices in IoT systems often dedicate what little computing power they have to providing features or services. Design constraints, such as the need to increase performance or battery life, may restrict the ability of system designers to implement security effectively. Furthermore traditional approaches such as rule-based based schemes are unsuitable for these kinds of systems due to the large data overhead. Therefore, anomaly based detection methods, which attempt to identify deviations in measured statistics against a normal model of operation of a system, can be beneficial to use in resource constrained systems.

One of the most common advances in the field of HIDs is the way data is handled at the device level. Most approaches optimise data handling to reduce state space and cleverly optimise data processing. Data processing optimisation, can be achieved by feature engineering, this process is achieved by carefully selecting relevant features of data and reducing its dimensionality to decrease the number of operations needed to be conducted by the device. If done carefully, this approach is still able to preserve the patterns of behaviour within the dataset making it still useful for anomaly detection, whilst greatly reducing overhead. One way feature engineering is applied in the IoT is through bit pattern matching [165], this allows to store a small chunk of the dataset in a lookup table, greatly reducing the number of comparison operations.

Further work takes advantage of the devices limited behaviour for effective anomaly detection. Small resource constrained devices execute fewer and potentially less complex operations than general purpose computing platforms. This results in less complex patterns of communication and behaviours, making it easier to detect when such patterns have changed. A raise in popularity in these cases is the use of immunity based techniques. The immune system has been successfully applied to the information processing domain. In particular, it performs complex computations in parallel and decentralised patterns. Furthermore, an immune system can learn new information and recall learned information. Due to the decrease in complexity authors have been able to train these systems a lot more efficiently [111]. Another further development has been the increase in use of pseudo or full modelling techniques. Using Hidden Markov Models, IDSs are capable of learning the system behaviour an find anomalies. To render this technique less resource intensive weak-HMMs are developed. Using domain knowledge of system behaviour the formation of the HMM can be guided to make less memory intensive effective anomaly detection schemes [162].

Host based systems in the IoT have some further challenges that need to be addressed. A DoS attack against an IoT network has the potential to be significantly more detrimental than one against a standard network. This increased vulnerability is due in part to the low computational power and battery power characteristic of IoT devices. This has lead to a raise in popularity of battery drain denial of service attacks [16, 17], these attacks target a

device to perform power drain intensive operation to drain the battery. If they are successful it requires human intervention to change the battery something that is to be avoided in large unsupervised systems. Authors ave proposed battery monitoring techniques at the host level to detect these kinds of attacks [107]. This is an additional feature of interest that traditional IDS techniques do not consider. However, as devices often do not have the ability to self monitor their battery drain [17], it becomes critical for an IDS to monitor these behaviours.

3.5.3 Collaborative Intrusion Detection for IoT

A collaborative IDS makes use of different IDSs together in a single system. It may include both a network intrusion detection system as well as Host based ones. Or as is often the case in IoT systems, the system spans across various subsystems composed of various devices and using different technologies, to safeguard these systems different kinds of IDSs need to be in place and they need to cooperate to prevent multi-protocol attacks and cross system intrusions. A collaborative IDS may span various different sub-networks, communication protocols and even geo-locations, it may be all network based or contain several different types all linked together. This distributed nature makes the design of an IDS particularly complex.

One of the main drivers behind the need of collaborative IDSs is the multi-protocol nature of these systems. This scenario may render several current techniques less effective and be unsuitable to a single IDS scenario. This is especially pertinent in the case of pattern matching IDSs as the same attack across different protocols may take a different form, and therefore bypass these techniques. To cater to these difficulties literature has proposed data aggregation methods that allow for, non protocol restricted pattern matching and behaviour analysis [187, 49]. These types of approaches make use of local aggregator nodes collecting specific data to a sub-cluster or sub-component of a system, which then communicates to a central component (in a specific format) and centrally does the pattern matching [49].

A defining feature of IoT that is being considered across all configurations of IDS is their constrained nature. Their restricted capabilities lead to the development of simpler detection engines. This is often fine as their behaviour is also restricted, however, a simpler

engine is not capable of detecting attacks across multi protocols and multi systems, which in cooperation may exhibit vastly different behaviours. These raises the need for several different systems to be deployed. In these circumstances, the core difficulties are around how the data is communicated across the network. If an anomaly detection finds an anomaly in a sub-part of the system it is difficult to understand how it will spread across the other components or if it would spread [191]. To do so authors propose specific risk assessments across the different layers, with efficient head nodes able to assess the impact of attacks at the different levels [191].

One of the biggest issues around collaborative IDSs is around the authenticity of communication. When you have several different monitors and reporting mechanisms spread across a large network it is essential that the data is authentic. Wrongful reporting, or illegitimate information may raise false alarms and cause meaningful damages to the system. This is exacerbated in the IoT as these devices may struggle to implemt full cryptographic solutions that resolve these problems in standard systems. To mitigate these circumstances authors have proposed ways to new ways to define behaviour of devices to validate if they are indeed communicating in a benign way [34, 157]. These approaches are particularly effective in the context of insider attacks. As these are recognised components within the network they may cause a lot of damage. Authors therefore propose metrics based on expected behaviours [157], this creates a level of trust between the devices that may be altered as to replace traditional means of authentication.

3.6 Techniques for use in Intrusion Detection Systems

Within the various different approaches to construct the IDSs, there are techniques and algorithms used for training the IDS and constructing rules. These techniques vary from carefully curated databases to complex machine learning algorithms and may be used to construct any of the different types of IDS.

3.6.1 Rule Based/Misuse Detection/Policy Based

Misuse detection also referred to as Rule based and Policy based approaches generally rely on rules of the known intrusions taken from known attack databases. They are considered efficient and are used in most real time systems as they need less processing power than behaviour based detectors, and thus can handle large volumes of traffic without slowing down the normal activities. Their major criticism is the deficiency in detecting zero-day attacks i.e. the attacks which are previously unknown.

In the context of IoT, these approaches become particularly pertinent to very simple systems where known issues arise. Particularly in the case of WSNs, there are several known attacks, specifically routing attacks, that may not be stopped by careful programming, as these kind of systems are vulnerable to these attacks by their very construction. It is therefore desirable to deploy an IDS with preset rules for such attacks. The downside of these approaches is that patterns may be easily bypassed by careful packet obfuscation or slight changes to known behaviour. One approach that has been popularised to counter this is the usage of automated rule learning techniques [187]. The automated learning of rule has the advantage that it is more adaptable and thus better able to deal with new attacks. However this is a backward facing approach as the rules are based on the current behaviour and only after the attack having taken place can a rule be automatically generated.

Advantages: Accurate in finding known attacks, easy to deploy, widely used. [58, 181]

Disadvantages: Cannot yet find unknown attacks, tends to be protocol specific, isn't customisable to your setup, may have high memory overload [24].

3.6.2 Signature Based

An often stronger type of detection, although still pattern based, is signature-based IDS. Signature based detection refers to the detection of attacks by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware. Although signature-based IDS can easily detect known attacks, it is difficult to detect new attacks, for which no pattern is available. An area that has shown a lot of promise

in the context of signature generation is through automatons and model based signatures. The restricted behaviours of these devices allow the creation of modelling techniques to formalise the attack signatures [125, 68]. These approaches make use of the automata to specifically model the communication of a protocol or behaviour of a device, with each transition being a message exchange/action performed. The incoming data or behaviour of the device is then encoded and compared to the modelled approach and if deviating marked as anomalous. This approach is not only a new way to describe behaviours, modelling has several key advantages, being able to reason, and calculate system wide properties could give security professionals key insights about their system that would not be possible with traditional approaches. This approach is relatively new, but could be extended to not only manually create signatures, but our previous work has proposed modelling based techniques for the automated generation of stochastic attackers [17, 18]. Although in early stages, this approach may be able to bypass a lot of the current downsides of signature based approaches to encompass unknown attacker behaviours.

Advantages: Accurate in finding known attacks, bespoke to the system unlike specification based. [58, 181]

Disadvantages: Hard to train attacker behaviour, may have high memory overload [24].

3.6.3 Anomaly/Statistical

Anomaly/statistical-based intrusion detection systems were primarily introduced to detect unknown attacks, in part due to the rapid development of malware. The basic approach is to create a model of trustworthy activity, and then compare new behaviour against this model. Although this approach enables the detection of previously unknown attacks, it may suffer from false positives, which are previously unknown legitimate activity is classified as malicious.

There are two strategies to anomaly based detection 1) Black Box, referring to training without domain knowledge of the system, often done using neural networks; and 2) White Box, in this case the semantics used by the detection system are an abstraction of the underlying system. In the contexts of a lot of modern machine learning problems, black box

approaches are very popular. They allow a relatively simple push to go approach, that might find patterns in data or be quite accurate at classifying data, however they provide very little insight into *why* they make decisions. The problem is that black box detection system face the challenge of transferring their results into actionable reports [161]. Actionability however is a very difficult metric to measure and is not often considered in academic approaches [64]. In our review only a single paper considered actionability as a metric, Buennemeyer et al. [38], in which the authors evaluated their IDS using a user study. The second approach is white box detection, this approach requires much more curating than the black box system. It requires the building of behaviour profiles either automatically or manually so that a malicious attack may be linked to the constructed profiles. Whilst this is often impractical in highly complex systems, the IoT allows for some easier behaviour modelling, greatly facilitating the usage of these methods.

Building system behaviour is an essential part of any effective anomaly detection technique. Statistical approaches, which are a subcategory of anomaly detection, are particularly effective in homogeneous systems such as WSN. A statistical anomaly approach, is a white box approach that relies on knowledge of the devices behaviours. Parameters are selected and specified for each device [107], or averaged across the network behaviour [45]. However, approaches using these techniques are prone to large amounts of false positives, in the work of Cho et al. [45] the authors propose that an anomaly occurs when the packet size is above average, this kind of anomaly detection will obviously lead to issues, if for example the sensing environment changes. To mitigate the lack of adaptability authors propose dynamic anomaly detection techniques which can dynamically change the expected *normal* behaviour. Authors using this approach [30, 116], use a central gateway to monitor system operation and update the expected behaviours of the various IDS components. Specifically, Luo and Nagarajan [116], make use of autoencoders as their primary detection mechanism, the output of the detection is then sent to a central node, which, given the results recomputes the autoencoder and sends the new version to the local detectors. Whilst this approach tackled the downside of adaptability it is unable to provide insights like a white box detection would. These

difficulties are further complicated in heterogeneous systems, where white box behaviour is much harder to map.

Not only is it very difficult to have an anomaly detection approach that is both adaptable but also actionable. Further difficulties arise when trying to actually collect the behaviour. Two main behaviours need to be considered, one is at the device level, and one at the system level. At the device level (often done by HIDS), anomalies are categorised by changes in expected behaviour, which may be classified as insider attacks, device take over, or the device being targeted by an attack. At the network level the general behaviour is evaluated against the expected operation of the system. However, some of the characteristics of IoT systems are that, behaviours might change, devices may go offline, and devices might wake up after long periods of time, therefore, getting a baseline behaviour is difficult. To construct baseline behaviours large amounts of data is needed and due to the lack of public data, researchers are forced to assemble their own datasets. However, in general this is not an easy task, as most lack access to appropriately sized networks or if done in production environment it may only collect a window of behaviour; as extensive testing is time consuming and disruptive to normal operations. In the context of the IoT data collection also needs to be aggregated. If working across networks and systems, results of analysis need to be transferred across components.

Advantage: can detect unknown attacks, dynamic (if new data is collected) [140, 97]

Disadvantage: suffers from false positives, lack of datasets or realistic testing environments, hard to adapt, hard to act upon alerts [170, 115, 64].

3.6.4 Stateful

Stateful protocol analysis is the process of comparing predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events to identify deviations. Unlike anomaly based detection, which uses host or network specific profiles, stateful protocol analysis relies on specifications of universal profiles that dictate how particular protocols should and should not be used. The “stateful” in stateful protocol analysis means that the IDS is capable of understanding and tracking the state of

network, transport, and application protocols that have a notion of state. A stateful detection methodology is very similar to signature based approaches, it differs in term of what is being detected. Signature based approaches are looking for signature of attacks, conversely stateful detection is using the protocol exchange signature as the means of detecting good behaviour.

As most common stateful detection approaches are based on traditional protocols, bespoke stateful approaches are developed for IoT systems. Further issues however arise in the context of constrained IoT networks as the unreliability of the communication and usage of UDP rather than TCP, may lead to a lot of false positives. This approach suffers from a lot of the downsides of rule based detection as it is highly non adaptable. It is likewise very actionable as the relatively simple state based description of the system allows to easily see where the pattern has deviated from expectation. However, in practice, working from a specification of expected behaviour that doesn't consider implementation specifics has some downsides. In the field of protocol analysis we see that specs often do not cover implementation specific behaviours or allow for multiple correct behaviours [19], further complicating the detection. Furthermore, protocol specifications may be prone to attacks, and correct behaviour within the protocol may lead to data leakages or attacks that would not be detected by this approach. In large IoT multi-protocol IoT systems, protocol specific detection may be too restricted and need for various different detection components, making a general behaviour based approach much more effective.

Advantage: Identifies unexpected sequence of commands [153, 181]

Disadvantage: very memory intensive, cannot detect attacks if they are within protocol behaviour, if protocol implemented different to specification it may cause false positives [181].

3.6.5 Clustering

Cluster analysis is defined as the technique of grouping data objects based on the information found in it that describes the objects and their relationships. The primary goal of clustering is to separate objects such that there is higher similarity within objects of a group and higher difference between the groups. By clustering a device within the systems together with other devices you can make general assessments about it e.g. if the device is clustered with known

malicious devices it can be assumed to be malicious. Clustering is a highly popular technique in constrained systems as their restrictive behaviours makes the cluster creation much easier, as there are few multi purpose devices fitting to multiple clusters. This technique however is mostly used in WSNs and homogeneous systems [56], as clustering heterogeneous systems may be much more complex.

Clustering techniques may extend beyond the clustering of devices. Clustering techniques are used for message classification and also for efficient data processing. As there is a huge amount of data collected in IoT systems, and the devices are less able to handle it, clustering techniques have been proposed to more efficiently process it [112]. These clustering approaches are used to categorise data into normal behaviours and at risk behaviours as well as to perform dimensionality reduction. This allows to greatly reduce overhead of the IDSs and allow for faster computation. This approach is emerging to deal with the issue of big data, and their approaches make it so that relatively low information loss is achieved whilst greatly reducing the dimensionality. These techniques re very helpful hand in hand with other anomaly detection techniques as they help improve their efficiency significantly [112, 74, 43].

Advantages: Works well on static data, can find some very specific attacks [180, 110]

Disadvantages: Not efficient on dynamic systems, works if data with strong correlations hence struggles with heterogeneous systems [178, 46]

3.6.6 Computational Intelligence

A system is computationally intelligent when it: deals with only numerical (low-level) data, has pattern recognition components, does not use knowledge in the artificial intelligence sense; and additionally when it (begins to) exhibit (i) computational adaptivity, (ii) computational fault tolerance, (iii) speed approaching human-like turnaround, and (iv) error rates that approximate human performance [193, 50]. Although there is not yet full agreement on what computational intelligence *exactly* is [50], there is a widely accepted view on which areas belong to CI: artificial neural networks, evolutionary computation, artificial immune systems, swarm intelligence, and soft computing. These approaches are capable of autonomously acquiring and integrating knowledge, and can be used in either supervised or unsupervised

learning mode. Computational intelligence differs from Artificial intelligence a field widely used in multiple other techniques listed above.

Artificial immune systems are a relatively recent advance in computational intelligence, their self adaptation and robustness makes them a very interesting approach for intrusion detection in the IoT. Artificial Immune Systems are adaptive systems proposed by Leandro Nunes et al. [42]. They are inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving, as such they take specific components of the immune systems as inputs. The main difficulty in the usage of this approach is the matching from network data to the input of the immunity algorithms. Their ability to self adapt makes them particularly pertinent to systems where configurations might change and attack circumstances might be persistent, however will not be immediately useful for detection. This approach may be particularly useful in the detection of physical intruders or changes of behaviours from a device. Arrington et al. [21], make use of these techniques to detect human intruders within a smart home, based on sensor data from the room. The rise of popularity of this technique is in part due to the easier mapping of behaviours to the behaviours of a cell in constrained systems, as Liu et al. [111], propose attack signature matching to "antigents" and normal behaviours to "self elements" which is in part possible due to restricted nature of IoT.

Another popular form of computational intelligence is, swarm intelligence. Swarm intelligence is defined as a collective system capable of accomplishing difficult tasks in dynamic and varied environments, without any external guidance or control, and with no central coordination. The definition is self explanatory as to why it would be beneficial, however very few approaches have explored its used in the context of intrusion detection. The specific application of these techniques comes in the form of behaviour discovery. As these systems are often unsupervised, routing patters are often pre-fixed, or dynamically discovered. One technique that can be used to find optimum routing paths is Ant Colony Based Optimization (ACO). ACO uses the concept of Stigmergy, which is the indirect communication via interaction with the environment. Each packet (or ant) leaves a pheromone trail along the path if a successful result is reached. Paths with the most pheromone are

considered the optimal paths. This is of course adaptive as if paths are reconfigured new trails will be discovered as the new optimal. This technique is used to find routing attacks by Arolkar et al. [20]. Routing attacks are characterised by malicious routing of packets to either drop them or decrease throughput, so if a large number of packets is being routed on unoptimal paths these attacks are easily detected.

Advantages: Good for known attacks, can identify wide range of attacks given enough data, protocol agnostic (given enough data) [184].

Disadvantages: This approach relies heavily on pattern matches, whilst attack behaviour often only takes place once and varies significantly in between attacks [151] due to numeric analysis requires lot's of data pre-processing and therefore needs human supervision or doesn't scale well to different datasets.

3.7 Evaluation of Intrusion Detection Systems for IoT

The review we have conducted has found that almost all work in the area of intrusion detection focuses strongly on accuracy as means of evaluation. What we also found is that, despite almost all papers claiming there is a need for new techniques and new approaches, they all present results with very high accuracy of prediction. These two statements are contradictory, if the only metric of evaluation is that the IDSs are great at detecting attacks, then there would be no need for further IDS being developed. We acknowledge that some of the proposed solutions aim to tackle very specific areas for which there were no previous existing IDS, however for the majority of the works this is not the case. The real issue we have found is that to be a successful and more importantly a useful IDS, further characteristics are required than prediction accuracy.

In the remainder of this section we will present the current ways IDS are evaluated, we will discuss why we believe accuracy is not the sole metric of value and propose new evaluation criteria supported by literature and our own investigation. We then propose a technique to evaluate IDSs for the IoT by these criteria and our case study evaluating the surveyed approaches following this methodology.

3.7.1 Methods to Evaluate Intrusion Detection Systems for IoT

With the increasing variety and complexity of IDSs, the ability to evaluate them becomes a key concern. In order to ascertain which IDS is most suitable for a specific task a clear evaluation needs to take place, for instance one might wish to see which IDS is most suitable to detect attacks on their system, they might wish to test different configurations of an IDS to inspect performance. Whatever the need may be it is key that means are in place to evaluate multiple IDSs, as to allow for informed decisions. There has been a lot of work in this area as discussed in the survey by Milenkoski et al. [122], the main differentiating factors in testing an IDS is whether one were to use a *benign workflow*, a *malicious workflow*, or a *hybrid workflow* and any of these can either be *live* tested or tested using *trace* based approaches.

A *benign workflow*, is a purely normal system behaviour, these may involve CPU intensive workloads, I/O focused workloads, network intensive workloads and system wide workloads to test the hardware as well as the operating systems. Several tools are available for the testing of traditional IDS each with focus on different aspects of workload (e.g. httpbench [3] for network testing using HTTP). However these tools may not be suitable for IoT systems as they are either using the wrong network protocols, be operating system dependant or simply not relevant since for example sensors might not do any I/O operations.

A *malicious workflow*, is used to test the attack detection accuracy of IDSs. These workflows take the forms of attack scripts targeting the system, they may be manually constructed or already existing tools may be used. The manual process is very time consuming as it needs to collect various different attack scripts from many different available databases. Once the attack scripts are found, they may need to be adapted to fit the scenario, this may require expert code analysis and security knowledge, this is particularly compounded for scripts aimed at systems to test host based IDSs as tricky configurations may be needed1 [120]. The process of manual collection is also limited, not all attack scripts are available online and even though there are some excellent resources (e.g. expoitDB [4] and mitre [5]), manually curating a list of of scripts is time consuming, complex and often bespoke to only one scenario.

[3] http://freecode.com/projects/httpbench
[4] https://www.exploit-db.com/
[5] http://cve.mitre.org/

To alleviate this issue ready made tools are available, these tools, referred to as penetration testing tools, are able to scan for vulnerabilities and launch various pre-set attacks on systems with much less need for expert configuration. Popular pre-prepared attack environments such as Kali Linux OS [6] and metasploit pen-testing tool [7] (available on Kali Linux) are able to act as *jack of all trades*, able to target a variety of system but not specialising in any specific type or vulnerability. Whilst this second approach is much less time intensive it is also less specialised. Both these approaches suffer from the downside that an attacker may use multi step approaches and scout the system prior to attack, this complex behaviour as well as unknown attackers could not be covered by these techniques. Finally, a *hybrid workflow*, combines these two techniques to test both the system under *benign* and *malicious* behaviour.

There are two ways to examine the different kinds of workflow, the first way, *trace* based, will use existing datasets or synthetically generated data to test the IDS in a non live environment; the second approach, *execution* based, will test the IDS in a system using live attackers and/or data. The generation of traces may take the form of an existing dataset, several standard datasets exist from a variety of sources, the most famous being the KDD datasets from DARPA [8]. Despite these datasets being around since 1990/2000/2001 respectively, they are still widely used, they allow to test network based and host based IDSs making them attractive for a variety of research. Perhaps the most desirable aspect is that one can easily compare their results to those of other authors. The downside is that they are quite outdated and would not truly be representative of modern systems [161]. Alternatively one may wish to generate bespoke traces. One common approach is to use a testbed. A testbed has many advantages to get more pertinent traces, however construction of a realistic testbed may be expensive and challenging. If a testbed is too simplistic it may not capture activities of a real-life environment [161]. Finally, one may use a real world production environments to generate traces. This is by far the most attractive option, however, it is often unfeasible. This approach may also only capture snapshots of system behaviour, as it does not have the full flexibility to run different scenarios like a testbed could.

[6] https://kali.org
[7] https://metasploit.com
[8] http://kdd.ics.uci.edu/databases/

The second way is *execution* based evaluation which can be run in a testbed or production environment. Its main attractiveness over a trace approach is that it is able to capture several more metrics than a trace evaluation. Metrics can be run for performance, overhead, ability to scale to different devices and many more, making it the more complex but more precise option. After a workload is determined evaluation should be performed based on metrics.

Metrics are widely characterised as *security metrics* and *performance metrics*. Performance metrics refer to how the IDS is able to cope with the inflow of messages and how the IDS impacts the ability for the system to operate effectively. These may be evaluated on the basis of throughput, delays, computational cost etc. However, these are difficult to benchmark and may be implementation and system dependent [122]. In our review of 50+ IDSs for the IoT, none ran performance based evaluation on execution based environments (although some did so using simulations).

The second and much more common evaluation metric is *security*, these metrics are all about evaluating the IDS attack detection capabilities. The core evaluations that can be performed are 1) False-negative rate, the percentage of attacks that are missed; 2) True positive, the percentage of correctly labelled attacks; 3) False positive, the percentage of non attacks which are labelled as attacks; To get a complete evaluation of accuracy these are all combined. Some more complex analysis may attempt to calculate costs of mislabelling (i.e. false negative/positive), based on the impact of the attacks. As the reader may note, these metrics are exclusively evaluation of attack detection and do not consider several other more human aspects of security that should be considered just as valuable [63].

Suggested Comprehensive Evaluation

An IDS is a complex system whose intent is to provide information to the security professional about potential intrusions to their system. Naturally it is essential for this system to be accurate, however, several things are also needed to make the IDS *useful*. Papers usually indicate a detection rate of 90% and above, using various evaluation methods, what would be more realistic in practice is if these same IDS could get a fraction of those results in realistic environments. For usability purposed False Positives are possibly the most disrupting factor

as they may cause the security professional to break legitimate workflows, lose time and money chasing non existing attacks and cause more disruption than a false negative could. In their talk at ESORICS2017, Etalle et al. [63], propose three further metrics specific to the evaluation of IoT IDS: 1) Actionability, not unique to IoT, actionability refers to how descriptive the alert is, this metric takes one step forward in attack detection as, whilst it is useful to detect an attack, if the alert is without context it is impossible to fix it; 2) Adaptability, very specific to IoT systems, devices change continuously and for certain types of IDS the cost of adapting (such as anomaly based detection) may be very high; and 3) Scalability, a lot of traditional IDS are designed for single servers or small networks, however IoT systems could include thousands of devices and this needs to be a key consideration. A combination of these metrics together with the evaluation of accuracy would make for a much stronger analysis and lead to better IDS solutions.

3.8 Tools for Intrusion Detection in IoT Systems

As an outcome of the differences and challenges inherent with designing IDSs for the IoT, several authors have proposed new tools and mechanisms to defend it. In the following section we produce an executive review of all new works tackling attack detection in constrained IoT environments since 2008. We break down the work into network based tools, host based tools and collaborative tools in the following section. An initial analysis and review is presented in Sec. 3.8.1, further categorisation is available in Appendix. A, a further summary of each paper is omitted for brevity but is in the works. A full representation of each paper and their techniques is presented in Fig. 3.1.

3.8.1 Analysis and Summary of Proposed Tools

So far this article has: 1) presented an overview of deployment strategies for IDSs in the IoT, 2) discussed that a constrained network may take many forms and locations of deployment, 3) presented techniques for the training of an IDS and detection of attacks, and 4) presented an overview attacks that are specifically pertinent to these scenarios. We use these four

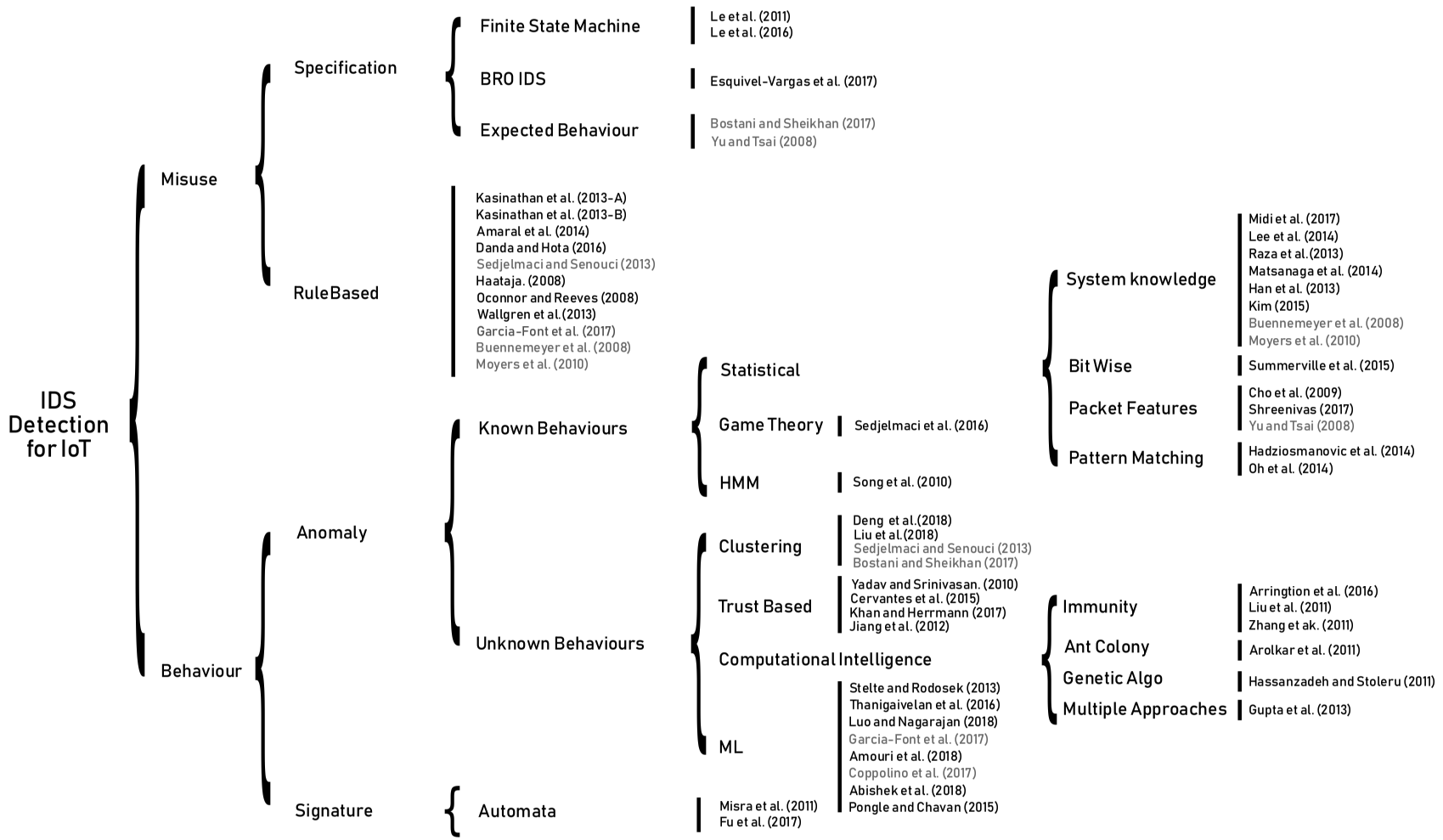


Fig. 3.1 Full representation of the techniques used by different articles. Grey references denote hybrid approaches and the article is present in each of the hybrid techniques.

core points of interest as the basis of our summary of the literature. For our review we have surveyed 51 articles from the year 2008 to 2018, specifically looking for articles that propose intrusion detection mechanisms for IoT deployments. Each collected article is broken down into how the IDS being presented achieved these four core aspects and the results are presented by means of heatmaps in Tables 3.1- 3.4. Heatmaps can be interpreted as follows: the tables are split into four columns, the first columns contains list of criteria evaluated, the subsequent three columns contain all the papers using that criteria divided by Network IDSs (col 2) Host IDSs (col 3) and Collaborative IDSs (col 4).

Breakdown of Techniques used

The surveyed papers used twelve unique detection techniques, across the different deployment types as show in in Table 3.1.

	NIDS	HIDS	CIDS
Rule based	[55, 75, 70]		[91, 90]
Signature	[154]		[8]
Anomaly	**A**	[165, 107]	**B**
Statistical	[45, 148, 117]	[93, 162, 136]	[38, 128]
Stateful	[68]		
Clustering	[56, 112, 86]		
CI	[21]	[111]	**C**
Specification	[105, 62]		[187, 104]
Trust	[43, 92, 186]		
Automata	[125, 68]		
Game theory	[155]		
Misuse	[135]		[49]

Citations	**A** - [163, 76, 171, 116, 179, 70, 77] **B** - [157, 144, 5, 9, 187] **C** - [74, 79, 20, 191]

key 1 2 3 4 5 6 7

Table 3.1 Techniques Used to construct Intrusion Detection Systems in IoT

The technique employed the most was by far anomaly based detection. This is aligned with our presented hypothesis regarding it being difficult to determine rules or other strict specifications for IoT deployments. This statistic is slightly skewed by the fact that unlike some other techniques, anomaly based detection might encompass a wide variety of ap-

proaches. Another interesting thing to note, is that whilst anomaly is by far the most popular technique no work using this technique has been implemented on live environments, results mostly focusing on simulations and detection accuracy as the core metric. We hypothesise that this might be an outcome of the machine learning community being the main driving factor behind these IDSs, a field in which accuracy is often used as the main metric of interest. Similarly to anomaly detection statistical techniques have also shown to be quite popular. These technique whilst similar to anomaly detection, relies more domain specific knowledge and is therefore more suited to simpler systems, it is widely used in WSN systems. Perhaps unexpectedly, Computational Intelligence (CI), a relatively new, and non traditional method has come in close third. We note that this result is biased by our review being focused on somewhat recent literature with the earliest work from 2008, which unintentionally coincides with the first use of Immunity Based techniques (a CI technique) for intrusion detection [113]. This technique is highly adaptable a strength which favours IoT deployments. Its relative novelty also incites investigation in its feasibility in this area leading to more works being published. Since the technique has been used in many tools since, as recently as 2018, its safe to say its been a good match. Specification, Rule Based and to a lesser extent signature based techniques have come as clear favourites behind the discussed top three. This is aligned with IDSs in more traditional systems and comes at no shock. They are however more popular in simple IoT deployments, as they tend to perform poorly in dynamic environments. Making them less suitable for general IoT usage. One further trend that is the lack of popularity of misuse detection. Whilst misuse detection suffers from several downsides (see Sec. 3.6), it is vastly popular in commercial IDSs for traditional systems (in part due to its actionability), so it being the second least popular technique in academic literature is perhaps unexpected.

The very nature of how these systems are constructed, such as clustered networks, WSNs, and massively distributed systems, leads to the development of more novel techniques. Techniques such as clustering, automatons, trust based and to some extent game theory are naturally more suited to these kinds of infrastructures and have consequently attracted attention from literature. Due to the limited behaviours of IoT sensors clustering based technique have become more prominent especially in network based systems, with three

papers. Another new technique that is made possible largely in part to the fact that these systems operate unsupervised and interact machine to machine only is trust based detection. Similarly taking advantage of these restricted behaviours two articles have proposed the use of Automaton based techniques, drawing from the many advantages of modelling but greatly reducing the complexity of the systems modelled (and therefore reducing state based issues).

Breakdown of Attacks Detected

In Table 3.2, we see that papers are catered to detect 14 unique attacks.

	NIDS	HIDS	CIDS
Scanning	[135]		[79]
Web Exploits			[79]
Routing Attacks	**A**		**B**
Rank			
Information Theft	[135]		[191, 20, 74]
MITM	[68, 62, 135]	93	[74, 191, 20]
Replay	[68, 62]	93	[191, 74]
Spoofing	[68, 62]	93	[191, 74]
Message Drop	[68, 62]		[191, 74, 5]
DoS (resources)	[155, 77, 75]	[107]	**C**
DDoS	[125, 45]	[136]	
Worm		[165]	
Injection		[165]	
Anomalous Behaviour	[171, 116, 21]		[157]
Cracking cred/passwd	[186]		
KillerBee	[163]		

Citations	**A** - [56, 154, 179, 187, 148, 117, 43, 70, 92, 86, 105] **B** - [104, 34, 20, 49, 187, 144] **C** - [91, 90, 121, 37, 128, 191, 74]

key 1 2 3 4 5 6 7 8 9 10 11

Table 3.2 Attacks Detected by IoT IDSs

We also observe that 20% of the literature focuses solely on the detection of routing attacks. These attacks (including: blackhole, selective forwarding attacks, sinkhole attacks, Sybil attacks, wormhole attacks etc.) have risen in popularity due to their effectiveness against WSN networks and are a huge threat to the IoT. As interconnectedness becomes more the norm, these attacks disrupt the ability for a system to achieve its goal greatly damaging the ability for it to function. Another prominent threat across any internet infrastructure

is DoS and DDoS attacks, this is even more so exacerbated as devices suffer from lack of resources and security measures to mitigate them. Similarly, the combination of protocol related attacks such as MITM, Replay, Spoofing, Message Drop and Information Theft is a hot topic for many papers. This is in contrast with the techniques in use as there is relatively little focus on techniques specific to these threats, such as stateful detection. The general focus on research in the behaviour of these systems seems to instead have left secure protocol techniques behind, making attacks on these a huge issue. Another rising trend is the detection of attacks bespoke to the specific system, such as IDSs specific for KillerBee a Zigbee vulnerability suite, and rank attacks which are specific to hierarchical RPL networks.

Breakdown of Deployment Scenarios Covered

In Table 3.3, we see IDSs are designed for 15 unique deployment scenarios. As the definition

	NIDS	HIDS	CIDS
WSN	**A**	[162]	**B**
IPv6	[148]		[8]
6LowPan	[45, 43]	[107]	[91, 90]
IoT	**C**	[136, 165, 111]	**D**
RPL	**E**		[104, 157, 144]
SmartGrid			[191]
Relay Comm		[93]	
Smart Home	[21]		
ZigBee	[163]		
ICS	[76]		
Bluetooth	[75, 135]		
Smart City	[70]		
Clustered	[86]		
BACNet	[62]		
Healthcare	[92]		

Citations	**A** - [30, 56, 154, 186, 116, 77] **B** - [34, 74, 20, 49, 187] **C** - [111, 155, 125, 55, 68] **D** - [9, 79, 121, 5, 38, 128] **E**- [105, 171, 179, 117]

key 1 2 3 4 5 6

Table 3.3 Locations for Deployment of IDSs

of IoT is quite general it is very difficult to design an IDS solution that can bw suitable to its

full scope. However the majority of IDSs make this claim with 14 different papers designing general IoT IDS solutions. Closely following these numbers are WSNs with 13 solutions, WSNs are much less abstract than general IoT systems, however they share a lot of constrains around computation power and connectivity. It is however much easier to reason about these systems as they are homogeneous allowing for much easier behaviour analysis.

A raising area of focus is RPL, a protocol designed for constrained devices which has risen in popularity. This raise in popularity is in part due to the large number of new attacks on these systems, making IDS research for these very appealing and necessary. Similarly relatively new technologies IPv6 and s6LowPan (a IPv6 subset for constrained devices) are focused by 7 papers. What is exciting is the sheer range of different systems covered by these tools. IDSs have been proposed for a variety of technologies including emerging paradigms such as Smart Cities, Smart Grid and even Healthcare. This shows trends to design security for systems that may not even be a reality yet, and most definitely are not mainstream. As the IoT will move to encompass large swathes of our lives its positive to see some security solutions are already being thought out for these specific circumstances.

Methods for Evaluation of IDS

In Table 3.4, we show the breakdown of evaluations used.

	NIDS	HIDS	CIDS
None	**A**		**B**
Mathematical	[113, 76, 77]	[136, 113, 93]	
Simulation	**C**	[107, 162]	**D**
Trace	[56, 76, 116, 62]	[136]	[49, 191]
Execution	[135, 125, 62]	[165]	**E**
Usability			[38]

Citations	**A** - [30, 55, 105, 75, 171, 179, 68, 86] **B** - [74, 8, 20, 187, 128] **D** - [9, 79, 104, 157, 34, 5, 144, 191] **C** - [21, 45, 56, 163, 154, 155, 186, 148, 117, 43, 69] [92, 77, 179] **E** - [91, 90, 79, 34, 38, 121]

key 2 4 6 8 10 12 13

Table 3.4 Evaluation Performed on IoT IDSs

Out of the 51 surveyed tools, 12 authors didn't evaluate their IDS in any way. This was perhaps unexpected as when a paper is presenting a novel approach to detect attacks a lack of evaluation makes it hard to assess its effectiveness. Although several different deployment areas are explored several authors made attempts to deploy live IDSs for evaluation, which is particularly powerful. Whilst this technique can give the most insight in evaluation, it makes it very difficult to compare as it is unlikely other researchers will have a access to the same setup. A more reusable approach is the use of trace evaluation which relies on datasets from real systems. A dataset can be shared and is therefore useful to compare different IDSs performances. It is worth noting that the IDSs may not always be comparable as some are very specific to some subsystems and protocols and therefore effective comparisons are unlikely. The most common way by far to evaluate was the use of simulation tools, this comes as no surprise as the maturity of these tools has significantly escalated with several options available, either more general or system specific e.g. WSN [123]. Simulation tools have the advantage of accessibility over real systems, and allow for comparative quick deployment.

A total of 24 tools were evaluated using simulators, a further breakdown of which tools are used by the different papers in Table 3.5.

	NIDS	HIDS	CIDS
Contiki/Cooja	**A**		**B**
Matlab	[56, 92, 191]	[162]	**C**
R	[69]		
Avrora	[163]		
Tossim	[154, 155]		
OpenSim	[21]		
Omnet++	[186]		
NS2	[77]		
Qualnet		[107]	
Not Specified	[45]		

Citations	**A** - [179, 148, 117, 43]
	B -[9, 104, 157, 144]
	C -[9, 79, 34, 5]

key 1 2 3 4 5

Table 3.5 Summary of simulators used to evaluate IDSs

The most common simulator used was Contiki/Cooja, a popular IoT simulation tool that focuses on communication and allows for design of various different setups. Perhaps unsurprisingly this is closely followed by Matlab, a general purpose programming language very popular in various engineering fields for simulation. Various other tools are used, with specific focus on Network Simulators, this is due to the IDSs being assessed being NIDS. Interestingly it seems that other types of IDSs have many less tools available. Contiki is used across the different approaches as it is a multi purpose simulator for IoT, more analysis would have to be done to see if the same evaluation could be performed on other tools or if Cooja is the only option. Likewise the fact most of these tools are network layer tools (although not all), means that several characteristics of the devices and the impact of the IDS and attacks cannot be evaluated.

3.8.2 Collecting Tools

To evaluate the efficacy of our testbed but more interestingly to gather a comprehensive evaluation of the capabilities of IDSs for IoT we firstly needed to gather implementations for the tools. To do so we firstly classified the tools into four categories, i) tools which had code available online, ii) tools which didn't have code online, however discussed implementations and therefore should have code, iii) tools which had implemented in simulations or theoretically analysed, as perhaps they has since prepared an implementation, and, iv) tools which had no evaluation nor discussed implementation details.

Secondly we needed to get a updated list of contact details for all the authors, as some of the papers were quite old and due to the nature of academic positions moving around a lot, many of the emails on the papers were outdated. To get up to date emails we first looked up to see if the author had moved to a new institution and had new contact details, this allowed us to find several new emails. In the case of the authors leaving academia as was often the case for PhD and masters students, we looked for Linkedin profiles for emails, even though this was not always possible. Whilst it was often the case for professors or senior members of staff to have remained in the same place, our intuition was that it was worth contacting the full author list as it is often the case that the implementation is performed by the less senior

members of staff and therefore it is more likely to get access from them. For some authors we were unable to identify their new email or whether the current email was still valid. In total 136, email addressed were collected, to the best of our knowledge with the most up to date details.

The third step was contacting the authors for their tools. We constructed an email template, customisable for each tool paper, as can be seen in Appendix B and Appendix C. The email has seven components, 1) a custom greeting to the specific authors addressing them by their title, 2) a short introduction about who I am as well as all my contact details, 3) an introduction to the work I am working on, 3) reasons for contacting them with specific reference to their paper, 4) a personalised comment on their work, manually generated, 5) request for the tool code, 6) a query asking how authors would like to be cited in case of publications and if any licenses applied to their tools and 7) a closing statement thanking them and letting them know that if they may have any question they may feel free to contact me.

Emails were sent out, in two stages, the first stage was sent out to any author in category ii of classification, except for tools that were not suitable to test in our testbed, an example of this was tools that looked at physical intruders in a home. A second stage was sent out to authors in category iii and iv of classification whose proposed solutions may reasonably be implemented (and may have been since), this included approaches who implemented their evaluation through a network simulator. A total of 84 authors were contacted via email in the two stages.

The resulting collection totalled to, three tools in category iii, whose code was available on GitHub, of which all three were based on CONTIKI simulator, and non suitable to our testing environment. One tool pertaining to category ii, which responded to my emails and provided a code implementation (not publicly available). A single tool in category i whose implementation was available online. And no responses in other categories. The net collection resulted in 4/51 papers with code shared publicly, 1/84 authors who responded to a request for implementation, and 46/51 papers with no code accounted for. The net total of tools runnable on our proposed execution environment would be 1/51, 3/51 could be run

in a network simulator and the single email response was of a simulation run in Java which constituted a sub-part of their discussed evaluation. We do not claim that our inability to find these tools means they were unavailable but merely that following our protocol, these were the ones available to us. We therefore relegate the initial goal of evaluating the available tools under the same environment to a future work.

3.9 Survey Thoughts and Discussion

In this work we have extensively analysed and summarised the current state of the art in intrusion detection in the IoT. We have reviewed and summarised 51 proposed tools and technologies and characterised them. Our analysis has shown several new insights into how research in this field is developing, and potential gaps for future works. We have identified a major gap for the reproducibility of the tools evaluation and also discussed shortcomings in the current evaluation process, namely 1) single focus on detection accuracy; 2) lack of unified evaluation methodology; and 3) disregard for evaluation regarding usability of these tools. Our findings indicate that perhaps the very large numbers of papers in this area is in part due to the inability to build upon the software and the lack of research into tools that can be monitored and supervisable. We see a need for a shift to focus away from development of new techniques to a focus on making these tools that could scale and be usable across IoT deployments. The analysis shows that this is in part due to *how* the tools have been evaluated. We see a thorough comparative evaluation by these tools through this evaluation a beneficial and necessary task, we propose a methodology to do so in Sec. 3.10, although we reserve the task of doing so for future work as there currently is not enough access to the proposed tools.

3.10 A Methodology for the Unified Evaluation of IoT IDSs

The review we have conducted has shown there is a dire need for better evaluation of these systems. Whilst several of these approaches make use of simulation, which one would presume provides more ease in comparing results, there is still a lack of disclosure of both

datasets and the tools themselves. We observe that this issue is not unique to the area of IoT, as even on traditional IDS papers, there is a lack of available datasets.

The metrics of evaluation are something that we choose as a core feature of our approach. Whilst accuracy is very important for any prediction system, perhaps not as much as one would think. The ability for an IDS to perform, run, and be analysed is just as if not more important. Consequently, simulation might not be enough to realistically evaluate the system. There are of course limitations in terms of constraints of access to realistic test beds for implementation of the system. Our proposal therefore meets a middle ground able to evaluate systems consistently, with relative ease of implementation and accessibility. However to fully assess the IDS techniques as per the criteria highlighted in Etalle, 2019 [64] and discussed in Sec. 3.7.1 we require means to assess the systems under the same environment. To do so we propose the usage of a virtualised testbed, which we discuss in Sec 3.10.1.

In this section we investigate what core characteristics of IoT systems are essential to the effective design of a security testbed and propose a solution to easily test the security of various IoT paradigms. We present a design methodology for an IoT Security testbed and present means to test different IDS setups under a unified environment.

3.10.1 Building a IoT Testbed

IoT systems often operate without supervision and are formed of various types of devices, so it is particularly important to be able to test and evaluate the systems prior to deployment to ensure smooth functioning. One of the challenges in ensuring the security of the IoT is that we do not yet have a well-established means to test these systems. As they are often heterogeneous and very specific, no single use case will be easily able to cover the full spectrum of scenarios. The many options for different configurations and few means to evaluate the best suiting solutions call for a need to easily test the available scenarios.

One of the core driving factors of the IoT, is the sheer number of computation devices. Devices in IoT systems no longer take the form of multi computation workstations like in traditional systems as these have been replaced by constrained functionality multiple device systems that are interconnected and distributed. This has lead to a steep increase in

components of the system, but also to a wide variety of configuration options. Due to these limitations it becomes much more difficult to setup a hardware based testbed and alternative solutions should be investigated.

The IoT contains many complex architectural components, types of operating systems or lack thereof, network protocols, integration to cloud providers, applications etc. Whilst tools exist to test certain sub-components, the security of the subsystem or part of the architecture does not imply the security of the IoT system as a whole. Existing techniques for testing systems are able to capture certain complexities of these systems, for example tools such at NETSIM [147] and more recently IoTSim [189], are excellent ways to see how communication data passes through a system and in the case of IoTSim even look at application layer interactions. These may be used to test different network configurations, however, they do not capture how the device itself reacts to the communication, they do not capture latency and computing power of the devices and more importantly they cannot be used to test an implementation of a security solution. Another note is that most of these evaluations are based on a *trace evaluation* [122], this means that the evaluation is non live and done on datasets of existing traces or network data, thus making many forms of evaluation (such as system overhead) infeasible, or purely an approximation.

Some of the reason that work in the area of IDS has mostly focused on proposing algorithms to detect patterns in network simulators (discussed in Sec. 3.6), is that they focused on accuracy as the metric of evaluation. However this single analysis is not able to capture several key metrics of IDS performance (see Sec. 3.7). In our review we found that in fact, accuracy is not the only useful metric to evaluate an IDS and that evaluations done only on that basis are often biased to the underlying system [64], some other metrics, such as actionability, on the other hand, are system independent and of much more use to a real world application. Despite these excellent and mature simulation tools, we are still not able to capture core characteristics that would help in truly assessing the effectiveness of the security solutions.

What is desirable to successfully evaluate these systems is a testbed that is able to be dynamic and adaptable and that can be quickly deployed, similarly to network simulators,

but that is also able to capture unique device reactions like in a hardware implementations. As a middle ground, we propose the solution of a virtualised software testbed of IoT devices. Virtualisation has several key advantages: 1) It is quick to deploy; 2) highly flexible in terms of: data collection, configurations (as they are fully customisable) and the types of sensors (as clones of various devices can be quickly deployed) and 3) monitorability, using virtual components we have easy access to several performance metrics and data about the devices that would be much harder to collect in real world systems and provides huge benefits in terms of IDS explainability and creating richer datasets.

3.10.2 Proposed Testbed Structure

The aim of our system is to quickly and systematically deploy custom made IoT device images, in specified network configurations, in conjunction with attacker machines (with attacks of choice), and to evaluate an IDS within this environment. To this end we propose the use of a hypervisor. A hypervisor is a tool (running on bare metal or in software) that creates and runs virtual machines. A hypervisor acts as the central deployment point from which virtual machines can be created and configured. Through use of a hypervisor several network configurations can be specified to mimic various deployment scenarios. For use in security scenarios it also has the advantage that the sub-networks can be isolated and self sufficient, to run experiments in a safe environment. Whilst some works have looked at the use of virtual testbeds [156, 103], with some even looking into its uses for IoT [156], very little work has been done in the context of virtual testbeds for IoT security experimentation, especially for general purpose scenarios. A diagrammatic representation of the proposed methodology is presented in Fig. 3.2. The testbed consists of three levels: 1) the hypervisor that monitors and deploys the systems, 2) the virtual network where the devices are deployed, and 3) the connectivity layer where we can manage routing and data handling for the systems.

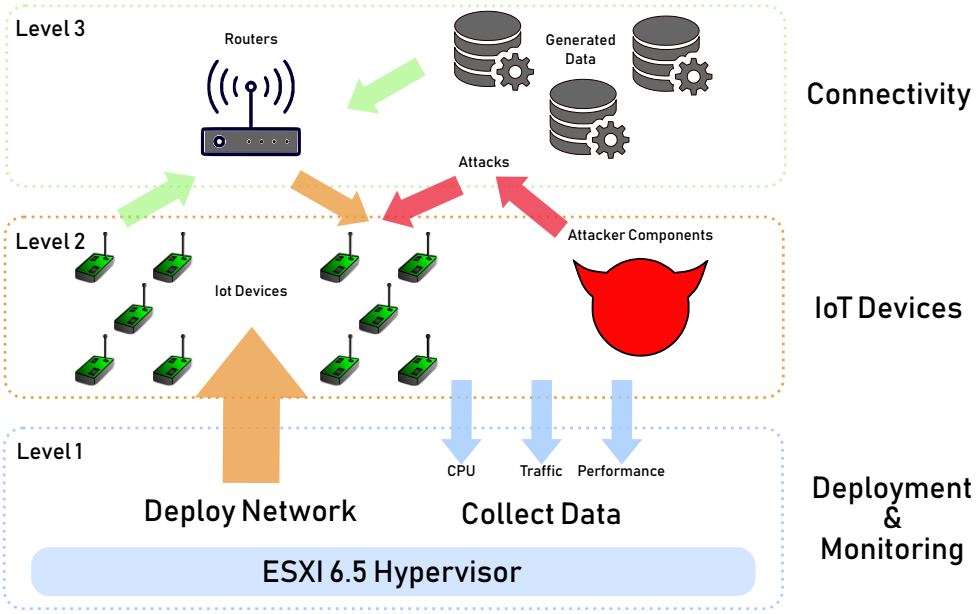


Fig. 3.2 Diagram showcasing the testbed setup. Layer one comprises the hypervisor, network structure and performance data. Layer two is comprised of the devices present in the network. Finally Layer three manages connectivity, routing and data flow

Level 1 - Deployment and Monitoring

One of the many advantages of working in a virtualised setup is the granularity and ease with which it is possible to collect data. Our specific setup runs on top of VMWare ESXI 6.0 [9]. ESXI allows for 4 different layers of data collection for all devices in the network, the first two layers are used for system information and are relevant to our analysis whilst layer 3 and 4 are mainly used for debugging. The first two layers are for: long term usage (Level 1) and short term usage (Level 2) they respectively cover: Layer 1 - performance evaluation including - CPU usage (usage in mhz, cpu entitlement and total megahertz used), Disk (capacity, total latency, provisioned, shared, usage on average and total used), Memory (consumed, overhead, swap in rate, swap out rate, swap usage, total mb, average usage), Network Usage, System heartbeat, Virtual Machine Operations; Layer 2 - all included in Level 1 and additionally includes: CPU idle time and reserved capacity, Disk information about reading and writing, all memory information excluding rollup values, and all information on virtual machine

[9] Available at: "ESXi: Bare Metal Hypervisor." VMware, 12 Apr. 2020, www.vmware.com/uk/products/esxi-and-esx.html.

metrics within the network. For specific insight on device and network wide performance impact of an IDS, Level 1 is often sufficient and can be run continuously in our setup. Further metrics of accuracy are performed uniformly across different datasets and testbed configurations to analyse consistency as well as prediction power. These extra information not normally available in traditional network datasets allow for much more precise evaluation of the system. We can assess decision making of IDS components to increase explain-ability and provide a richer feature space for training.

Level 2 - IoT and Attacker Devices

To allow for quick and easy deployment we create a set number of device archetypes. The archetypes represent several different IoT devices, running different functionalities. Each of the archetypes has different unique characteristics, namely: computing power, protocols for communication, and functionality. Likewise there are attacker archetypes to mimic various intruders, these attackers are Kali Linux based machines running different exploits pertinent to IoT systems, and/or DoS attacks (with the intent to overwhelm the benign devices). We also construct IDS archetypes, this was a much more complex setup as the different architectures of the IDSs required different setups and a single machine type was not possible. Depending on the architecture of an IDS several different setup components had to be initiated. The final component is the environment, this is used to mimic the varied scenarios and environments of the IoT we create a software defined environment able to capture specifics of IoT deployments. Through use of the environment we are also able to test routing attacks.

For this purpose we provide a preset of “IoT” devices one can choose from. To create these devices we make use of TinyCore, a bare-bones 8MB Linux distribution, many more options are available this is just, in our opinion, the most lightweight and flexible option. Our constructed machines are customised to have different applications and communication stacks, and are built to mimic low computational power sensors. This has the advantage that hundreds of machines can be hosted simultaneously, the machines can be quickly built and deployed and they are not bloated by needless extra applications that may interfere with

the other functionalities. A limitation of this infrastructure is that we are not able to setup firmware based devices which are common in the IoT. To deploy the IoT system, the selected IoT archetypes are cloned automatically and put online, they each are assigned static IPs and the routing paths are configured to mimic the chosen network configuration. The static IP allows for allows for routing configuration, identification of the device, and less volatility. This is all setup in a safe virtual network isolated through virtual switches to avoid security threats.

Level 3 - Connectivity and Environment

The third components is the generation of the environment. Our testbed allows for flexible configuration resembling the same characteristics we use in previous modelling approaches [17, 18]. Firstly, a network configuration is specified, this may take the form of a traditional interconnected system, or resemble disconnected networks such a mesh, which can often be found in WSNs. To create these configurations routing paths are set up, to mimic the inability to reach components and specific communication paths. Central routing machines are used for this purpose, acting as part of the environment. Subsequent to selection of network structure is device selection. One of the core differences in the IoT, is that the devices can be deployed in a myriad of different environments, to simulate this we propose an environment to mimic several of these characteristics. We focus on Disconnected Intermittent and Lossy (DIL) environments as this is a unique circumstance of interest from a security perspective. The environments are controlled by the routing devices which have control of all the communications. The messages are then lost, re-routed and communication channels are closed to mimic these circumstances. The *badness* of the environment can be customised for flexible and adaptive security testing. Another components that is handled by the environment is data readings. Dummy data is fed to the sensors so that correct functioning can take place, e.g. taking temperature readings, through JSON databases once again held in the environment devices. The fourth component is the attacker devices, for the attackers we provide Kali Linux Archetypes with a variety of exploits. These can be configured to automatically start their specific attacks upon deployment. The attack patters may be random,

selecting random components or targeted if configured by the user. We note that in this environment we assume perfect attacker knowledge of the devices, knowing which protocols are in use and access to all the devices (ignoring routing restrictions that the devices face). This is because we cannot assume the attacker is itself an IoT component but rather may resemble a powerful hacker, or group of.

Finally a NIDS component can be deployed, the desired NIDS machine is pre-configured to listen to traffic across the network. The routing machines will route messages through it so that it may guard the whole network automatically. The network traffic in the VLAN is switched to promiscuous. There are some limitations with this approach, we are unable to test CIDS, as the tiny IoT devices cannot yet act as IDS agents and the same goes for HIDS. Although future extensions envision the deployment of multiple IDS machines with collaborative components and proximity based monitoring for testing of CIDS.

Our proposed design is but one of the available virtualisation options. An alternative using docker could allow for a similar setup to be run from ones own personal computer. Although the docker approach may be even more lightweight it has less granularity of monitoring than a hypervisors. Although with some supporting infrastructure a lot of the deployment of devices can be automated through scrips at all levels, the initial setup of the virtualised approach has a lot more overhead than the usage of simulations tools and consequently it may be difficult to compare to other works and to make the work fully reproducible. We envision that for future work we can create a framework to allow for the easy deployment of this setup that can be used by other practitioners to easily setup a system in a reproducible manner, although this is currently out of scope.

3.11 Chapter Conclusion

Depending on the architecture of an IDS several different setup components had to be initiated. We make the following observation about IDSs in the IoT:

1. Despite many IDSs proposing multi purpose *IoT* solutions (14/51), these proposed tools were often not evaluated and didn't back up this claim of coverage.

2. Some of the most effective IDSs with the most complete evaluation were those designed for very specific deployment scenarios and attack threats. Leading us to believe that perhaps it is enough the extend current approaches to cover these rather than redefining the field.

3. Although each of these papers had high degrees of accuracy they were not convincing enough to stop authors from designing new systems. We discuss that the main factor behind this is that accuracy is by no means the only metric of interest and a new evaluation methodology needs to be in place.

4. Out of the 51 surveyed tools almost no tools shared their code, implementation details or datasets. This makes it impossible to compare the tools or build upon existing literature.

What we see is a field rich with repetition which seems to have reached very little in terms of consensus of *what* new advances are required in IDSs for IoT systems. What we hoped to achieve was the systematic evaluation and comparison of all the proposed literature to find out what the state of the art and current gaps were. But the reliance on different evaluation methods and hesitation for authors to share their code has made this impossible to do via our own testing and evaluation. For this survey we had to instead resort to a secondary analysis and structured review. We foresee that this trend of new tools being proposed will probably continue until these limitations are rectified.

Chapter 4

A Modelling Technique For The Evaluation of IoT System Interactions

4.1 Chapter Summary

This chapter discusses the proposed formal modelling technique for the evaluation of IoT system interactions. Our *gentleman/rude device* approach [16] models device communications as concurrent CTMCs. The focus is on the interactions between IoT devices and their behaviour under attack. Allowing the observation of device behaviours and addressing the means to model flexible and varied IoT scenarios (Chal. 2). This allows the security professional to assess the security of a setup and quantify potential threats. We showcase the flexibility of our approach in two scenarios. Firstly a hypothetical constrained IoT deployment involving battery powered devices, equipped with DoS prevention capabilities in the form of client puzzles [16]. This showcases that our model is able to assess the optimum setup for the system. Secondly in a much larger IoT setup, a model is presented of the interactions between components in a smart grid, including power generation, controller strategies and consumer energy demand [14].

4.2 Chapter Introduction

The IoT is composed by myriads of heterogeneous devices, and consequently, knowing how a system will interact prior to implementation it is non trivial. The aim of this thesis was to use formal models as a basis to quantify and observe these behaviours. The presented work validates the hypothesis that in the IoT the complexity exists within *how* the devices interact. This is supported by the fact that the individual devices behaviours are mostly simplistic, certainly much more so than traditional workstations existing in classic environments. Consequently, the focus was to model how the devices communicated between each other, through network protocols, and the consequent effect of this communication on the devices themselves. The variety of protocols, means that the flexibility to model several different interactions is required. The focus of an attack is either to disrupt a system or devices through a series of device interactions which can be observed in this manner; or to steal/compromise data and in that case it had to be captured through protocol interactions rather than the device interactions. Although out of scope, to still be able to capture the second kinds of attacks a second approach with means to formalise protocols and to formally verify their correctness is discussed in Appendix F.

To devise a modelling approach for an IoT system, the approach needed to encompass several unique traits that are of interest to a security assessment. Based on our review of the literature (part of which is discussed in Chap. 3) we identified the following characteristics as crucial to a security assessment tool:

1. *flexibility/adaptability*, several scenarios are possible with these kinds of system, so the ability to adapt is crucial;

2. *impact on the devices*, as devices are often constrained e.g. battery powered, it must be possible to assess the impact on the device itself;

3. *mitigation strategies*, as a security professional or system administrator, when deciding how to setup a system it is essential to evaluate the effectiveness of different defence mechanisms;

4. *attacks*, this is the most important decision making factor in effective system deployment as identifying bottlenecks and weak points is incredibly useful;

5. *adverse scenarios*, whilst not unique to the IoT something that has become incredibly common in these systems is that the way the devices communicate and how they are design make for several complications e.g. routing issues, reaching devices across a large distance, battery drainage and offline devices; all make for further consideration in system design.

To evaluate the effectiveness of these considerations, a scenario was devised that would include each of these. This approach observed the impact of DoS attacks on a system of interconnected devices, to asses what the optimum mitigation strategy could be [16]. Devices were designed as resource constrained, battery powered, and with disconnected routing. DoS attacks were selected as an attack scenario as they are one of the most common form of attacks as well as highly impactful to system operations. As form of mitigation a form of DoS delay mechanism called client crypto puzzles was selected [88]. As source of IoT constrain we focus on battery power, a finite resource which would be consumed by normal operations and which would also constitute the objective of denial for the attacker. Every possible deployment scenario was then analysed in terms of mitigation strategy and we quantified the effectiveness of the attack to choose an optimum deployment. This highly abstract but powerful scenario represents a situation likely to take place in a constrained IoT network and was able to showcase the effectiveness of the modelling approach, this first case study is presented in part within Sec. 4.4. The aim of this research is to quantify the potential impact of a DoS attack on a multi protocol network within the IoT and to gauge how a potential mitigation method affects performance. The key contributions include; a model of two types of IoT device networks, one with DoS mitigation in place and one without; and Verification and simulation of these networks to investigate trade-off between security and throughput under a DoS attack.

The second scenario involved the application of the same reasoning and similar modelling methodology to quantify the impact of a kind of DoS attack on the smart grid. To do so we approached colleagues who were working on security of smart grids and discussed

the modelling scenario. This led to the development of a second case study, Arnaboldi et al. [14]. The model includes heterogeneous devices, in the form of power plants, which have constrained resources, the intent of the attacker was to break or temporarily deny these devices so that power demand might not be met. Using a variation of the same modelling principles a security assessment to devise optimum mitigation strategies against these attacks was ran. This included the ability to model a realistic daily energy usage of the smart grid of the UK. The initial case study idea was by R. Czekster, our contributions was the design of the models and the security assessment, the work is in part presented in Sec. 4.5. The proposed scenario of the smart grid allowed us to experiment with real-world data to see if the modelled system behaved as expected and captured real interactions. The aim of this work is to evaluate different power plant configurations in terms of operational characteristics such as cooling down and maintenance time as well as type (e.g. nuclear power or solar panels), aiding managers to make better decisions when designing power systems, and potentially mitigating the impact of these attacks by choosing better load controller configurations. Our model shows the trade-offs of mixing different power plant types to withstand Coordinated Load Changing attacks, reducing the hazardous effects they have on the infrastructure.

As certain aspects of related work and introduction overlapped the publications are adjusted to allow the chapter to flow, the full texts can be found at Arnaboldi & Morisset [16] and Arnaboldi et al. [14].

4.3 Related Work

Since their advent, systems security properties have been modelled and verified using a variety of tools including probabilistic model checking. Analysis of DoS mitigation techniques has been widely covered. Tritilanunt et al. [174] used coloured petri net to verify the effectiveness of HiP client puzzles for DoS mitigation. The authors mainly used simulation under different scenarios of possible DoS attacks and proposed techniques to predict DoS attacks in advance. Similarly, Basagiannis et al. [27] also looked at HIP, making use of verification techniques. They used probabilistic model checker PRISM, introducing a probabilistic attacker model

to analyse the effectiveness of HIP and created different attack paths to break the DoS mitigation technique. This work focuses on a single complete exchange between an initiator and respondent, creating a Dolev-Yao-like attacker.

Several papers address modelling IoT, adopting various different approaches. Aziz et al [25] and Santosh et al. [177] have worked on modeling a specific IoT protocol on the transport layer, looking at MQTT and CoAP respectively. Fruth [67] on the other hand, examines various properties of a Wireless Network protocol including connectivity and energy power through PRISM, and evaluates it on a Wireless Sensor Network System. The author evaluates the battery drainage of certain randomised protocols. Throughput vs security is a common research question in network analysis. Abdelhakim et al. [3] present work on this particular topic in the context of wireless sensor networks. Their paper introduces a concept of security routing, optimised with throughput to present optimum routing.

Continuous Time Markov Chains (CTMC) models have been successfully used to simulate attacks on a variety of systems. Baumann et al. [28], make use of CTMCs to model Flooding DoS on a theoretical network. Through their models they were able to show the impact of the attacks on the systems throughput and evaluate its effects. They could also perform security checking for different DoS rates and scenarios. Our proposed modelling methodology uses a similar approach to modelling systems of devices and observes the impacts of the attacks, but rather than focusing on general flooding of messages looks at the specific problem of load balancing and power management. Previous work in this area by Norman et al. [132] used PRISM to observe run-time strategies in order to achieve a trade-off between the performance and power consumption of a system. Our approach extends this to look at the influence of an attacker on these balancing strategies

DoS attacks have long been one of the most common and dangerous threats in many computer networks. Their detection is therefore the first step required to perform an effective response [41]. These attacks become even more dangerous as the IoT spreads across a vast amount of spectra and parts of life. The literature on security concerns highlights similar scenarios of DoS attacks against IoT systems and CPS [109, 150, 40]. Attackers have started to focus on specific vulnerabilities of IoT systems to optimise their attacks. Liang et al. [109],

showed a simple Distributed DoS attack on an IoT scenario, however, they have demonstrated that the result of an attack, if propagated to a CPS, could be massively impactful. In their work, Roman et al. [150] mention key features of how the way these type of systems are setup can cause security concerns. The unique characteristics present in smart infrastructures may render it vulnerable to new avenues of attack such as battery drain and new types of DoS. These new challenges raise concerns for security professionals such as what security vulnerabilities their specific system could be subject to, and what impact it might have.

Load changing attacks were mentioned in Dabrowski et al. [52]. In their work the authors have discussed a simulation concerning the impact that load changing has on power management. Their attack is based on the fact that when operating a power grid, providers have to continuously maintain a balance between supply (i.e., production in power plants) and demand (i.e., power consumption) to maintain the power grid's nominal frequency of 50/60 Hz. Their Matlab simulations show that this balance can easily be broken through a botnet attack. Through their power analysis, they also estimate the number of devices needed to disrupt a country's power grid. This work is one of the first to show that a potential attack can be staged against a power plant without the need for manipulating the controls itself, but just by external device activity. Our proposed approach takes inspiration from this external influences, but scales it to the representation of any power grid. Also, rather than using simulation, our model uses model checking features present in Markovian solvers (PRISM tool) to evaluate the impact these attackers have on the modelled system.

Another work which studied coordinated attacks by botnets and disruptions to the power grid was done by Soltan et al. [160]. Their work introduces the concept of Manipulation on Demand via IoT (MADIoT), through simulations they show how external influences of high power devices can cause disruptions and power outages. They have demonstrated the interdependence between supply of power and the demand, and how this can be exploited to cause disruptions. These types of attack are the core focus of our work, however, Soltan and his colleagues focused on simulations of attacks. In contrast, it is our wish to quantify what these attacks mean from the perspective of a potential supplier. Through model checking

we could investigate how the attacks affect the power system, and we can adapt it to better adjust, prepare, or respond to these attacks.

Specific to the context of coordinated cyber-attacks on smart grids, Moya & Wang [127] have developed correlation indices suitable for identification of these disruptions. Sun et al. [166] have proposed a *Coordinated Cyber Attack Detection System* (CCADS), strongly inspired in IDS concepts as well as its benefits to cyber security efforts to mitigate the effects of such disturbances in smart grids.

4.4 Case Study 1: DoS Attacks and Mitigations in IoT Systems

A DoS attack targets the availability of a device or network [114], with the intent of disrupting system usability. The most common method is referred to as Flooding DoS [124], and may be used as an attempt to deplete the devices' resources including memory, bandwidth and/or battery. A DoS attack against an IoT network has the potential to be significantly more detrimental than one against a standard network. This increased vulnerability is due in part to the low computational power and battery power characteristic of IoT devices [167].

The extant literature has delineated several potential approaches that may be effective in the mitigation of a DoS attack [169]. This case study focuses on one such method, known as "Client Crypto Puzzles"[23], one of the most common mitigation techniques. We evaluate the probability of the system (or subsystem) being denied within a specific time frame in an IoT network. Using our proposed model we are able to assess properties such as: i) At what point is the mitigation technique doing more harm than good? ii) How does denial of a single device impact functionality of the entire system? iii) Does it create a snowball effect?

Client puzzles take many forms, but the general purpose is to force the sender to perform a computationally intensive task prior to authentication, consequently reducing their ability to spam messages [23]. Client puzzles have been adapted in IoT systems [84] and have been shown to successfully decrease effectiveness of a DoS attack. It is however of high computational intensiveness. In the specific context of IoT, an additional consideration is that

a client puzzle (especially one of high complexity) can place strain on the battery, causing high delays in throughput whilst a client is occupied with solving the client puzzles. If the puzzle drains battery at a rate equivalent (or more) than a flooding attack, the increased security may actually harm the system. In this case study we model the trade-off between security and throughput in addition to the impact the increase in computation has on a device's battery life span. It is also the particular case where in the scope of the IoT one device being denied will not harm the system as one device may be performing an inconsequential or very small task. We model this through the connectivity of devices. We can observe the potential snowball effect of a system as the denial of one particular device will increase the probability that other devices are also going down. Through the model we can observe the scenario where DoS mitigation, throughput and decrease in battery are at optimal balance to obtain the best possible result in all three cases. From this, we may model the ideal setup given specific number of devices, connectivity of the devices and DoS strain. In particular we demonstrate that in some cases mitigation techniques can actually increase the likelihood of a DoS, due to drainage of battery.

4.4.1 Model for IoT Devices

To model objective is to observe the impact of an attacker aimed at draining the battery of a set of IoT devices. A flooding DoS attack is modelled to drain the devices effectively (based on measurements taken from the lab). As form of mitigation a form of DoS delay mechanism called client crypto puzzles is selected [88]. As source of IoT constrain we focus on battery power, a finite resource which is be consumed by normal operations and which would also constitute the objective of denial for the attacker, in order to define attacker objectives. The model abstracts an IoT network under DoS strain, and it is implemented as a system of *devices*. A device is a sensor connected to the internet with its own power supply. A key aspect of the IoT is that different devices might have different battery lives and different security features (in this case study, we focus on whether a device is implementing a DoS mitigation technique). Hence, we consider the following device properties: a battery life, a message queue and connectivity (what other devices it can connect to). Battery life is a

measure that is drained whenever a computationally intensive operation takes place such as sending a message and computing a client puzzle. A device can only hold a limited number of messages, and after the queue reaches capacity it cannot receive more until it has replied with acknowledgements. If a recipient device queue is full the initiator waits until it either frees up or timeouts and then re-sends. To model connectivity each device has a set of other devices it can communicate with and receive messages from. To implement the concept of processing time we implemented arbitrary delays when processing client puzzles.

We introduce the concept of "gentlemen devices" and "rude devices". A gentleman device will utilise "politeness", i.e. they will send a message and wait for an acknowledgement (or timeout if acknowledgement takes too long) from the recipient and conclude his current discussion with the device before initiating another message exchange with the same device. It can however simultaneously hold exchanges with other devices. Rude devices on the other hand may continue sending messages to devices within their connectivity before the full communication is over, replicating a flooding attack. The effects of these rude devices flooding spreads as even gentleman devices connected to the flooded device will not be able to commence an exchange if its message queue is full.

The attacker or rude device can have different rates of attack, mimicking different strengths of a DoS attack. We assume it sits outside the network and is not part of the connectivity so it can target any node, but gentleman devices cannot perform an exchange with it. To further simulate the attack strength, at each attack a proportional amount of battery is drained depending on the attack's intensity (rate). We use stochastic probabilities to target random parts of the system as we assume an attacker has no knowledge of the system setup. Due to the connectivity element an attack on one part of the system may exert a higher impact than an attack on another part e.g. If devices B and C only communicate with A, the denial of A stalls the whole system whilst the denial of B still allows the system to function.

Formally, a rude device R is a tuple $R = (S, s_{init}, A, R, L)$, where $S = \{idle, active\}$ is a set of states, $s_{init} = idle$ is the initial state, $A = \{msg, ack\}$ is a set of actions, R is a matrix containing the rates at which any of the actions are performed, e.g., $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ shows there is a rate of 0 to go from *idle* to *idle*, a rate of 1 to go from *idle* to *active*, a rate of 1 to go from

active to *idle* and a rate of 0 to go from *active* to *active*, finally L is an atomic proposition defined as guard→action where the guard must be true in order for the action to take place and take the attacker into the next state.

The behavior of a rude device is defined as follows: if the device is in state *idle*, there is a probability to move to the state *active*; if the device is in state *active*, the attacker chooses a random node in the network and starts flooding it. If that particular part of the network has mitigation technique in place at each message it has to solve the crypto hash before sending again. The attack continues until the device's battery has been drained. The attacker then goes back to idle. The guard is used to make sure the attacker behaves within the scope of the model. The first atomic proposition assures that the attacker does not target multiple parts of the network simultaneously and then switches to active and the second guard follows the steps to fill the message queue and drain the battery.

The other nodes in the network or gentleman devices may either have DoS mitigation techniques or not. A gentleman device G is formally defined as $G = (S, s_{init}, A, R, L)$, where $S = \{idle, sending, receiving\}$ is a set states, $s_{init} = idle$ is the initial state, $A = \{msg, ack, challenge\}$ is a set of actions, R is a matrix containing the rates at which any of the actions are performed, e.g., $\left(\begin{smallmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right)$ shows there is a rate of 0 to go from *idle* to *idle*, a rate of 1 to go from *idle* to *sending* as well as a rate of 1 to go to *receiving*, a rate of 1 to go from *receiving* to *idle* and a rate of 1 to go from *sending* to *idle*, and L is an atomic proposition defined as guard→action, there are guards to enable the correct behavior of message exchange (e.g. *idle*, A *to* B, B *to* (ACK) A, *idle*).

The behavior of a gentleman device without mitigation technique is as follows: when *idle*, G is active and has a chance to begin a communication between any of his connected devices. From idle it can transition to any of two other states *sending* and *receiving*. If *sending*, G sends a message to a connected device and the battery is drained, it then initiates a timer, if the acknowledgement is not received before the timer runs out the device goes back to *idle* however the reset drains the battery, if it is received it finishes the exchange and also resets to *idle*. If *receiving*, the message is added to the queue and the initiator is noted as to direct the ACKs to the right initiator (multiple messages may be received at the same time).

The acknowledgement is then sent and the device is reset to *idle* and the queue is decreased. To implement a device with mitigation techniques, we add the following properties to a gentleman device; if in state *receiving* before it can send the acknowledgement to a initiator with client puzzles there is a time delay to portray the time it would take to solve a puzzle. The time delay increases when the size of K increases, as it mimics how a harder puzzle is more difficult to resolve. We refer the reader to [23] for some examples of client puzzles.

Client puzzles have been investigated as DoS mitigation tools in various different network environments [66]. In this modelling approach we wanted to identify its merits and deficits in the specific case of resource constrained IoT. A more difficult puzzle, whilst more effective in delaying an attacker has higher computation impact on the device, similarly to computing hard cryptographic functions [54]. By measuring both the throughput of the system and the well being of the system by battery we can assess both the benefit of the puzzle towards delaying the attacker but also the negative impact that challenging every communication with a puzzle has on the overall wellness of the system By observing each different setup options in the experiments in Sec.4.4.5 we can establish what an ideal configuration might look like.

4.4.2 Experiments

To test our model we ran a variety of experiments with different systems and security setups. One of the key aspects of our experiment was how the impact of the DoS attacks scaled with different attack rates, different setups and how it effected the throughput and security to investigate the viability of these mitigation techniques in the context of the IoT. The ideal scenario is when the probability of being denied within time T is low and the throughput after T seconds is high. Observing these circumstances in finite systems where there is a set number of devices, we looked at all possible setups the system could take in terms of how many devices are protected and by what level of client puzzles, and then tested them under different DoS strains. Using the result we can tell which setup is the best suited to a particular rate of DoS and which setup will maximise throughput and security. We theorised that given the circumstances of the IoT and the relatively high processing times at certain levels of puzzle difficulty it would be the case that the lowering in chance of denial would

not be as significant as the corresponding lowering in throughput caused by the processing delays, this is analysed in the case study provided in Section 6.1.

We made use of statistical model checking when examining the larger models through PRISM's simulation engine. This approach is particularly useful on very large models when normal model checking is infeasible. Essentially, this is achieved by sampling: generating a large number of random paths through the model, evaluating the result of the given properties on each run, and using this information to generate an approximately correct result within a specific Confidence Interval (CI). Let X denote the true result of the query $P=?[...]$ and Y the approximation generated. The confidence interval is $[Y-w,Y+w]$, i.e. w gives the half-width of the interval. The confidence level, which is usually stated as a percentage, is $100(1-Confidence)\%$. This means that the actual value X should fall into the confidence interval $[Y-w,Y+w]100(1-Confidence)\%$ of the time [131]. We tested for the following properties.

Throughput: the number of messages processed over a given time interval (cumulative messages sent/current time). By definition if a message takes longer to send the throughput will decrease, hence adding computationally intensive tasks that delay the transmittance of messages is going to decrease the systems throughput. However if they delay the likelihood of devices being taken down by an attacker the theory is that in the long run it will actually increase the throughput under DoS attacks. To calculate the throughput of the IoT system we make use of PCTL formula $R\{Msg\_sent\}=?[C \Leftarrow T]$ or what is the cumulative total messages sent by the system in time T and then divide the answer by the value T. The value *Msg_sent* is a reward structure that assigns one reward every time a successful message exchange is completed.

Likelihood of System being denied: we defined the denial of a device, when either its battery has been completely drained or its connected devices have been drained and it therefore cannot transfer from the *idle* state. The whole system is down when all devices have been *denied*. This is once again monitored through PCTL over a restricted time, running different variables one can optimise the number of protected devices as well a what strength of protection to optimise the implementation least likely to be denied.

Snowball effect due to denial: of further interest is the ability to recognise the *critical* sectors of a system. We defined a *critical* sector by examining the impact it's denial has on the rest of the system. Highly critical sectors are also the parts of the system which require more securing. We achieve this by measuring the *snowball effect* or rather after the denial of device A what is the new probability of the rest of the system being down. Theoretically highly *critical* devices will increase the probability substantially whilst non *critical* devices will make a small difference.

To observe the best case scenario of a particular setup, we define a setup as a particular spread and strength of mitigation techniques on a given configuration of devices. We observe the balance between security and throughput, the higher the ratio the better the setup for that particular DoS strength. To observe and test the initial hypothesis in Section 6.1 we create a case study on a specific scenario and evaluate the results for each possible implementation given specific assumptions.

4.4.3 Experiment Setup

We have used the model described above in PRISM Model Checker as well as PCTL to perform both quantitative verification and simulations, various properties where checked and the model was tested under different scenarios. For verification the full state space is explored whilst for simulation we use the following setup. **Number of Samples:** 100,000, **CI:** 0.001 and **Maximum Path Length:** 1,000,000. For the purpose of the model we were interested in evaluating two key properties: throughput and probability of denial.

We can also examine which part of the subnetwork has been denied first and specifically the time required for the denial to take place. The setup for each experiment was: **Client Puzzle Difficulties (K size):** 5 to 20, **Time of system (Time Units):** 20 to 200 units, **Devices in Network (All protected):** 5 and **Rude Devices:** 1. This differentiated from the case study where all the variables were tested in all possible setups (with some specifications explained in section 4.4.5) to find the optimal setup.

Results from the initial setup highlight the key factors of our model i) the way client puzzles drain battery ii) the effectiveness of mitigation techniques to avoid denial of service

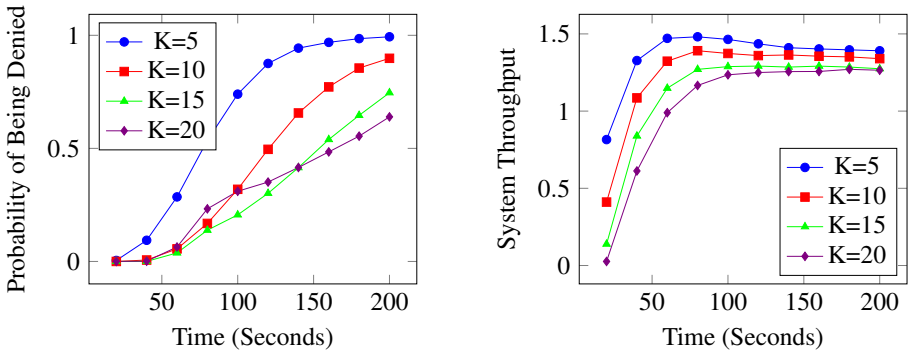


Fig. 4.1 The graphs represents a system being targeted by a DoS attack, the one on the left displays the probability of a DoS attack being successful over time (20s to 200s). The graph on the right represents the throughput of the system over time (20s to 200s). Diagram from Arnaboldi & Morisset [16]

and iii) demonstrating that in finite time it can sometimes be useful to have a less intensive client puzzle as they can disrupt more than help.

4.4.4 Results

We demonstrated the potential strain that client puzzles place on a system and as can be seen in **Fig. 1** at lower times (i.e. before any part of the systems are under DoS or haven't gone down yet), the higher client puzzles create such a strain that they increase the likelihood of going down rather than diminish it.

It can be observed that the harder the puzzle, the lower the throughput, whilst the security is increased. Furthermore, due to the extra drainage in power of the more difficult puzzle, a smaller value of K performs better in earlier times (At time 80s we observe that K=20 has a higher denial probability than K=15 this is due to the extra processing strain and battery drainage of the harder puzzle). As can be observed towards time 200s the throughput comes to a stall as different parts of the system are denied. It is to be noted that as the lower puzzle difficulties are declining in throughput, K=20 is on the rise, as the increase in security allows for some nodes to still output messages.

4.4.5 Evaluation

We apply our model to a specific scenario to demonstrate its application. We assume a potential system engineer views their IoT network as under constant DoS attack. They wishes to know what would be the best way to optimise the balance between throughput of their system and the likelihood of being denied when implementing security mitigation for these DoS attacks. We wished to observe the system for a time period long enough to see impact of the attacks and mitigations in play. We empirically observed that given our setup the initial 100 time steps of the CTMC were sufficient to observe the decrease of performance of the system and most configurations were successfully denied in that given period. The device setup is the following; A is connected to B, B is connected to A, C is connected to A, B and C, and D is connected to C. We also assume a rude device E, which is connected to all devices.

We ran every single scenario of setups (A protected, AB protected, B protected...) and for each scenario tried every single possible value of K. We set the value of Battery to be 50, a maximum message queue of 5, Time at 100 time units, a single rude Device (E) with rate of attack the same as the rate of any gentleman device (ABCD) and the values of K from 5 to 15 tested on every protected device, the results can be observed in **Table 4.1**. If we use our formula of throughput/probability of denial, with the highest value being the most optimal result. We see that the scenario AD provided the best ratio at every different value of K, with scenarios including C being connected (the one with the most connectivity) not performing as well.

The results support our initial hypothesis that mitigation techniques actually increase the likelihood of denial through battery drainage alone. This is evidenced by the scenarios which had C protected that proved to be the most inefficient (throughput dropped significantly). The rude devices have an equal chance of attacking as the gentleman devices. As a consequence, since every single connection to a protected device will drain the battery and since there is equal chance a rude device will start an attack, the strain on the battery caused by normal message sending (and computing of puzzles) is more detrimental. However if we take ACD which theoretically protects the network on all levels and D which is not very connected and

Table 4.1 Every Scenarios For each setup of Protected Devices (PD) and Unprotected Devices (UD), each setup has a single rude device E targeting the other gentlemen devices

		K = 5		K = 10		K = 15	
PD	UD	Prob.	Throughput	Prob.	Throughput	Prob.	Throughput
A	BCD	0.655	0.733353	0.712	0.713375	0.729	0.726894
AB	CD	0.624	0.654857	0.645	0.636341	0.829	0.646891
ABC	D	0.887	0.684555	0.963	0.580316	0.996	0.513519
ACD	B	0.956	0.547798	0.986	0.346246	0.997	0.241135
ADB	C	0.946	0.423408	0.951	0.309469	0.972	0.267136
AC	BD	0.964	0.772681	0.979	0.662404	0.988	0.582111
AD	CB	0.461	0.739893	0.415	0.690475	0.381	0.687251
B	ACD	0.496	0.741337	0.487	0.723239	0.465	0.717923
BC	AD	0.817	0.713618	0.899	0.617848	0.959	0.547188
BCD	A	0.938	0.520275	0.968	0.3279814	0.985	0.228197
BD	AC	0.927	0.454418	0.927	0.362362	0.954	0.313664
C	ABD	0.901	0.843735	0.91	0.72117	0.959	0.663217
CD	AB	0.897	0.555124	0.919	0.38078	0.939	0.279741
D	ABC	0.859	0.610293	0.878	0.536433	0.862	0.535952

test both these two under a longer period of time (200s) we can see the results are altered, as the attackers will continue until a device is denied. On the other hand, if it takes longer to take down a device, the overall system lasts longer and therefore the throughput is higher with the protected devices. We also show that identifying single *critical* devices and protecting them rather than the whole system can be a valuable technique. These results exemplify the potential application and benefit of these techniques.

Our attacker model assumed a rather naive attacker that did not have a priori knowledge of the system. This was captured in the sense that it randomly selected a device to target. This leads to potential situations in which if it targeted the weak device (C) it would be more impactful than selecting a border device (B). In practice this may or may not be a realistic assumption. Even if an attacker initially may have no specific insight of the network he may be able to gain further information through reconnaissance and network scanning. Which could be quite likely, although he would not gain insights about battery drains through these techniques. If we wanted to adapt to match a more specific attacker this could be achieved by giving higher probabilities to target certain devices, or even introducing non determinism

(as described in Chap. 5) to determine the optimum attacker strategy as an outcome of the current mitigation setup.

As can be expected one simple solution to increase both the throughput and the security is to increase device battery or upgrade processing power. An alternative solution may be to replace the *critical* devices with more potent ones. However assuming the device specifications are constant and the only customisation is the strength of the puzzles and which devices to protect, through our tool we are able to identify what produces the best results. The topic of battery adds a further layer to the common question of security vs output and creates several additional layers of complexity. What we have gathered from our analysis and results is in line with our initial hypothesis. Furthermore, this model enables the discovery of *critical* sections in an IoT system which might not be as easy to find compared to our simplified case study.

4.5 Case Study 2 - Load-Changing Attacks and Mitigations in Smart Grids

Cyber security is a major concern when evaluating critical resource infrastructures. Malicious attacks and unintended damages significantly increase each year, and it is therefore important to know what effects they will have. Due to strict supply-demand requirements in power grids, to maintain equilibrium is of paramount importance. In practice, the entities that regulate and those that actually maintain equilibrium depend on the politics of the country, and we do not refer to them specifically, rather we refer to a Cyber-Physical System (CPS) component called Load Controller. Whenever it is required to increase or decrease energy levels, it may trigger costly responses, e.g. turning on new energy sources or disconnecting areas from the grid. Historically, power generation and demand have been very separated; however, this scenario is changing as a consequence of security risks and possible attacks, so the current mitigation strategies may not be applicable any more. The current role of a load controller is balancing energy supply accordingly to demand while maintaining the

frequency of the current at around 50 MHz in Europe (or 60 MHz for other countries such as USA, Brazil and Japan).

Organised attacks aimed at power infrastructures are called Coordinated Load-Changing Attacks (CLCA), where synchronous connections or disconnections of high-wattage units such as water heaters or air conditioning units are used to cause disruptions in energy provision. CLCA are here considered as *black boxes*, i.e. they do not require extensive knowledge as to the particularities of grid operations in order to be employed. If sudden spikes or drops such as synchronised turning on or off of several devices takes place, they cause the equipment to short and break, causing damages and reducing the availability of power supply. The load controller can easily cope with common occurrences in terms of imbalances, adjusting energy flows accordingly; however, sudden usage spikes may unadvisedly cause the grid to collapse. Malicious users could profit from those situations as they could infect a considerable number of high-wattage devices to coordinate actions that impairs energy distribution [52, 53].

In the context of this case study, we are interested in two types of systems: on the one hand, we consider CPS [106, 146], i.e. systems with limited resources (low power, energy, processing or other capacity related issue) employed in a variety of equipment ranging from sensors to smart grid components. On the other hand, we are taking into consideration large scale infrastructures such as the Smart Grid, *Active Buildings*, different types of power plant (that can be solar or nuclear, for example) and other power reliant schemes. We are specifically focusing on modelling their interaction, i.e. how power supply mechanisms react due to CPS usage, as well as the influence one has on the another. Through this model we can observe the daily usage of energy in the smart grid, and calculate its ability to cope with duress. Our modelled attacker (or adversary) attempts to exploit the CPS mechanisms to decrease availability and cause damages; our model captures the probability of success and the tolerance of a theoretical smart grid made of various different types of power plants, against type of attack.

In this application context, it is worth defining a CPS and its roles. It consists of components with two parts, i.e. a computing part integrated with a physical counterpart, both

connecting and communicating with other CPS to achieve common objectives. For example, a CPS could be an embedded system attached to a heartbeat sensor component to collect meaningful health data from patients inside a care unit, or could be smart meters exchanging data on power system energy transmission and distribution. These infrastructures usually encompass a sizeable number of entities in hyper-connected environments, deployed to help users improve productivity, bottleneck assessments and much more. The presence of general purpose CPS in residential, commercial and industrial settings is ubiquitous as many vendors offer solutions that vary from smart home sensing devices to closed circuit televisions.

The simplistic nature of CPS implementations makes them prone to software contamination: for instance, installing malware in high-wattage devices, e.g. air conditioning units or water heaters. Due to this, they are natural recipients for coordinated attacks aimed to disrupt grid infrastructures as a synchronised event may cause over demand or influence voltage and frequency to inadmissible levels. This is particularly unsettling for many reasons as it may impact overall energy costs for customers or, in extreme cases, lead to preventable casualties in healthcare settings.

4.5.1 Threat Model

We are working on the assumption that our designed adversary has gained illicit access to a large number of IoT devices and formed a botnet. Using this botnet, our envisioned attacker targets a smart grid infrastructure through excess energy usages and causes a spike which damages the load controller. When a power plant experiences very high load, it will employ one out of three mitigation controls, primary, secondary or tertiary [52, 160].

1. *Primary Control* distributes the load to other power-plants in the vicinity.

2. *Secondary Control* makes an assessment to return to normal operation if criteria is met.

3. *Tertiary Control* frees up resources from previous Primary and Secondary controls.

The most damaging target is Primary Control, as switching on and turning off further plants is expensive and time intensive. The objective of the attack is to continuously trigger the

Primary control and consequently cause the most damage, including potential damage to turbines and machinery caused by the strain.

Further to this consideration, power suppliers are constantly balancing the frequency of the supply. Sudden spikes may affect the frequency significantly enough to activate security mechanisms of power plants for which they detach themselves from the grid. This may lead to blackouts and disruptions. A smart intruder who has gained access to a botnet can choose when to turn them all on synchronously. If she controls enough devices, she may induce a spike and make suppliers detach. In order to inflict the highest damage, the attacker needs to make sure that she will cause a spike; however depending on current usage, this may or may not happen. If the attacker controls a fraction of the devices and turns them all on, but these devices are already operational due to expected daily usage, the spike may not trigger. Perhaps counter-intuitively, depending on the daily usage, it might be a lot more damaging to trigger the spike at a lower usage time such as mid afternoon to cause the most disruption.

The effectiveness of a spike will also depend on the type of supplier providing the energy. Whilst if a nuclear supplier is spiked it might take a very long time to recover, gas power plant have a much higher adjustment rate and are therefore more resilient to these attacks. In our model we loosely reflect the response behaviours of hydro, gas and nuclear power. These are only loosely based on realistic power plants and only wish to showcase the flexibility of the modelling approach, a similar approach could be used adapting real values in the model. We model the demand borrowing real values[1] in the UK, which we scale down to limit the number of power suppliers in the model. The attacker is modelled at every hour of the day (with the mean power usage at that time), and we calculate the success and impact rate of the attacks.

4.5.2 Problem formalisation

We demonstrate our main idea through the definition of a small order CTMC that will guide our process when we extend it to the set of power plants, load controller and botnets. We

[1]The demand across the UK of the 27 September 2019.

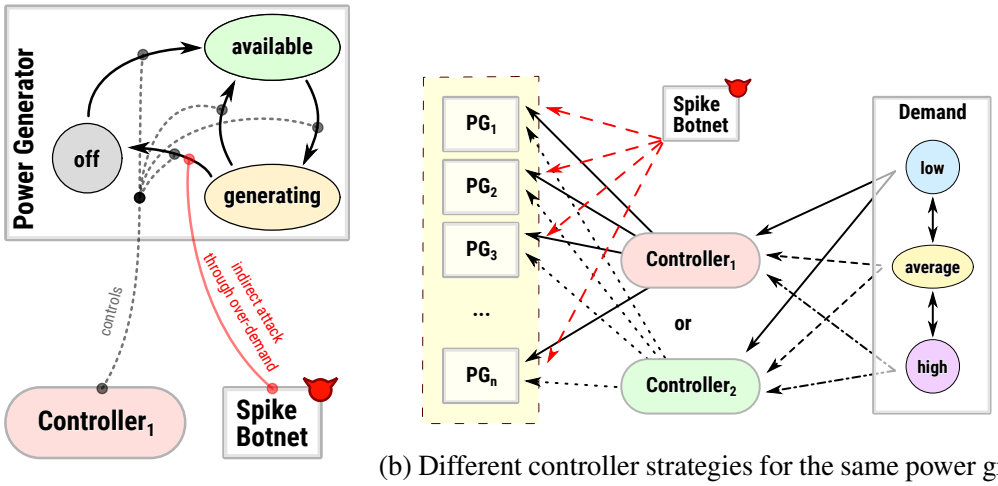
are working here with a very reduced state space, explaining our initial state, some possible transitions and what we actually mean by an attack.

In a simplistic view of our model, we have a single power generator (PG) that can be in three states S_{PG}: (i) available (a), ready to supply but not yet generating energy to the grid, (ii) generating (g), currently providing energy to the grid, or (iii) restart (r), detached from the grid, not generating, nor supplying, and in the need of a restart. The states in S_D model the demand D loading the grid at the average expected level m (for *medium*) plus or minus small deltas, modelled by two additional states l (for low) and h (for high). The attacker B is a botnet controlling a large amount of infected devices. Its states in S_B are modelled as simply 0, when the infected controlled devices are all off, or 1 when those devices are switched on. In summary we have the following states

$$S_{PG} = \{a, g, d\}$$
$$S_D = \{l, m, h\}$$
$$S_B = \{0, 1\}$$

Following the notation introduced in Chap. 2, the CTMC relating to this simplistic model is defined as a tuple (S, S_0, R, L) where $S = S_{PG} \times S_D \times S_B$, and $s_0 = (a, m, 0)$. The rate matrix R is a square matrix of dimension $|S|$ whose entries are zeros apart from those corresponding to the transitions we modelled. So for example, given two states s_i and s_j, then the transition rate r_{ij} is the mean per time unit that we expect the transition from s_i to s_j to happen. The labels in L associate valid proposition to states: we associate them to desirable properties. In particular, we label all states where the current demand in the system is above the current supply as *"overDemand"*, for example $(d, m, 1)$, where if the PG is detached it cannot match the demand m. As the reader may notice, the controller of the power generator does not appear among the states. This is not because there is a single PG, but by modelling choice that needs to refer to a more complex system with multiple PGs to be illustrated.

The basis for our model is presented in Fig.4.2. It is our intention to model a botnet (B) influencing the setting of power generators to let them go off, or detached. In the normal



(a) Attack-defence game between controller and attacker.

(b) Different controller strategies for the same power grid.

Fig. 4.2 The controller and the attacker role in our problem formalisation fine grained to the transitions of a PG (a) and coarse grained to the whole system with multiple PGs (b). We remark that the attack is not directly done to the power generator transition from *generating* to *off*, but the attack indirectly causes it through spike over-demand. Diagram from Arnaboldi et al. [14]

operative state of the system, PG is available, then generating, or disconnected. The demand operates at the medium value m, for some time going to low demand or high demand. A controller dictates the response of the grid in case there is an excess of demand. It is a deterministic transition attempting to create an equilibrium in the grid. In a system without an attacker, two different controllers may behave in the same exact manner, however under an attack the control strategy determines how effectively the load is re-balanced. We define the optimum controller strategy as that one which minimises the time where a system is in a state of over supply or over demand. What dictates the effectiveness of the controller is the responsiveness of the PGs, i.e the controller decides that a PG needs to be turned on to meet the demand, if the PG is a very slow one, this will lead to large amounts of time offline. By modelling different controller strategies, one can easily envision that an optimal controller can be selected against a specific attack and under a specific load. The less trivial research question is whether an optimal strategy can be found to optimise the power supply for a specific grid or CPS. We investigate this problem by modelling the power supply in Sec. 4.5.3, and look at the way three different controller strategies impact the effectiveness

of a spike Botnet attack. We could potentially have a set of PG to work with (Nuclear, Gas, Electricity, Wind, and so on) where the Spike Botnet engages in an attack-defence game to roughly estimate the state of the controller to direct decisions on when to switch devices on or off.

4.5.3 Energy Supply Demand Trade-off Model

Our work focuses on modelling the balance of energy supply and demand. To do so we have created three entities: i) *suppliers*, power generators mimicking different types of plants typically attached to the grid, ii) *consumers*, modelled as average values of energy demand per hour, they also are subject to positive and negative variations across time, and iii) *spike botnets*, a large quantity of compromised IoT devices, they might be thought as high wattage devices such as water heaters or air conditioners that can be synchronously turned on or off, they produce sudden spikes or drops of energy that unbalance the grid and trigger the automatic security disconnection mechanism of the PGs from the grid. One way to mitigate disturbances is to employ *load shedding* or *tie-line tripping*, techniques employed to disregard incoming requests in order to maintain integrity and avoid breakages [53].

The way the demand supply trade-off is modelled is as following: the *consumers* will have a certain energy requirement and the *suppliers* will need to be turned on to meet the demand. However if the Botnet is successful in taking down a PG, there will be a temporary situation of under supply. To meet the demand, more PGs need to supply energy. Their *responsiveness* is subject to several limitations. First, a PG requires some time to be fully functional, especially if it gets suddenly detached from the grid; some are quite fast, while others are expected to be much slower, like nuclear power plants. Second, the amount of PGs is finite; if the botnet takes down enough PGs whilst they are unable to reattach themselves to the grid, there will be no way for the suppliers to meet the demand. Third, independently of how many PGs are taken down, at peak demand, the power grid might have very few spare resources, so if one PG is taken down at 18:00 during dinner time the impact might be a blackout for the whole grid. A toy example showing how the formalism is used on a small case study is attached in Appendix-C.

4.5.4 Power–Energy considerations

In energy terminology, *Unit Commitment* (UC) [137] tries to find the least-cost dispatch of available generation resources to meet electrical loads. In the past, different strategies were conceived to deal with UC related issues such as simulated annealing [158], dynamic programming optimisation [138], particle swarm [173], genetic algorithms [175, 130] or combination of those and other techniques. UC is a relevant problem in energy as the demand/supply requirements are usually uncertain.

We are interested in working here with UC in an abstract way, representing supply and demand for CPS or smart grids. There are several generating resources available for use by energy managers such as nuclear, thermal (using fossil fuels such as coal, natural gas or oil), or other biomass. When deciding which power plant to turn on, several decision variables come into play such as generation level (in Megawatts, MW) and number of generating units that must be turned on. A power plant employs different technologies to generate energy, for instance, nuclear based have to be turned on and then cooled down, tasks that usually take considerable time. Other sources such as solar panels on the other hand have different maintenance peculiarities than those of, for example, wind turbines. Lastly, when broken, each system would involve different resources and equipment, which translates to greater time not producing energy which impacts the grid in its entirety.

Another source of concern is directed towards hourly *fluctuations* (used interchangeably here as either *surges* or *peaks*) that may be present in a day. These differences in demand-supply, if greater or lower than specific thresholds (defined by energy operators) may cause severe damage to the grid and or its turbines (if used), even sometimes causing permanent damages which impact projected energy yield. Our model captures the so called *sudden* surges to the energy grid, where infected high-wattage appliances synchronise their operations to trigger disruptions. Fig. 4.3 shows the daily energy usage for the United Kingdom scaled down to roughly its 1% (e.g. Glasgow's population) with noticeable peaks in demand

according to specific times of a given day[2]. We remark that we scaled down the data to allow for smaller state space in the model.

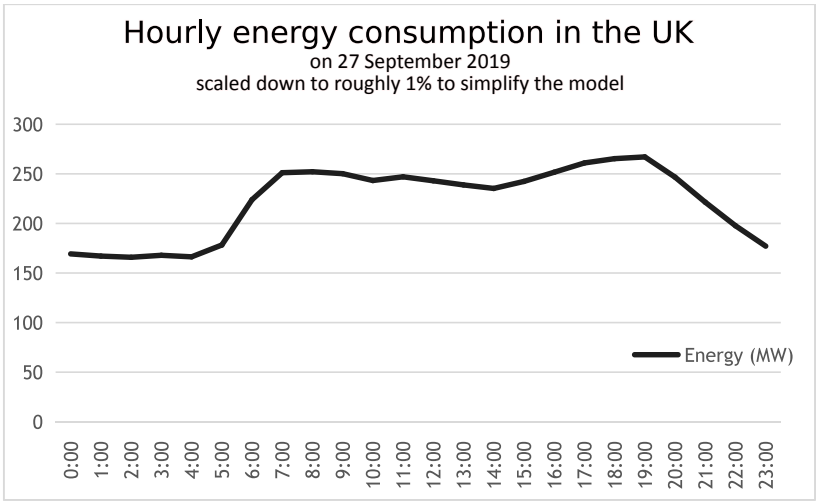


Fig. 4.3 Usage data for the UK, scaled down to about 1%, in MW, across 24 hours on Friday 27 September 2019. Diagram from Arnaboldi et al. [14]

Fig. 4.3 highlights an expected usage trend, with more power demand at times such as breakfast and dinner and a sharp decrease when (the most) people are asleep. These are known trends and therefore power supplies are build to cater to them. It is noticeable that the load is distributed in a way that if there is a reasonably expected raise in consumption at any point along the line the load controller will be able to handle it. This *tolerance* value is what the attacker has to outmatch in order to cause the disruption, perhaps unexpectedly, this means unexpected above normal usage is much more damaging than just mass usage. Through our model we highlight this by showing that the most impactful attack times (varying with the different power plants), may not be the ones of most usage.

In our setting we are dealing with a load-balancing system that tries to maintain the equilibrium to keep the frequency to 50 MHz; this is done balancing the demand (produced by users turning on their devices whichever they are) and the supply (the set of power generators required to meet the demand). We are combining this balance with a control strategy, where we mimic the manager's decisions as to which power generator should be

[2]Original data is available online from the balancing mechanism reporting service in the UK at :https://www.bmreports.com/.

prompted at specific times to cope with demand. These decisions address the fact that under surges or spikes it could be possible to increase the security level if the right measure is taken at the right time, working as a protection against those sorts of disruptions. There are many ways to find reasonable forms of mixing which power to be switched on or off at any moment, and a common strategy is to use greedy algorithms for selection. At the core of these algorithms it resides the fact that a prompt reaction should be deployed as soon as possible, so the next available power plant should be turned on adding power to the infrastructure so it can handle the demand accordingly.

4.5.5 Cyber security model applied to CPS

In previous sections we discussed the problem from a general perspective. For our experiment we focused on a scenario involving somehow realistic implementations of power generators and we adopt the real demand on a specific day in the UK. For the model discussed here, we are interested in using CTMCs due to the required dynamics for our cyber-security problem. Our modules are illustrated in Fig. 4.4: they consist of power generators PG (nuclear N, hydro H, and gas G), the demand D, modelling the supported threshold to try to withstand disruption case demand $\approx$ supply, a controller C, greedily selecting which power generator to use prioritising next according to design decisions, and finally the set of IoT devices I (the Spike Botnet in the figure), symbolising infected high-wattage components under the control of adversaries. These choices were made to highlight a variety of setups mimicking a modern smart grid.

We are considering that each PG has its distinctive design possibilities, usually dealing with high loads of energy in different ways. For our specific case study we are interested in how the different prioritisation of PGs could potentially affect the availability to cope with attacks. In Fig. 4.4, we adopt a tailored notation to represent our modelling choices, for instance, initial states are marked with a *small dot* whereas rates are shown as textual descriptions decorating some transitions (e.g. *slow, fast* and so on).

All PGs start in *available* state when ready to supply, in the *generating* state when supplying to the grid, or in *off* state when offline. We have added a module Controller to

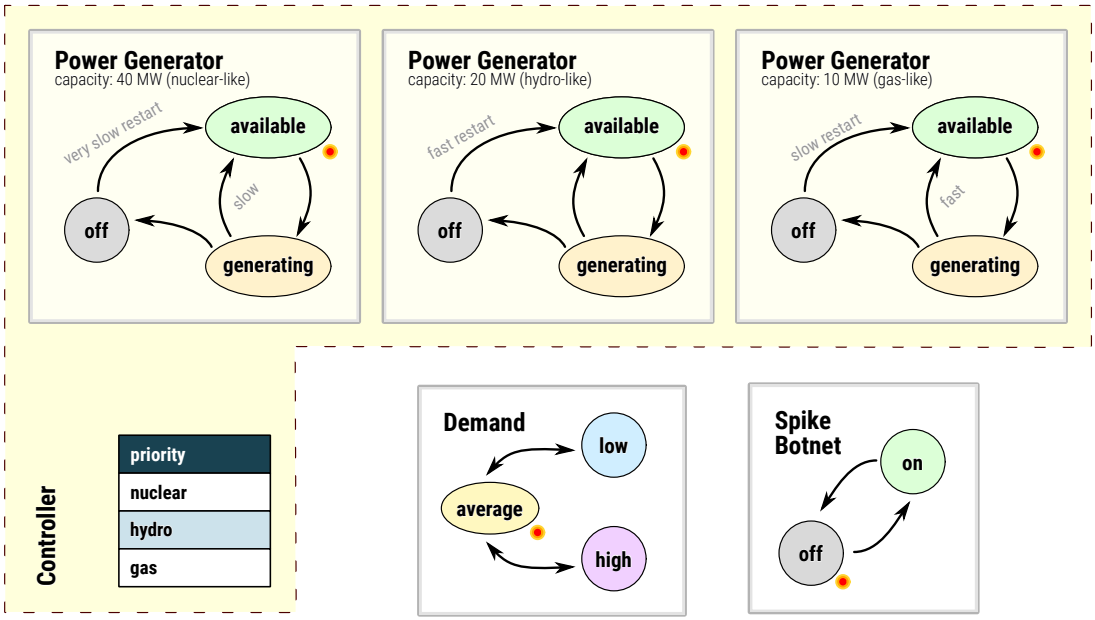


Fig. 4.4 Model and PRISM modules, representing Nuclear, Hydro and Gas power; the demand, along with and variations from the expected value; and our designed attacker controlling a percentage of the systems devices. Diagram from Arnaboldi et al. [14]

cope with the supported variation, parameterised to withstand increments or decrements of $\pm 1\%W$. Finally, the infected devices (botnets) are either *on* or *off*, modelling either an attack in place or inactive, respectively. In our model, the Botnets are composed by unique devices that are only controlled by the Botnet and do not represent daily usage. The state delays (they will be converted to rates due to our CTMC representation) for the model are shown in Table 3.

We changed the PGs characteristics through fixed parameters that map their behaviour. We assume that the power generators' capacity does not depend on the hour of the day, even if it may not be the case, i.e. in night hours we obviously expect solar generator to have reduced supply capacity. This can anyways be modelled, as we run experiments by per hour, and this time can be reduced to be more fine grained. The Table 2 shows the parameters we used, where the number of module instances mimic the specific scenario of matching the power supply for a location populated roughly as much as Glasgow is. We do not realistically refer to Glasgow's power plan scenarios, only to the expected demand of roughly its population. The supply will always cater to the daily demand, however the IoT devices

may still cause spikes to offset the load management and break PGs. It is reasonable to consider that attackers may select regions where the difference between supply and demand are close. In low consumption regions (in Fig. 4.3 they correspond to early morning) or high (start of the day at 8:00 and then at 20:00), the power plants are taking decisions as whether to increase or reduce the power supply to meet the demand. Our modelling approach focus on closely inspecting those time spans, aiming to evaluate which power plant configuration would be best suited to be adopted in case the infrastructure was to be targeted for attacks.

Table 2 Parameters for the model in PRISM and total of instances for every PG.

PRISM Module	**Observation**		**# of instances**
Nuclear (N)	Generates	40MW each	4
Hydro (H)		20MW each	5
Gas (G)		10MW each	6
Controller (C)	$\pm 1\%$ tolerated deviation from normal		–
IoT Devices (I)	up to 30% expected wattage		–
		Total:	320MW

We are considering the following scenarios for our analysis:

1. **NO-ATTACK:** normal grid operation. It is expected to be in *over supply* state most of the time;

2. **ATTACK:** in this scenario we are modelling the coordinated effort to disrupt the grid. The attackers could profit from the peak hour knowledge to direct their efforts. We are varying Controller to prioritise which PG would be triggered first to investigate the possible mitigation strategies as follows:

   (a) **ATTACK-N:** Controller prioritises first on Nuclear, then Hydro, then Gas.

   (b) **ATTACK-H:** Priority on Hydro, then Nuclear, finally Gas.

   (c) **ATTACK-G:** For this one, first turn on Gas, then Nuclear, finally Hydro.

For our four experiments we have instantiated a total of 17 modules, addressing in terms of PG a total of 320 MW. This power corresponds to a grid that could be deployed in a region to serve the power needs of the whole of Glasgow (population $\approx 598,830$ people).

The reachable state space of our NO-ATTACK model is around ≈ 300.000 whereas in the ATTACK-* models, it sums to 57 million (with the modified Controller policy and attack botnets). We computed the model properties related to the probabilities of the system being *over supply*, *in equilibrium* and *over demand*. In terms of rates, we are considering the values of Table 3.

Table 3 Parameters of individual modules used in the model (time scales in minutes (m) or seconds (s).

Module	**State**		**Time**
	From	**To**	
Nuclear	Available	Serving	30s
	Serving	Available	40m
	Serving	Offline	<1s
Gas	Available	Serving	<1s
	Serving	Available	30s
	Serving	Offline	<1s
Hydro	Available	Serving	<1s
	Serving	Available	<1s
	Serving	Offline	<1s
	Offline	Available	20m
Demand	Normal	Negative	5m
	Negative	Normal	1m
	Normal	Positive	5m
	Positive	Normal	1m
IoT Devices	Off	On	1m
Spike Botnet	On	Off	1m

4.5.6 Results

Among the several interesting analysis that could potentially be performed using our model, we direct the focus to the probability for the over demand when a spike is successful. In such a case, we consider an attack successful if the spike makes a PG go offline, after having caused an excess in demand. The controller prioritises the order of activation of PGs (we assume all are under its control). If all preferred controllers are on, an alternative type will be activated.

Our results show key times in which the demand may exceed the current supply, attackers may use those ranges as clear opportunities for disruptions. We have four main scenarios (NO-ATTACK, ATTACK-N, ATTACK-H and ATTACK-G) as described in the previous section, each one comprising one hour of a day, totalling $4 \times 24 = 96$ PRISM models[3]. The attack scenarios resulted in a state space of an average of $57,264,556$ million states, so the experiments were run in parallel on a multi-core server with 16 processing units. The model parameters were tuned to represent the daily energy usage of an area populated roughly as much as 1% of the UK (like a big city such as Glasgow in Scotland), with mean usage and expected variations at each hour of the day. In our first experiment, we produced the baseline usage to find the likelihoods of exceeding the demand (compare with results in Fig. 4.5), this allowed us to evaluate the expected behaviour of the system without a spike botnet.

The next three experiments each modelled a different power prioritisation to meet the demand, this allowed us to investigate which power generator type performs best under the threat of a spike botnet, interestingly there are quite a few differences as highlighted by Fig. 4.6. Fig. 4.5 shows our demand probabilities for a day. It is noticeable the close relation of the demand with the daily power consumption of Fig. 4.3. The difference is that, with this graph, stakeholders may also inspect the probability that the demand will exceed and then anticipate load-changing opportunities of attacks. The lowest probabilities are during early morning (before 6:00) and after 10pm. During the day, there is a considerable probability (around 40%) of finding the system under high load whereas the chances increase to approximately 60% by 5:00 to 19:30, slowing down from then until 23:00.

Fig. 4.6 shows the results from our attack models. As expected, peak hours are more susceptible for spike attacks; however, it is possible to infer the probability of the system where it is in over demand, information that may be used to have auxiliary power generators readily available to avoid energy interruption in the event of attacks. The different controllers actually performed significantly differently with respect to security. Prioritising Hydro led to the scenario where an attack is the least likely to succeed across the day. Nuclear, on the other hand, performs rather poorly, this is due to it's slow transitions from *serving* to *available* and

[3]We have used PRISM version 4.5.

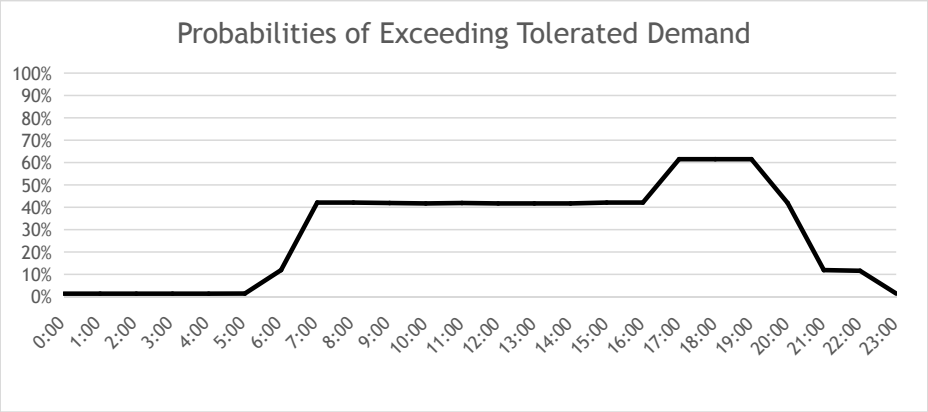


Fig. 4.5 Probabilities for demand raising above tolerated values, in a day to day scenario with our PG setup. Diagram from Arnaboldi et al. [14]

due to it's inability to recover after a spike attack (within 1 hour). Finally, prioritising gas shows more areas of attacks than hydro. It also reaches a slightly lower height at peak time of 18:00 (peak time of success for all attack scenarios), rendering it better over hydro power for that particular time period.

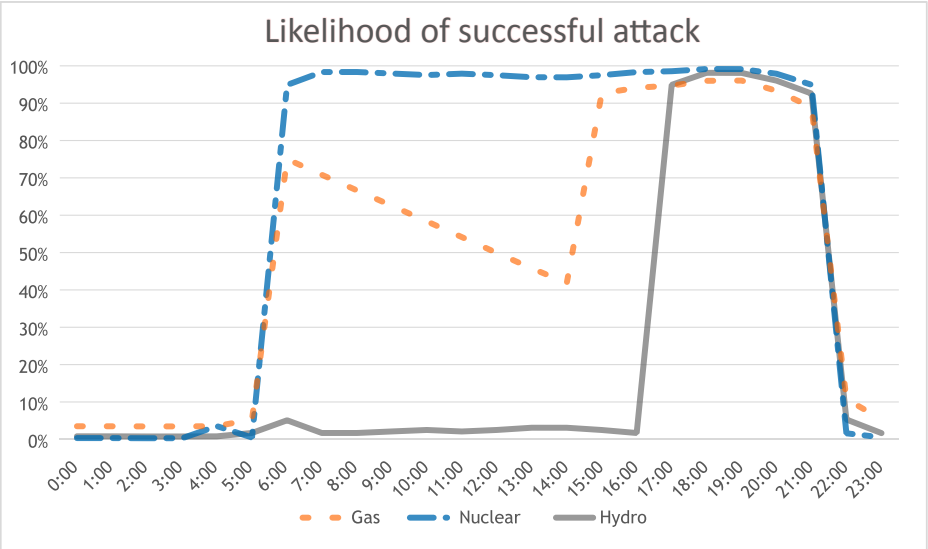


Fig. 4.6 Likelihood of blackouts caused by spike botnets and different control strategies. Diagram from Arnaboldi et al. [14]

We stress that the numbers we used to model the PGs are partially reflecting the reality, as we borrowed them from what specifications we could find, but they may not be as realistic as wanted by experts in real specific domains. Building on our example, one could always re-run

the experiments with different characteristics and settings, to better reflect their scenarios of interest. It would then be possible to compute the probability of being under high or low demand given these new parameters and devise reasonable countermeasures to avoid peak attacks. These peaks are not necessarily cause by malicious attackers, but may happen due to other benign causes. In Brazil in 2007, for example, hackers were promptly blamed as the cause of a major blackout, but further investigation concluded that some faulty equipment triggered a cascading failure from a poorly maintained systems[4].

4.6 Limitations

Both case studies were able to showcase different aspects of the approach. Through the presented evaluation we were able to demonstrate each of the characteristics we had identified as important for modelling IoT systems. However, we encountered certain limitations. Firstly, scalability is quite an issue, as the complexity of systems expands with the number of components and their connectivity this same approach will struggle to characterise large IoT setups. This is a known issue with these types of approaches[48]. Although our novel lightweight modelling of network connections allows to observe various scenarios, as is exemplified by the case studies, this is ultimately something all model based investigations will be limited by. Secondly, we made the assumption that the wellness of the IoT system is measured as a quantification of expected battery life under attack. This may omit factors around protection of key resources and doesn't take in consideration system specific knowledge. Furthermore the way we modelled battery drainage was based on tests performed on a single device type and in a controlled lab environment. Although our model was able to replicate the results from the lab based environment [16], we know battery drainage is impacted by several external sources which may not be visible in a lab and consequently results may vary when deploying such a system in a non controlled environment. Expanding on this, whilst for both these approaches we have attempted to match the evaluation to real data, one in a lab setting and one from data from the UK powergrid. We cannot and do not claim to guarantee that

[4]Official document: http://www.aneel.gov.br/cedoc/adsp2009278_1.pdf

the security evaluation is completely representative of the real world. This is particularly true of the second case study, although we discussed this strategy with power-grid experts which approved of our approach, evaluating this on a real environment is an impossibility. These assumptions are made in order to allow models to scale, and whilst there is a trade-off between accuracy and scalability we have provided evaluation to show that this compromise still allows for interesting evaluations. Finally although we have set the foundation for a modelling methodology able to abstract device, attacker and mitigations in IoT systems to evaluate their impact we do not claim to be able to cover all circumstances and certainly extensions will need to be put in place for research in new attack techniques.

4.7 Chapter Conclusion

This chapter presents a formalism to model interconnected IoT devices and evaluates it with two different IoT scenarios. The use of CTMC allows to model complex device interactions with an element of time which is crucial in any security assessment. Their stochastic nature also allows to observe complex device behaviours and to models various different scenarios. This very high level device abstraction has the key strength that quite large systems can be abstracted without running into state based explosions as quickly. Furthermore, alongside the ability to observe interactions on the system, we also showcase the power of formal verification to assess the system. Allowing a system designer to run security assessments without need to implement the system (albeit necessitating knowledge of his expected deployment).

Modelling the behaviour of devices was the first step to the ability to deploy model-based IDSs. System specifications using formal models have the unique benefit that it is highly flexible and therefore more adaptable than implementing a hardware system. This chapter showcases how this modelling formalism is able to capture vastly different scenarios and even includes the ability to model various other security components. Allowing to run a more thorough attack analysis to observe many different attacker behaviours. This is particularly useful as an IDS often does not sit in isolation within a system and is often paired with other

components. One drawback of this technique is that device interactions need to be manually modelled. Conscious of this drawback in Appendix F, a technique to easily specify device interactions through use of a graphical interface is presented however in the future we wish to investigate more robust approaches making use of model learning [1].

Chapter 5

From Model Behaviours to Intrusion Detection

5.1 Chapter Summary

In this chapter the formal modelling technique for the impact evaluation of attacks on IoT systems is presented. This is partially published in two works [17, 18]. Our Lightweight IoT System under Attack (LISA) [18] specifications are used to model the behaviour of an attacker on an IoT system. This approach is then applied for use in attack detection [17] as per our research aim. A technique is also presented to match network data to model traces in order to apply this detection. We showcase the flexibility of our approach in a theoretical constrained IoT deployment involving battery powered devices. These behaviour traces are then used as the basis of detection for a static attack trace from a real-world hardware IoT testbed. A case study evaluates the effectiveness of detection for a battery DoS attack on a constrained IoT system. A further extension is also discussed for usage alongside existing machine learning systems. The chapter also provides insights on potential extensions, limitations and future works for this approach.

5.2 Chapter Introduction

This chapter presents our methodology for the deployment of model based intrusion detection in constrained IoT. As highlighted by our summary of the literature in Chap. 3 there are several works making use of behaviour based detection in IoT systems. However, the literature highlights there are gaps in this type of detection in the IoT and that by itself it may struggle to detect these attacks, or be unable to understand why the classification was made (Chap. 3 Sec. 3.6).

A key new difficulty in deploying an IDS in constrained IoT systems is that having access to or gathering bespoke data for the system may not be feasible [64]. In constrained IoT deployments there may not be the luxury to evaluate the system in traditional means e.g. by doing penetration testing [87, 10]. These critical circumstances might therefor require for alternate means to perform security assessments and provide security to the system. As such this chapter presents a technique based on the formalisation of the interactions, allowing to bypass the data collection/pen testing step to perform a security evaluation and secure the IoT deployment.

Due to the complexity of attackers, it is not simple to create specifications of attacker behaviour, as a simple variation may be enough to circumvent them [44]. So the quick generation of large and impactful attack traces allows for predicting potential attack variations. This can be generated prior to a system being deployed without need for bespoke data collection, something essential to certain IoT scenarios we have identified. This requires for the system administrator to have knowledge of the system it is protecting, such as battery consumption of the devices actions, and consequently will require a complicated setup phase. We evaluate this approach on a IoT scenario and investigate its effectiveness in finding these attacks that harm the core aspects of IoT systems (Chap. 5.4). A larger case study of this same approach is currently in development, with specific focus on nightingale hospitals, the work so far is presented in Appendix E.

In order to successfully train an IDS for a bespoke system, a security professional needs to collect large quantities of data. To gather this data two main options exist: 1) make use of known attack datasets to train the IDS, or 2) make use of an exploit database or other pen

testing suites and replicate attacks on your own system. However, they come with several drawbacks [120], and may not be feasible for the particular scenario we have identified (Chap 1.1). We eliminate the usage of the first approach, as getting similar datasets for these particular scenarios is unlikely and in practice the uniqueness of these deployments requires bespoke data. This second approach is more precise [120, 32] as it allows to search for bespoke attacks to the IoT network and construct a dataset which is unique and effective for the specific system. Whilst this approach produces the best suiting dataset it is not applicable for the following reasons: 1) One must find and implement the attacks, which is a difficult process that might take a very long time [120], which is unavailable in critical scenarios. 2) One may need to cause major disruptions to one's own network by running the attacks which might obstruct work and productivity. 3) We do not have reference of attacks on these systems as behaviours will be unknown and perhaps sporadic. We therefore require a further approach.

We extend the work presented in Chap. 4 by providing transition rules to mimic network communication and using non determinism to mimic complex attacker strategies. We formulate the problems as the following: **P1:** is it possible to train a IDS for a specific IoT system making use of a model rather than system data from a physical implementation? The model would be able to simulate a real-world security assessment (e.g. penetration testing) but would have the advantage that it could run parallel to the real system without causing downtime (Fig.1), or even be used prior to having the system implemented. **P2:** by making use of non-determinism and probabilities could the modelling approach recreate rich attacker behaviours to train a predictor that is harder to circumvent?

The remainder of the chapter is broken down as follows, Sec. 5.3, discusses similar work in the area of modelling system interactions and attacks, Sec. 5.4 presents our modelling methodology, case study and experiment results used to evaluate our signature based approach and provide solutions to P1 and P2. Sec. 5.5 discusses as a further extension for anomaly detection using automatons, Sec. 5.7, provides insights and limitations about our methodology as well as discussion regarding future extensions, and Sec. 5.9 concludes the chapter.

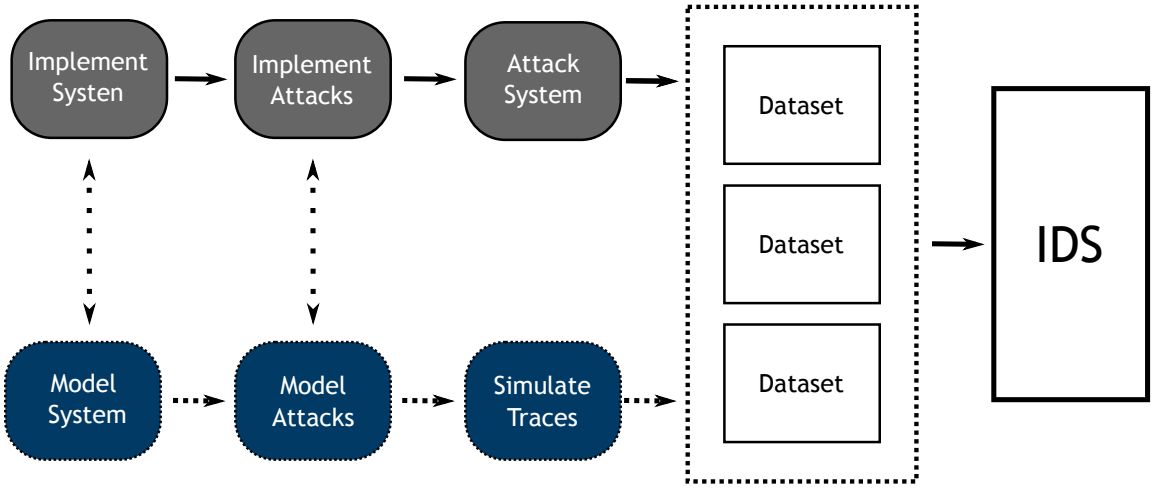


Fig. 5.1 Running model along-side real system (or without necessitating any implementation) to generate further datasets and train an IDS. Diagram from Arnaboldi & Morisset [17].

5.3 Related Work

Several papers address modelling IoT, adopting various different approaches. Fruth [67] examines various properties of a wireless network protocol namely connectivity and energy power through PRISM, including quantifying the battery drainage of certain randomised protocols. In previous work [16] (Chap. 4) we model basic flooding DoS attacks through PRISM and look at the effectiveness of different attack strains against a mitigation technique, in defending systems of interconnected IoT devices.

Our proposed method combines these approaches to recreate an accurate representation of system behaviour and represent a wide range of DoS attacks. PRISM has been widely used as a excellent method to evaluate and verify models of IoT systems and protocols, combining these two models by adapting both the system models and the attack models we successfully model the behaviour and general properties of a bespoke IoT system. We then use the inbuilt verification capabilities to ensure correctness relative to mimicking system behaviour by establishing benchmarks and tests.

DoS attacks have long been one of the most common and dangerous threats in any internet system. These attacks become even more dangerous as the IoT spreads across a vast amount

of spectra and parts of life including safety critical and potentially life endangering ones such as IoT Healthcare and Intelligent Transportation Systems (scenarios in Chap. 1.1).

The extant literature highlights several new DoS attacks against IoT system taking advantage of unique qualities and IoT infrastructures [109, 150, 37]. One such attack, battery drain attack focuses on exhausting the devices battery power as replacing it might be costly, difficult and lead to extensive periods of downtime. These kinds of attack are very subtle as the behaviour of the attacker might not necessarily mimic more common attacks such as pure flooding, they attempt to find battery intensive operations (not necessarily malicious) and repeat them until the device is out of power. This is only one specific example of the literature cited above, however, what all of the above have in common is that they are specialised in their intent of disrupting IoT devices and many of the current detection systems do not account for them [150].

The growing use of internet services in the past few years have facilitated an increase in DoS attacks. Despite the best preventative measures, DoS attacks have been successfully carried out against various companies and organisations enforcing the need for better prevention/detection mechanisms. This is partially due to the vast new avenues of attack (often unique to IoT) that rule based schemes such as SNORT [149] struggle to detect. Further work attempts a more scalable approach that models behaviour of a network (stationary or non-stationary) and labels abnormal packets as a potential anomaly [31]. Limitations of this approach are a large number of false positives as well as lack of information regarding the attack (e.g. the specific vulnerability the attack relies on) as opposed to a rule based based IDS which can indicate what rule is broken. The approach suggested in this chapter allows to bypass these limitations. By modelling behaviour of a system, one can detect anomalies by modelling various attacks, it can also provide accurate data of the system behaviour whilst being targeted, allowing for less false positives. To predict “unknown" attacks, the modelling approach uses a stochastic attacker that attempts different behaviours allowed by the system policy. Using this data it can create a wide range of attack signatures and simulate an attacker probing the system. Similarly to a rule based technique we can maintain actionability as we are able to return to the model to observe why anomalies are observed.

We are not the first to seek means to formalise network traffic in a formal manner, as Xu et al. [185], used a similar approach using finite state machines. They do not use it in the context of security however they showcase how it may help in gaining useful insight into network behaviour. Like the previous work we use a very similar feature extraction approach to construct network traces, with some slight variations to fit our desired construct.

5.4 A Model Based Approach for Deployment of a IDS in an IoT Network

In brief, our model is a Markov Decision Process (MDP), representing the IoT network, the attackers, and some processes monitoring the security metrics under consideration. A trace of the model (corresponding to a sequence of actions of the MDP) should match a trace of the actual system, and vice versa, such that it becomes possible to train a IDS for the actual system on the traces of the model. The main strengths of our approach is the ability to easily represent various configurations for the IoT network as well as multiple types of attackers. MDPs have some key advantages: they have substantial tool support such as PRISM Model Checker [99], they rely on probabilities and non-determinism to recreate systems and they provide the ability to find the optimum paths through the system using the reward function. Through the reward function we create traces of behaviour that mimic attacks on systems by assigning rewards to successful (damaging) behaviour. Our results highlight that through this methodology we were able to consistently produce datasets that resulted in accurate IDSs (detecting attacks on real world systems) and that could be trained in a fraction of the time. The core contributions of this work are 1) A model of an IoT system that enables the generation of synthetic data sets of network behaviour 2) Modelling of attack behaviour against a system to train a real world IDS 3) A quantitative analysis and validation of this model against a real world implementation of the same system to validate our methodology.

The work is split into the following sections; In section 5.4.1 we introduce our IoT system model and attacks model that generates the network behaviour; In section 5.4.2 we highlight our assessment methodology; In section 5.4.3 we discuss the setup for the experiment;

Section 5.4.4 provides an analysis of our results and section 5.6 concludes and discusses future work.

5.4.1 IoT System Model

We model the system as a synchronisation of three core components: a set of devices, a set of monitors (each assigned to a device) and an attacker. We also measure the impact of the actions on the system, specifically their effect on the devices battery and ability to operate successfully. This is something not available at the network level and it allows to make much more informed decisions. Common process algebras such as CSP by Hoare [82], the semantics of the traces are a set of action of the processes. These kinds of traces allow for a human reader to understand the way the system operates. However an IDS can draw very little information from these traces and they fail to capture the concept of *messages* through the network. Specifically we need to be able to capture the intercommunication between devices at each transition within the trace (as per a log in a real system). We customise the trace semantics of standard process calculus, to produce datasets. The formalism captures the effect of a device *sending* an action and the other device *receiving* it.

Our approach is general and could in theory be applied to several other modelling paradigms e.g. petri nets.

The system as a whole is a tuple $\Phi = (D, M, T)$ where $D = \{D_1, .., D_n\}$ is a set of devices, $M = \{M_1, ..., M_n\}$ is a set of monitors calculating properties of their corresponding device and $T = \{t_i, ..., t_n\}$ is a set of times to calculate the changes over time of the system as a consequence of actions being triggered. We also introduce means to model an attacker as a malicious device. Given a global set of actions γ, a device D is a pair (A, P), where $A \subseteq \gamma \times [0,1]$ is the set of *active* actions, where $(a, p) \in A$ means the process chooses action a with probability p, and such that $\sum\{p \mid (a, p) \in A\} = 1$; and $P \subseteq \gamma$ is the set of *passive* actions.

We model the set of actions of the device as the communication capabilities of the real world device. This allows to capture the full set of abilities of its behaviour and increases the accuracy of the benchmark. This also eases the addition of further devices as they are simply

modelled with the full send and receive action spectrum without the need to alter the rest of the system. The behaviour of a device is in the form of a guarded communication, which in our model means that the communication is reliant on a set of conditions being true in order to be triggered (Rule 1,2 and 3). An action a in the device can only be triggered to begin a communication if it doesn't violate the capabilities of the system, such as remaining battery and time per message, which is enforced by a monitor.

A monitor is the part of the system that enables its correct functioning as well as monitoring dangerous behaviour. It calculates the shifts in battery of the various actions and synchronises with the devices to ensure correctness. A monitor M controls value λ, where λ is the remaining battery of the device. Given a global set of battery drains Ω the λ is measured as a quantity that is linearly drained by a ω_a where $\omega_a \in \Omega$ is a constant battery drain of an action, the monitor will update its λ value to λ' after each corresponding device action. The drain of each action is a fixed value calculated from the real world device, as such each action is associated to a single device only.

Rule 1. Given two devices $D_1 = (A_1, P_1)$ and $D_2 = (A_2, P_2)$, a communication initiated by device D_1 on an active action $a \in A_1$, triggering corresponding receive action $\bar{a} \in P_2$ in D_2, with an associated probability p takes the form:

$$\frac{(a,p) \in A_1 \quad \bar{a} \in P_2 \quad p > 0}{(A_1, P_1)||(A_2, P_2) \xrightarrow{(a,p)} (A_1, P_1)||(A_2, P_2)}$$

Rule 2. Given monitors M_1 and M_2 holding battery values λ_1 and λ_2, devices D_1 and D_2 are controlled by their respective monitors. The monitors calculate the drain in battery caused by action a and $\bar{a}$ from constant drain values ω_a and $\omega_{\bar{a}}$ in the form:

$$\frac{D_1||D_2 \xrightarrow{(a,p)} D_1||D_2 \quad \lambda_1 > \omega_a \quad \lambda_2 > \omega_{\bar{a}}}{\lambda_1 \rhd D_1||\lambda_2 \rhd D_2 \xrightarrow{(a,p)} (\lambda_1 - \omega_a) \rhd D_1||(\lambda_2 - \omega_{\bar{a}}) \rhd D_2}$$

These measurements can further aid the IDS in making informed decisions regarding the impact of the various actions in the system and were used to quantify the effectiveness of the attacker. Through this synthetic data the IDS will get a wide range of attacker behaviour that

will lead to system failure, including potentially unknown attacker behaviour. Doing a similar approach without the model would require attacking one's own system and implementing an attack to collect data as per a penetration test (these approaches were compared in the experiments in section 5.4.2).

We differentiate each transition as a network packet running through the system, checked by the monitor of the device. Therefore they must be unique and fit all the possible behaviours of the device. As each action belongs to a single device it enables the corresponding devices to be uniquely identified.

Rule 3. Given two devices controlled by their monitors in the form: $M_1 \rhd D_1$ as CD_1 and $M_2 \rhd D_2$ as CD_2, and taking the total set of devices X, then the transition between CD1, CD2, taking system time t and being performed with probability p takes the form:

$$\frac{CD_1||CD_2 \xrightarrow{(a,p)} CD_1'||CD_2'}{(t, CD_1||CD_2||X) \xrightarrow{(a,p,t)} (t+(t_a+t_{\bar{a}}), CD_1'||CD_2'||X)}$$

In the computational view we compose a trace of the system inductively as a set of transitions in between states, where *prefix* is the prior transitions and the diagram describes a single transition in the form:

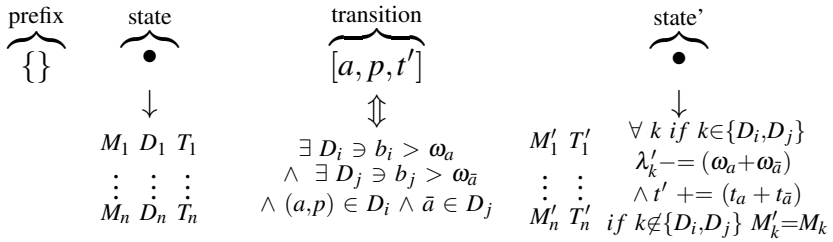


Fig. 5.2 Computational view of systems transitions

A trace of a model under attack should be a subset of the full (finite) model trace. The traces are however limited by the drain of battery either by standard behaviour or by attacker behaviour, as devices out of battery stop performing actions. This means that going from a set of traces one can reconstruct a data file of what has taken place in the system. The output

of the model is a set of transitions following the semantics described. More formally a valid trace σ_i within a system Φ is a finite list of tuples (a, p) active/passive actions, decorated with a discrete time t, which follows Rule 1,2 and 3 until termination as per Fig. 5.2. Given the maximisation of a certain properties the system selects the most optimum action to fulfil the desired condition. For our case study this involved looking for the paths that would deplete the least battery in the system.

By generating the outputs of the system as the behaviour spectrum, the model can describe everything that can take place in the system. By updating the probabilities we can cater to the specifics of the underlying system behaviour and make use of this to find unusual or potentially malicious behaviour. The rules can expand to include a wide array of behaviours and specifics to regulate devices actions and when they can be activated. These can include complex policies on whether actions can be activated at a specific time or whether some actions have higher priority allowing for very specific behaviour to be modelled.

Attacker Behaviour

An attacker synchronises with a subset of actions of the device. When an attacker synchronises on the device the monitor will synchronise on that action and calculate the respective drainage. The monitor keeps track of all these measurements for its respective device. Implementing the model in a tool like PRISM allows us to make use of Probabilistic Computation Tree Logic (PCTL) [26] to calculate various conditions of pertinence to the system, to compute the optimal attack path, and to simulate traces of the model.

An attacker's intent is to behave in a manner that shortens the traces of the system by draining the value of battery in the monitor in the most efficient way possible. To model the attacker we made use of non-deterministic behaviour in order to allow for anything to take place at any point. The advantage of non-determinism is that it allows for a system to arrive to an outcome using various routes. This can be manipulated to find optimal routes through the system and simulate varied behaviour. Unlike devices that are restricted by time and batteries of the devices they model, we allow for the attacker to have different levels of power to simulate various attacker strengths. An attacker, like the devices, has a set of

unique actions γ_A, however unlike other devices does not have a set of passive actions as it sits outside the connectivity of devices and cannot receive messages. An Attacker may synchronise with any device in the system, and the set of actions $a_{\gamma_i} \in \gamma_A$ each correspond to different types of attacks in the real system. To expand further on the actions of the attacker, these should be very flexible and we make allowance for any action that can take place in the system (only restricted by the setup and protocols).

Each action label will correspond to an attack message from the real attacker and can be converted for the log file. For our specific example, each action in the attacker corresponds to the attacker in our experiment sending different packets/targeting different parts of the system as per the *attacker experiment* in section 5.4.2. Beyond actions it is important for us to be able to monitor the behaviour of attackers looking at how many actions an attacking device can perform at a time T (whether by assuming a real attacker device or by simulating different powers of attack). This is highlighted by measurements of the system we implemented that were then modelled in the monitor of each device. The other information to keep track of is: the choices the attacker makes to take down the devices, as these are important behavioural patterns for the IDS to use and can give us insight on potential vulnerabilities as well as unknown attacker behaviours.

Unlike with the devices (whose intention is to cover the full spectrum of possible behaviours with the attacker), we are particularly interested in targeted behaviour. The attack actions therefore encompasses behaviours which are particularly damaging to the system (e.g. causes large drain of battery to the devices). As opposed to probabilistic behaviour we use non-determinism to find paths of behaviour that are particularly rewarding in terms of time taken to take down the system and in terms on lowering system usability (e.g. message throughput). To model non-determinism we remove the probabilities from the attacker action. This differs from probabilistic behaviour because the non-deterministic choice between process A and D_x is resolved at the moment the first action takes place. Conversely in the case of a probabilistic choice is done before the actions takes place [11], so if there is a conflict in the system where both probabilistic actions and non-deterministic actions exist the probabilistic action is resolved first. By not associating a probability to an action we allow

for the strategy of the attacker device to vary depending on what we are looking for in the system. Given a *policy* regulating the behaviour (corresponding to the available attack types) we allow for any action to take place at any point. This can be combined with a set of rules to find the trace of behaviour that allows to follow all the rules and yet still drain the battery as quickly as possible within these restrictions. Instead of a probability each action has an associated reward, and one can use this to find the path of most reward (or the best strategy to take down the system).

The non-determinism in combination with the reward structure *time* is used to find the optimum *attacker* strategy, or the most *rewarding* trace through the system. In PCTL it is written as $R\{\text{“}time\text{”}\}min =?[\ \mathrm{F}\ power = 0\ ]$ or the minimum time for the variable power (referring to battery levels) to reach 0. The value “time" is a variable calculated by the time for a single message to be sent by the attacker and cumulated for each message sent before the power reaches zero calculated in microseconds and the power drain is calculated by the formulas in section 5.4.2. These reward structures allow for simulated attack strategies that an hypothetical attacker might make to take down the modelled IoT system. Not all attacks rely on speed and intensity to take down the system, as highlighted by the running example in 5.1 where the longer trace (Trace 2) is faster, so we model different rewards and observe different attacker behaviours. We generate traces of less detectable attack by associating an predictability score to an action and therefore keeping the behaviour varied and realistic whilst still optimising time. This can scale to several scenarios. We use these “optimised" traces to create a large dataset that mimics different kinds of attackers.

Example Showcasing Model Approach

We show an example composed of: devices D_x,D_y and D_z, corresponding monitors M_x,M_y and M_z, and global time t. The small example is displayed graphically in Fig. 5.3.

Each device has different actions that are synchronised with some other devices. The monitors have battery values for the devices and each device action has a set $\omega_i \in \Omega$ of action drains impacting the monitor battery λ. Transitions follow the described rules to

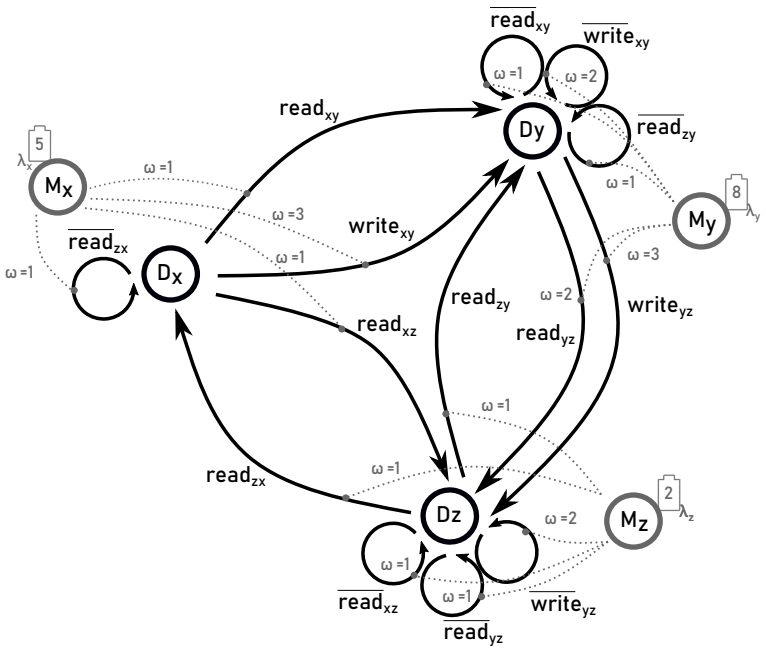


Fig. 5.3 Graphical representation of communication between example scenario devices, and monitors calculating their resources

construct the traces. Note that they do not represent the full possible set of traces but rather two simulations of the system until devices are drained, full details presented in Tab. 5.1.

Table 5.1 Example system model and its outputs

Devices :	$D_x = (A_x, P_x)$ where
	$A_x = \{(read_{xy}, 0.3), (write_{xy}, 0.5), (read_{xz}, 0.2)\}$ and $P_x = \{\overline{read_{zx}}\}$
	$D_y = (A_y, P_y)$ where
	$A_y = \{(write_{yz}, 0.8), (read_{yz}, 0.2)\}$ and $P_y = \{\overline{read_{xy}}, \overline{write_{xy}}, \overline{read_{zy}}\}$
	$D_z = (A_z, P_z)$ where
	$A_z = \{(read_{zx}, 0.1), (read_{zy}, 0.9)\}$ and $P_z = \{\overline{read_{xz}}, \overline{write_{yz}}, \overline{read_{yz}}\}$
Monitors	$M_x \ni \lambda_x = 5$ and $Drains_x \ni \Omega_{A_x} = (1,3,1)$, $\Omega_{P_x} = (1)$
& Drains:	$M_y \ni \lambda_y = 8$ and $Drains_y \ni \Omega_{A_y} = (2,4)$, $\Omega_{P_y} = (1,2,1)$
	$M_z \ni \lambda_z = 2$ and $Drains_z \ni \Omega_{A_z} = (1,1)$, $\Omega_{P_z} = (1,2,1)$
Trace 1:	$[write_{yz}, .8, 30]$ $[write_{xy}, .5, 50]$ $[read_{xy}, .3, 65]$
Trace 2:	$[read_{xz}, .2, 8]$ $[read_{xz}, .2, 16]$ $[read_{xy}, .3, 31]$ $[read_{xy}, .3, 46]$ $[read_{xy}, .3, 61]$

This very simple example may match a critical scenario involving three sensors. One might wish to use the sensors to know information about an earthquake that has just happened. However the sensors have limited batteries and therefore their usage is limited. Going through a simple trace (Trace 1) we observe there is an initial $write_{yz}$ action, it has associated

probability $p_{yz} = 0.8$ and takes an accumulated 30s. Following the rules, we see that in order for this action to take place (Rule 1) there needs to be a corresponding read action ($\bar{a} = \overline{write_{yz}}$) and the probability of this action taking place must be greater than zero. As an outcome of this action we also have a consequent battery drain (Rule 2) on each of the two device monitors M_y and M_z, based on the associated drain of the actions (Ω_{Ay} and Ω_{Pz}), assuming sufficient battery is remaining, this results in new values ($\lambda_y - 2 = 6$ and $\lambda_z - 2 = 0$). A trace may continue in this manner until the rules can no longer be applied.

Model Insights

In practice, even with the very simple case study presented we can see several variations. Given even a single simple optimisation criteria the corresponding traces will differ drastically. Showcasing this with an example, say we are an attacker attempting to drain the system as fast as possible, meaning no further actions can be performed. Given the initial setup as per Tab. 5.1, a possible optimum trace given this objective corresponds to $\sigma^* = ([write_{xy}, .5, t], [write_{xy}, .5, t], [write_{yz}, .8, t])$. Using Rule 2, we can calculate that after this trace both the battery value for D_x ($\lambda_x \in M_x$) and the battery value for D_z ($\lambda_z \in M_z$) are such that no other action can be performed. However given the change of one battery value by a single number from $\lambda_z = 2$ to $\lambda_z = 3$ a further step would be required $\sigma^{*'} = ([write_{xy}, .5, t], [write_{xy}, .5, t], [write_{yz}, .8, t], [read_{yz}, .2, t])$. Furthermore, this is a probabilistic system so the ordering of the transitions might vary. The number of optimal attacker behaviours is calculated as $\frac{(r+n-1)!}{r!(n-1)}$ where n is the number of *active/passive* action pairs in the system and r is the average trace length prior to the system no longer being able to function (i.e. rules do no apply anymore due to battery drainage). So even for this incredibly simple system composed of a couple of actions and limited battery power the resulting number of scenarios amounts to 210 possible optimum attacker traces.

5.4.2 Experiment Methodology

To evaluate the effectiveness of the models we tested and compared the modelled system in 5.4.1 with the more standard approach described previously. We use the collected network

data as the ground truth and compare it to the model traces approach to see the enhancement of performance in various different scenarios. Both the approaches output was used to train an IDS. The IDSs were then used to predict attack behaviour. The experiment evaluations was on the following basis: 1) Accuracy on unknown attack detection; 2) Ability to mimic devices behaviour and *smart* attackers. The setup of the experiment was the following:

Experiment - Device Setup : We set up a small IoT network in the lab and then modelled it to compare the results and to test out the effectiveness of our model in creating synthetic dataset. For the sake of testing we kept the setup simple to display the tool as the thing that needs to scale and not the system, as using this approach changing device setups is much easier than data gathering on a new system. Once the simple model is created it is trivial to add more (similar) devices, whilst implementing a new system in the real world can be very time consuming. We implemented a sensor network consisting of two devices. Each device had the following *actions*; they took sensor readings and then could *send* it to the other device at any time; they could also *request* the sensor data from the other device at any point. The devices used simple HTTP protocol for communication, and the behaviour was stored in Apache log format. To accurately represent the devices and to create *smart* attackers, several measures needed to be obtained. Both devices were equipped with a Mh3500 battery. We made a basic assumption that the devices are on constantly. We argue this is a correct assumption as due to our attack the device is constantly in log mode and therefore never in sleep mode. Beyond this assumption we calculated time to send a message/log a message, baseline battery usage, percentage increase in battery usage under different DoS strains (taken this value and dividing it by messages processed for second) and battery drain per message.

Experiment - Attacker: To validate the model we implemented a common DoS attack both in the real world and in the model. Our attack of choice was HULK, a DoS attacking tool which relies on several obfuscation techniques. This allowed to not be spotted whilst still outputting enough strain to take down systems effectively [1]. The attack specifies it has the following properties: 1) obfuscation of source client - this is achieved by using a list of

[1]HULK, Web Server DoS Tool -Barry Shteiman, Confessions Of A Dangerous Mind, url:http://www.sectorix.com/2012/05/17/hulk-web-server-dos-tool/

known user agents, and for every request that is constructed, the user agent is a random value out of the known list, 2) reference forgery - the referrer that points at the request is obfuscated and points into either the host itself or some major pre-listed websites, 3) stickiness - using some standard Http command to try and ask the server to maintain open connections by using Keep-Alive with variable time window and 4) unique transformation of URL - to eliminate caching and other optimization tools, they crafted custom parameter names and values and they are randomized and attached to each request. The tool was able to take down a web server within minutes from just a single host. Seeing as IoT devices will have less capabilities than any web server we hypothesized that this would be a good attack to use as its properties make for a good dataset that is not straightforward to detect. These properties and obfuscations led to different combinations of message structure that we used in the non-deterministic attacker.

To measure the time it takes per message we measure how many messages can be sent within a time period. This helps evaluate the accuracy in respect to the real world of our test attacker. In order to measure voltage usage across the different IoT devices, we attached an extra component in between the battery supply and the device to take the readings required. To measure battery drainage we utilized IoT battery lifespan estimator tool by Farnell [65]. This was used in combination with a variance we introduced on top of the calculator, to represent attack intensity and change to current. Through this we were able to estimate the different drains of the devices as an outcome of the actions they performed. We created datasets utilising three approaches and compared each dataset in two different experiments.

The first dataset (RWD) was constructed from data from the real system. We implemented the system of devices and the real-world attack and monitored the behaviour of the system. The data was logged across a period of twelve hours and used to train the first IDS. The second approach was a naive approach, we constructed a synthetic dataset (ND) without attacking the system but rather attempting random behaviour. This gave a comparison of the model with a different synthetic dataset this will help evaluate the effectiveness of the IDS predictions as they effectively should be random guesses. And finally, we followed our proposed approach (MD) following section 5.4.1.

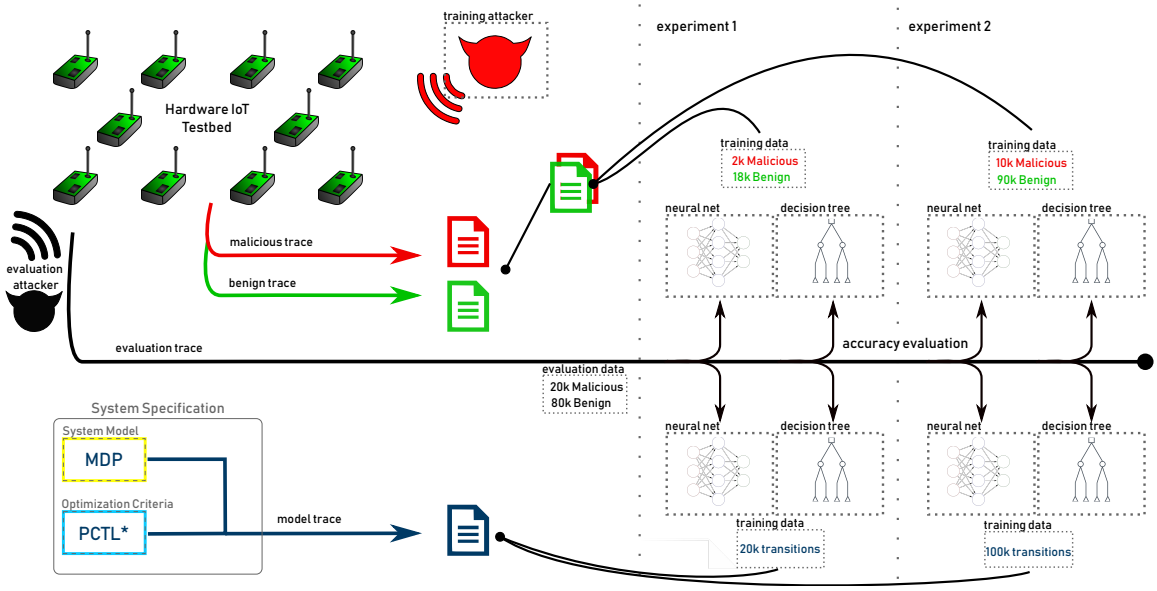


Fig. 5.4 Visualisation of experiment setup

Experiment 1: As our dataset relies on stochastic events and actions, we created three datasets from the approach and evaluated each one to benchmark its effectiveness, a mean score was taken. Whilst our model is able to recreate very large datasets quickly we choose to keep the dataset size uniform across the initial experiment to get a fair comparison against the other two datasets. The comparison was based on accuracy of prediction against unknown attacks given IDSs trained with each of the datasets. The unknown dataset consisted of real world data of the systems behaviour whilst being targeted by attacks that we had not modelled nor contained in the RWD. To measure accuracy we made use of the F score. The F score is a measure of a predictors accuracy, it is a measure of its precision over recall (a measure which takes in consideration both false positives and false negatives).

Experiment 2: The second experiment we ran was to test the effectiveness of the model in creating large quantities of behaviour and the ability to readjust in case of network reconfiguration. We used deep learning classifiers catered to large datasets and created a much more efficient IDS purely through synthetic data. One of the core strengths of our approach is that once the model is setup the datasets are very easy to generate and we wanted to test whether this, in combination with our *smart* attackers, will lead to the ability to train better performing IDS.

5.4.3 Experiment Setup

To perform experiments described in section 5.4.2 we implement a Python framework that runs through the various steps required to test the IDSs: data generation, data processing, normalisation and setting up of the IDS's classifiers. This automatic framework prepares the datasets and trains the IDSs so that we may perform Experiment 1 and 2. It is implemented using the scikit-learn machine learning libraries.

Achieving a rich descriptive dataset was paramount in training an effective IDS. Through the outputted model traces we were able to generate a dataset of different transitions through the modelled system. These traces were descriptive enough for a machine learning algorithms to construct rules about negative behaviour through supervised learning. The traces of the model correspond to the real system behaviour and each transition was labelled as either normal or abnormal behaviour, therefore they can be used to make informed decisions about the system. For instance, if the model traces of the attacker continuously target a device, the IDS can interpret this as a weak point and set a rule to limit this behaviour, as this could correspond to the behaviour of a real world attacker.

To allow for data to be interpreted by machine learning algorithms it needs to go through a process of normalisations. This is often due to categorical non-numeric features or continuous features. The data provided by most if not all internet protocols is categorical (e.g. agent names and method calls). As such, in order to evaluate it we first needed to go through an initial phase of pre-processing. The intent of pre-processing is to render the data machine readable whilst preserving patterns. The process we adapted was the process of binarisation. Binarisation allocates a numeric value to each unique feature for example if dealing with HTTP codes GET would become 0001, POST 0010, DELETE 0100 and PUT 1000. This allows for the features to maintain their patterns and their predictive power and be used normally. This initial step was applied to both the real world dataset and the naive synthetic dataset. This step was however not required for the model dataset as it already produced numeric features rather than categorical ones for efficiency.

The classifiers we implemented represented the IDSs. We choose to use two separate classifiers to get a better evaluation of the results. Each dataset was used to train two IDSs

and then all the IDSs were tested against a new dataset of attack to establish their predictive power and the strength of the datasets.

The first classifier we implemented was Multi Layer Perceptron (MLP) Neural Network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a non-linear activation function [190]. MLP utilises a supervised learning technique called back propagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. A linear perceptron is a function that can decide whether an input, represented by a vector of numbers, belongs to some specific class or not. Combining several together in an MLP and adjusting the functions and weights you build a statistically accurate classifier. The result is a non-linear perceptron that is able to classify non-linear classes.

The second classifier used was a Decision Tree Classifier. A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements [152]. Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). The rules in the branches are automatically constructed from the training data which is labelled. Using these rules it will be able to take in the test data and run it until it reaches an end node corresponding to a class (either DoS attack or normal behaviour).

5.4.4 Results

Following the evaluation criteria in section 5.4.2 and recreating the model described in section 5.4.1, we generated and tested three model datasets against our benchmarks of the naive dataset and the real world dataset. Beyond the accuracy of the results, we make an argument for feasibility and re usability of our approach. The results were acquired by initially training two classifiers for each dataset, these were trained with 20,000 samples of which 10% were attacks. The classifiers were then evaluated on an unknown and unlabelled real world dataset of 100,000 samples of which 20% were attacks (of two different unknown

types). The classifiers then attempted to label the new dataset to predict which ones were attacks.

Experiment 1 - Results The neural network trained on the real world dataset proved to be very accurate with a 85.5% prediction accuracy. On the other hand the model dataset trained predictor whilst still high, suffered from some degree of variance ($79.7 \pm 6.3\%$). What was of most interest however was the predictions outputted by the naive dataset of 0.9%. This combined with the relatively inconsistent results of the synthetic dataset ($\pm 6.3\%$) make a case for over fitting. Over fitting is the scenario in which a model is trained so specifically to the training data that it is no longer classifying DoS attacks and normal behaviour of the system but rather focusing solely on the training data and learning on patters unique to the dataset not the system. This is quite common in Neural Networks as they perform best with very large quantities of data [190], which for this part of the experiment we did not have.

The results of the decision tree, contrasting to neural networks do not suffer from the same inadequacy of over fitting and do not necessarily need large amounts of data. This was mirrored by the results, as the model datasets all performed to very similar standards and the added randomness traces which might have disrupted the neural network made for a more ample rule set resulting in near perfect predicting power in the model dataset (98.8 ± 0.6). The real world data which did not look at the possibility of random behaviour only achieved 77% accuracy and the random dataset had a predictive power of near 50% as expected.

Experiment 2 - Results We observed that our approach of using non-determinism to recreate attack traces was particularly effective for the rule based classifier however led to disruption during the back-propagation process of the neural network, as non-standardized data can create uneven results. This time using the much larger dataset of 100,000 transitions,the results were a lot more accurate (97.1%) than previously, confirming our hypothesis.

As highlighted by this example our model has one key advantage over the traditional approach. Data generation is fast and efficient. If we wanted to improve the training of the IDS used on the real world dataset to a similar level of accuracy, it would take several days of data collection and consumption of resources (electricity, system downtime etc). We argue that whilst the initial effort of creating a model might be time consuming and perhaps not as

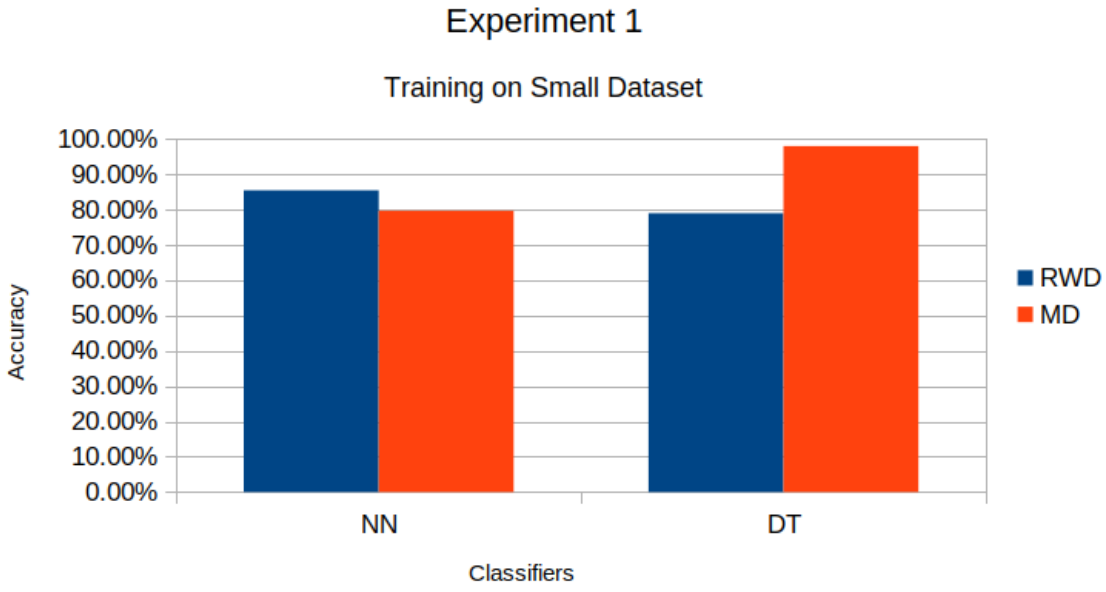


Fig. 5.5 Accuracy of trained classifiers in experiment 1 using model traces (MD) and real world data capture (RWD)

intuitive for a potential system administrator, the phase of dataset generation makes up for this effort both for speed and predicting power of the IDS.

5.5 Automata Based Extension

In the initial case study we investigated the feasibility of using formal model traces as the basis for IDS training. However, the process may also be extended to became the full basis of an anomaly detection system, matching the model traces directly to the traces extrapolated from network traffic. This works by generating an automaton that matches the optimum behaviour.

In order to recreate the system behaviours that match the optimisation scenarios described through the PCTL we use formal verification. In formal verification, finite state model checking needs to find an automaton equivalent to a given PCTL formula, i.e., such that the PCTL formula and the automaton recognise the same ω-language. In the case of PRISM Model Checker, the PCTL is specifically matched by a Büchi Automaton [36]. In this context

this automaton represents the full set of actions resulting from the discovery of the optimal system property of the MDP. Each valid path in the Büchi automaton can be traced back to the labelled transitions of the system model enabling for easy understanding about behaviours and consequences of action sequences. Whilst the states of the Büchi automaton do not correspond to the states of the MDP, a easy matching allows to see the current state in a system walk and to see consequent outcomes of particular behavioural choices in a system. This relation provides insight into how systems behave giving the user understanding about consequences of actions and outcomes of design decisions; allowing to use this for system customisation and evaluation.

5.5.1 From System Data to Model Behaviour

Whilst model traces take the form of a series of labelled transitions, network traces are much more complex. This requires transformation into more focused behaviour before they can be used for detection. We propose a model based network traffic characterisation method in order to compare the two. Each state in the model has specific characteristics related to where the packet is coming from, type of message, time message was sent and time it takes to reach destination. These are all characteristics that can be extracted from network traffic of an IoT system. Furthermore we wish to deduce the state transition system of the corresponding the Lightweight IoT System under Attack (LISA) model, which can then be compared to the expected "good" or "bad" behaviours depending on requirements.

The first step becomes identifying the state of the network (Δ). This is done on a per network package basis for each package within a window N, leading to state space $\Delta = \{\delta_1, \delta_2, \ldots, \delta_N\}$. This window is flexible, based on the type of threats, speed of computation, and length of traces one has for their system. Using the Δ, message flows Ψ are identified within the current window. A message flow ψ, is composed of a series of s, a, t_a, n transitions, each part of which is gathered from the network package. Using UDP/COAP as an example, UDP contains Source and Destination, which correspond to s and n respectively, the UDP message type represents the a and t_a is gathered as part of the environment data. A final transformation occurs for the sake of the time windowed gathering approach, the time for

action a as t_a becomes a cumulative increment from the initial time gathered at state δ_1 which each new time being added on to the next state to mimic the behaviour of the model. Using the characteristics of all the flows within the network capture a transition system is constructed. On account of noise or dropped packets a certain threshold variation α is allowed from the similarities of traces, which can be customised for strictness. Finally for efficiency only the first y packages are compared before a full comparison is made to quickly sort through the current scenarios. A diagrammatic representation of the approach is presented in Fig. 5.6.

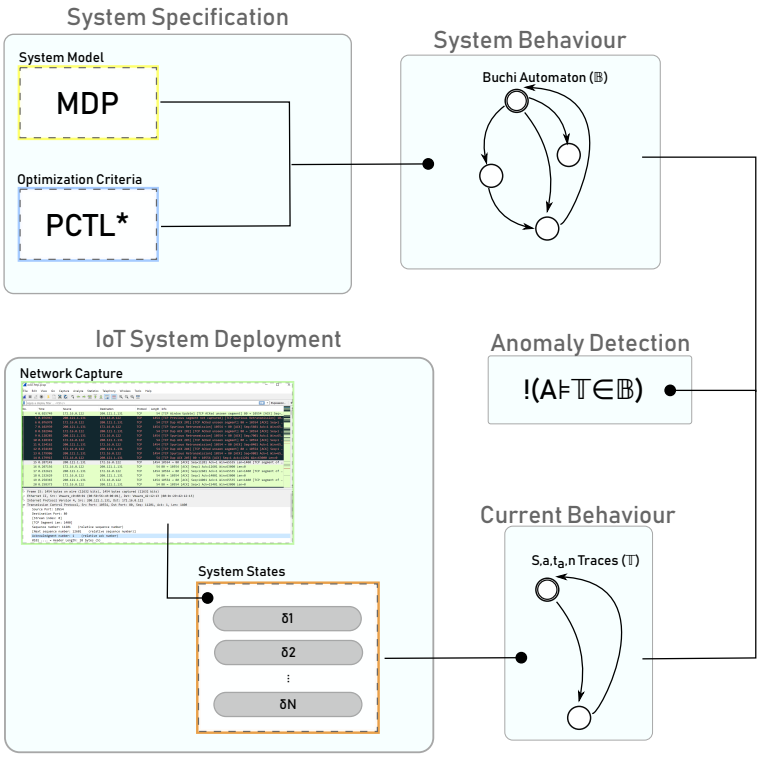


Fig. 5.6 IDS comparison with the model traces to detect deviating harmful behaviour

This much simpler approach suffers from the downside that in its naive form it does not account for probabilistic variations and potential packet drops. A further investigation into the feasibility, potential extensions and performance of this approach is discussed in Appendix E.

5.6 Summary of Approach

Our case study and proposed methodology has shown very promising results. We have shown that generating synthetic datasets of DoS attacks in IoT networks through this tool is effective and can be used as a basis for behaviour analysis. We believe that the ability for this approach to adapt easily to multiple devices and protocols in combination with its strong predictive power makes a good argument for its usage across various IoT networks. Our argument for scalability of this approach is two fold, firstly it scales well in terms of costs as you can make assessment prior to implementing the system and secondly, we can bypass several of the downsides of verification (in terms of state space) as we focus on simulation. Perhaps the most useful feature of our proposed approach is that it allows for the construction of datasets to be very efficient even if a device is added or the system is reconfigured. As this is a prominent concern in dynamic IoT systems this advantage is quite significant. Our approach is based on certain transition rules built on top of MDPs, however the very same or similar rules could in practice be applied to other formalisms.

In this chapter we included a case study of a single attack which worked very well. Our future work envisions the ability to model further attacks from a database to create an extensive set of attacks to create a much more predictive dataset. We envision that the ability to relatively easily plug and play any IoT system in combination with implemented corpus of attacks, could turn into a tool that generates synthetic datasets of attacks to train bespoke IDSs for any specified IoT system.

5.7 Discussion & Future Work

The conjunction between the world of formal models and network behaviour is not a obvious union and only some work has successfully explored its possible usage [68, 185, 125]. Automatons have been used as means for IDS before [68], however the authors have manually specified the actions of an automaton in the form of a specification based IDS. The advantage of a more general formalisation using MDPs is that several different specifications can automatically be observed in the form of PCTL queries. Through a PCTL formula the same

MDP can characterise several different systems leaving room for a rich system description and adapting to different requirements or deployment restrictions. We further expand this investigation beyond the specification of behaviour to use these approaches in unison to machine learning algorithms by using not just the models, but the traces themselves as part of the training process. Whilst this approach takes a further transformation of incoming traffic to be understandable by the classifiers this is not an altogether extra difficulty as data restructure is something almost always performed in ML contexts [192].

We are the only work, to the best of our knowledge, embedding specifics of the devices themselves such as battery power into the verification, in this manner, to obtain better security analysis and context aware system traces for IDSs, something which can be very useful in constrained environments. Another interesting contribution is that through these methodology we may also find unintentional resource exhaustion attacks caused by potential misconfigurations. It also opens up for several performance based evaluations and optimizations.

Whilst modelling is beneficial to gain system understanding as well as embedding knowledge into the prediction, writing formal models manually is hard, so learning approaches are desired. Attempts to use approaches such as these have been attempted such as the work of Misra et al. [125]. In this work the initial representation is adapted and improved based on new data coming into the system. This has the potential benefit that a imperfect representation may be used, perhaps allowing quicker formalisation, and then fine tuned over time. However it suffers from the same limitations of the previously mentioned work and the reduction in manual labour is limited. What is in fact ideal is a learning approach that requires no previous knowledge of the system. These kinds of approaches (without system specification) often focusing on reinforcement learning are a very new field and advancements are being made to allow for this. Some work has even incorporated these approaches in the area of anomaly detection in the context of video recordings [4]. However even with recent advances there are certain characteristics not present in network traffic data and that therefore cannot be automatically learned. Future work in *grey box* anomaly detection, could combine the automated learning approaches of reinforcement learning with

the capabilities of LISA to represent device characteristics to gain the best possible insight into the system, whilst improving usability. We do not yet have any insight into the feasibility of this approach and relegate it to very exciting albeit out of scope future work.

5.8 Limitations

In this chapter we included a case study of a single attack which worked very well. The attacker objective whilst non deterministic, is still based on a preset of behaviours. This specific attacker only allows us to capture a attacker whose aim is to disrupt system behaviour. We are consequently not able to observe attackers whose intent is to instead obtain information from the system or take over a device. Our future work envisions the ability to model further attacks from a database to create an extensive set of attacks to create a much more predictive dataset. We envision that the ability to relatively easily plug and play any IoT system in combination with implemented corpus of attacks, could turn into a tool that generates synthetic datasets of attacks to train bespoke IDSs for any specified IoT system.

One further limitation of this approach is the fact that we are dealing with discrete time values, so certain attacks such as a traditional flooding DoS may be lost within a certain window, and further tuning may therefore be needed. However, knowledge of these limitations allows us to work around them by introducing mitigations such as rate limiting. The types of attacks that may be covered by our approach are limited to detection of unknown suspicious behaviours by a node in the system, any protocol based attack such as man in the middle, routing based attacks, and resource exhaustion attacks.

Whilst modelling has certain advantages, there are some explicit scalability issues, as discussed in previous chapters. With this lightweight behaviour specifications we were able to model systems with very constrained behaviours of up to twenty devices [14], and we suspect this could scale further in loosely connected environments such as RPL. However, this is in part due to the attack type we have constructed in our experiments. If we were to expand to more complex attackers, as is our objective, we might find that even this level of scalability is lost.

5.9 Chapter Conclusion

In this chapter we showcase the power of our formalism to characterise IoT behaviour in the context of a white box specification. Through models we are able to embed domain knowledge into a rich description of the system that can capture a variety of behaviours. We advance this further in Sec. 5.5, and describe how models can be embedded in the detection engine to relate an anomaly directly to a known behaviour. This approach is not without downsides, as the modelling process might be non intuitive and require a potentially quite high initial overhead. Nonetheless, results from our experiments show that this approach is highly effective in detecting various attacks and capturing unique scenarios. A further feature of value is that the modelling approach can be used alongside existing techniques in order to add actionability to the predictions of other algorithms.

Chapter 6

Conclusion & Final Considerations

Reiterating our original aim:

The overall aim is to provide a new, well tested, methodology for the deployment of knowledge based IDSs for constrained IoT systems - through the use of white box formally defined system models.

In the process of achieving our aim we have contributed significantly to the existing body of research resulting in several publications, each included in this thesis [16, 18, 17, 14]. The methodology specifies devices as modular concurrent MDPs describing the IoT system. We present new trace semantics and rules to specify communications between these devices, allowing to capture routing behaviour. We devise means to capture elements of battery drain, by means of *monitors* - a contribution allowing to enrich the performance analysis. These further semantics allow to observe attacks pertinent to IoT deployments such as routing attacks and DoS attacks.

Having identified scenarios for which it would be beneficial to be able to deploy an IDS prior to system implementation, we devise an experiment protocol to evaluate the feasibility of our method. The protocol evaluated the effectiveness in training an IDS purely with model data against the training of an IDS using network data gathered from the system. To do so we implemented a hardware testbed and gathered real data of benign behaviour and malicious behaviour by means of a attack test on the system. We demonstrated empirical support of

the validity of this approach by achieving a high level of accuracy of prediction on a further unknown attack dataset from the same system (up to 97.7%).

We were able to address all challenges and provided evidence of having resolved them. In Chap. 3, we extensively surveyed the literature to assess the current state of the art, identify gaps, and propose means to uniformly evaluate the IDSs. In Chap. 4 we devised new means to capture device interactions through an Markov chain based approaches, showing how they could be used to evaluate the system. In Chap. 5 we present the new enriched trace semantics used for behaviour analysis, present our experiment methodology, and evaluate our approach. The following discussion evaluates the degree to which we achieved our aim and addressed the identified challenges.

Challenge 1 - Evaluation of IDS Solutions

The first difficulty identified as an obstacle to achieving our aim was the lack of unified evaluation. This was addressed by our survey presented in Chap. 3. This contribution categorised the current state of the art in IoT Intrusion Detection. We conducted the most extensive review of IDS tools for IoT systems to date, to the best of our knowledge, and categorised them by technique used, deployment scenarios, attacks detected and evaluation methodology. Our findings showed that out of the 51 collected tools, i) only 4 provided evidence of their evaluation either as a dataset or simulation code, ii) 12 didn't evaluate their IDS in any way, iii) of the 10 that evaluated by means of an execution or bespoke system trace *none* were compared with one another or shared their code, iv) of the 24 using simulation tools only 4 papers compared results with each other. These results confirmed that establishing which IDS is superior and running a cross evaluations is indeed almost impossible.

Our results showed: 1) there is a lack of available public datasets that are pertinent to IoT scenarios and could realistically be used to train and test and IDS, 2) despite the many added difficulties of securing an IoT deployment it doesn't necessarily mean completely new tools are needed but rather additional measures might need to be added to existing tools, and

3) one of the most common trends in IoT IDSs are the vast number of bespoke and unique scenarios, so by this very nature, the lack of comparison may be justified.

To resolve this we propose (Chap. 3.10.1) a methodology for the evaluation of these systems. Given an approach based on virtualisation several setups may be quickly deployed allowing to test the proposed tools in varied environments.

Challenge 2 - Modelling for Bespoke and Unknown Deployments

The second challenge was that there was no existing methodology to model interactions between IoT devices, to run network based security assessments. To be able to evaluate a system prior to deployment and assess attacks that are relevant to the system we devised means to embed our knowledge via models. The system is specified as a Markov chain, capturing device interactions and impact of attacks on the devices. We leveraged the relative simplicity of device behaviours, allowing for more accurate abstraction, to evaluate systems of interconnected devices (Chap 4). We showcased the ability to test out an attack on the system, evaluate a potential mitigation, and perform analysis on its efficacy. This resulted in the ability to assess *optimum* deployment scenarios. An outcome of our evaluation presented in Chap. 4, showed that a technique for DoS mitigation used in server environments (crypto puzzles [169]) may cause negative impact on the IoT system [16]. Showcasing the value of the models we have used in this thesis. We also were able to capture more complex assessments of system setups to identify the best method of mitigating an attack using smart grid as a case study. The results show that by using these models we can successfully replicate real world device behaviours and adapt to the many scenarios existing in the IoT.

Challenge 3 - Model Data For Intrusion Detection

The final challenge was that there was no current trace semantics able to capture the flow of messages between devices for use in a behaviour based IDS. Due to the constrains of the IoT systems we are considering our focus was to model the impact on the devices (Chap. 4) [16]. By focusing on core behaviours to do with routing and device resources we were able to maintain simplicity in our models, whilst still being able to capture meaningful attacks.

Through our case studies we were able to empirically evaluate that these characteristics were meaningful and useful in practice (Chap. 5) [17], showcasing the ability for our models to match to realistic deployment environments.

We investigated the feasibility of using these models as replacement to real world data in scenarios were data is unavailable or unfeasible to collect. We tested this technique in an experiment setup and found that we were able to deploy an IDS from the get go without having to rely on system data, enabling it to be used in these specific circumstances (Chap. 5.4.4).

6.1 Discussion

As we discuss in Chap 3, when evaluating an IDS there are several aspects that need to be considered beyond accuracy. We discuss our model based IDS methodology in terms of evaluation criteria proposed in Etalle S. [64]. As we discuss in the survey in Chap. 3

Adaptability

Adaptability corresponds to the ability for a system to be able to adjust to meet new conditions. For an IDS this might mean a network reconfiguration, new devices being added in, or new attacks being discovered. This is something our approach does very well, showcased in the exploration of multiple scenarios (Chap. 4.1) assessment of different controllers (Chap. 4 Tab. 2 and Tab. 3) and ability to regenerate traces swiftly (Chap. 5.4.4). We show that we can easily evaluate a variety of systems with a lower burden as adapting a model takes a lot less time than re-collecting bespoke network data on a real system.

Scalability

In part due to COVID19 related constraints we were unable to run analysis on scenarios involving large numbers of devices. We speculate however that the compositionality of our approach allows to reduce the complexity by having several independent systems working in unison. This reduces the difficulty in monitoring of each individual system and relegates

detection of specific threats to different subsystem. Although the relative simplicity of IoT devices may somewhat mitigate the onset, it is a well known fact that there are scalability issues with modelling large systems [26]. A known technique to mitigate this explosion is by using statistical model checking. We successfully investigate the usage of this approach in Chap. 4 showing that the analysis is still successful. In practice this could lead to the feasibility of scaling this to several deployments.

Actionability

One of the benefits that our approach provides is excellent actionability. A set of attacker steps from the model traces provides insight into what is going wrong and can be acted upon. We have shown in Chap. 4 that several assessments can be run on the models to gain insight on the system and discover optimal deployment scenarios. This can further mitigate the effectiveness of an attack before even applying the detection. Given quite a large initial effort of formalising the system, relating predictions to known states gives direct understanding. By means of these assessments, a threat model can be constructed so that an IDS may be catered to *security that matters* and is applicable to the system as shown in the detection of unknown attacks in Chap. 5.4.4.

Accuracy

In Chapter 5, we show promising results on a subset of attacks, paying attention specifically to relevant attacks from the literature (Chap. 3). In some of our experiments we are even able to show an increase in detection accuracy compared to more traditional network traffic based approaches (Chap. 5.4.4) [17].

In summary, we have presented an approach that is actionable and highly adaptable, with promising results in terms of accuracy. Although we speculate the possibility of scaling for realistic IoT deployment, this is one of the known downsides of this approach. Whilst the case study presented in Chap. 5 shows high accuracy, we are aware that a limitation of our testing protocols was the usage of a single system as a benchmark, so further analysis may be needed to confirm these results. Very few techniques can perform perfectly in all these

criteria. Examples of this is usage of neural networks and deep learning, whilst accurate they are highly non actionable. In the scenarios we identified for the focus of our research, adaptability and actionability were key factors and we were able to fulfil the aim by focusing on these. Whilst we have only evaluated this approach on small constrained IoT setups, it would theoretically be possible to apply this same approach to any given system, granted one had access to a model of the system and specific threats. In practice it is unlikely to be able to capture behaviours of complex systems in this same way, and this approach is most advantageous in constrained and simplistic IoT deployments where scalability is less of an issue.

6.2 Future Work

Intrusion detection in the IoT is a very active field, and it is continuously evolving (Chap. 3). However there is no doubt that a shift is required towards more understandable AI if we are to make the advancements required to secure future IoT deployments. With the increases in devices and variety of configurations [182], this becomes even more important. Whilst we have done significant work to discover what the current state of the art is in our review of 51 IDSs, and have discovered several interesting trends, we have identified some interesting areas for research: 1) a survey targeting researchers who wish to develop new techniques for specific scenarios and wish to know what areas need further exploration; 2) means for a system administrator to make decisions regarding what IDS option is best for his specific deployment; and 3) a unified evaluation of all the IDSs tools on a single testbed to get a conclusive benchmark of performances. Although out of scope for this thesis, these would help create better understanding of the field.

Even if perhaps simpler than implementing a system, formally defining all the behaviours of the systems in a model is no easy task. We have explored means to ease the specification process by means of a graphical interface, that can jump-start the specification significantly. This however is only the beginning. Techniques for automatically learning a system behaviour from data, such as reinforcement learning or model learning [6], are very useful

means to automatically gain knowledge about a "black box" system. Whilst perhaps not as customisable as a from scratch knowledge embedding, this "grey box" approach has the advantage that it may be fully automatic and scale better.

Protocols are deeply embedded into how devices interact. One of the many defining features of the IoT is the advancement of protocols for new scenarios [83]. Our interactions models in Chap. 4 was based on realistic protocol flows and the attacks in Chap. 5 on realistic threats, informed by other lateral works [19, 15]. Several IDS papers in the IoT have focused on protocol attacks as their area of detection [68, 62]. Fu et al. [68], propose an automata based IDS that focuses specifically on protocol attacks. When data is incoming to the IDS it is abstracted as an event in the automaton and deviations are seen as malicious. This specification based IDS makes sure that protocols are followed to the letter. Whilst on the other hand, Esquivel et al. [62], make use of BACnet specification diagrams as means to find deviating attack behaviours. However what we have observed from our formal protocol analysis is that even perfectly valid protocol implementations might contain attacks, and require further verification of the protocols. Due to this observation we found that using the protocol verification aspect to know which attacks are relevant may be a powerful approach in the context of IDSs.

6.3 Limitations

In this thesis we have conducted a thorough investigation on the feasibility of modelling for the use of attack impact evaluation in IoT systems. Although our results are promising for a subset of attacks and systems we have observed that modelling can only scale so far. In order to capture any form of impact on the devices we have had to abstract impact of battery drain as an outcome of messages. Whilst this was done using lab based experiments that fingerprinted the drain of a device and follows our real world analysis, this assumption will probably not hold if that same device were deployed in different environments, as we do not account for external factors in the models. Beyond this whilst battery is a realistic factor to consider in the IoT we do not consider more complex and dynamic constrains, e.g.

devices coming in and out of availability due to software updates and maintenance. The added complexity of capturing these scenarios would not be scalable for any sort of large system. Whilst there are other constrains that may be feasibly handled such as message queues, observing multiple such constraints together would add immense complexity.

Although we have observed a varied set of scenarios showcasing adaptability of this approach, a case study in a real setting such as smart factory, would be greatly beneficial to gain further insight. We make assumptions about types of devices and connectivity issues that are inline with literature, however as we did not have avalable meaningful datasets or system configuration, of different intrusions and mitigations we cannot guarantee our implementations are accurate. Particularly in Chap. 4, we may have exaggerated the way client puzzles are implemented in the real world, as more realistic environments may have conducted a preliminary assessment akin to our own before selecting security measures. Having this information would help increase the robustness of our results.

6.4 Concluding Remarks

In a world where machine learning is omnipresent there is very little room for non explainable predictions and black box models. So we instead work towards an enhancement to knowledge based representation able to capture semi unknown behaviours and analyse complex behaviours non-deterministically. The field of explainable AI and white box machine learning is an emerging and incredibly exciting field. We have just begun to scratch the surface in this, and there will be no doubt countless advances in the coming years. What we explore in this thesis is a deviation from standard research in behaviour detection, choosing instead to focus on the way we can explain predictions and embed our knowledge. We investigate scenarios in which there may be no other option but to use white box knowledge to construct explanations of the systems and enhance the IDSs predictions. To do so we devise a methodology that allows quick deployment of these types of systems. Our results show that by means of formalising simple interactions we are able to observe complex behaviours and learn a lot about how attacks effect systems. Our findings provide valuable insights into advantages,

and some disadvantages of using such approaches. This exploration hopefully serves as a stepping stone for future research in this area allowing for several new advances in the field of “white box” detection for constrained IoT deployments. We hope our practical usage of formal modelling techniques helps connect these three traditionally very separate fields(formal verification, network security and machine learning) - as working in unison could lead to several interesting advances, as we believe we have demonstrated in our presented work.

References

[1] Aarts, F., Fiterau-Brostean, P., Kuppens, H., and Vaandrager, F. (2015). Learning register automata with fresh value generation. In *International Colloquium on Theoretical Aspects of Computing*, pages 165–183. Springer.

[2] Abadi, M., Blanchet, B., and Fournet, C. (2017). The applied pi calculus: Mobile values, new names, and secure communication. *Journal of the ACM (JACM)*, 65(1):1–41.

[3] Abdelhakim, M., Ren, J., and Li, T. (2014). Throughput analysis and routing security discussions of mobile access coordinated wireless sensor networks. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, pages 4616–4621. IEEE.

[4] Aberkane, S. and Elarbi, M. (2019). Deep reinforcement learning for real-world anomaly detection in surveillance videos. In *2019 6th International Conference on Image and Signal Processing and their Applications (ISPA)*, pages 1–5.

[5] Abhishek, N. V., Lim, T. J., Sikdar, B., and Tandon, A. (2018). An intrusion detection system for detecting compromised gateways in clustered iot networks. In *2018 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pages 1–6. IEEE.

[6] Ali, S., Sun, H., and Zhao, Y. (2018). Model learning: A survey on foundation, tools and applications. *arXiv preprint arXiv:1901.01910*.

[7] Alur, R. and Henzinger, T. A. (1999). Reactive modules. *Formal methods in system design*, 15(1):7–48.

[8] Amaral, J. P., Oliveira, L. M., Rodrigues, J. J., Han, G., and Shu, L. (2014). Policy and network-based intrusion detection system for ipv6-enabled wireless sensor networks. In *2014 IEEE International Conference on Communications (ICC)*, pages 1796–1801. IEEE.

[9] Amouri, A., Alaparthy, V. T., and Morgera, S. D. (2018). Cross layer-based intrusion detection based on network behavior for iot. In *2018 IEEE 19th Wireless and Microwave Technology Conference (WAMICON)*, pages 1–4. IEEE.

[10] Anand, P., Singh, Y., Selwal, A., Alazab, M., Tanwar, S., and Kumar, N. (2020). Iot vulnerability assessment for sustainable computing: Threats, current solutions, and open challenges. *IEEE Access*, 8:168825–168853.

[11] Andova, S. (1999). Process algebra with probabilistic choice. In *International AMAST Workshop on Aspects of Real-Time Systems and Concurrent and Distributed Software*, pages 111–129. Springer.

[12] Armando, A., Arsac, W., Avanesov, T., Barletta, M., Calvi, A., Cappai, A., Carbone, R., Chevalier, Y., Compagna, L., Cuéllar, J., et al. (2012). The avantssar platform for the automated validation of trust and security of service-oriented architectures. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 267–282. Springer.

[13] Armando, A., Basin, D., et al. (2005). The AVISPA tool for the Automated Validation of Internet Security Protocols and Applications. In *International conference on Computer Aided Verification*, pages 281–285. Springer.

[14] Arnaboldi, L., Czekster, R. M., Morisset, C., and Metere, R. (2020). Modelling load-changing attacks in cyber-physical systems. *Electronic Notes in Theoretical Computer Science*, 353:39–60.

[15] Arnaboldi, L. and Metere, R. (2019). Poster: Towards a data centric approach for the design and verification of cryptographic protocols. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2585–2587.

[16] Arnaboldi, L. and Morisset, C. (2017). Quantitative analysis of dos attacks and client puzzles in iot systems. In *International Workshop on Security and Trust Management*, pages 224–233. Springer.

[17] Arnaboldi, L. and Morisset, C. (2018a). Generating synthetic data for real world detection of dos attacks in the iot. In *Federation of International Conferences on Software Technologies: Applications and Foundations*, pages 130–145. Springer.

[18] Arnaboldi, L. and Morisset, C. (2018b). Lisa: Predicting the impact of dos attacks on real-world low power iot systems. *Foundations of Computer Security Workshop*.

[19] Arnaboldi, L. and Tschofenig, H. (2019). A Formal Model for Delegated Authorization of IoT Devices Using ACE-OAuth. In *OAuth Security Workshop*.

[20] Arolkar, H. A., Sheth, S. P., and Tamhane, V. P. (2011). Ant colony based approach for intrusion detection on cluster heads in wsn. In *ICCCS*, pages 523–526.

[21] Arrington, B., Barnett, L., Rufus, R., and Esterline, A. (2016). Behavioral modeling intrusion detection system (bmids) using internet of things (iot) behavior-based anomaly detection via immunity-inspired algorithms. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6. IEEE.

[22] Aslanyan, Z., Nielson, F., and Parker, D. (2016). Quantitative verification and synthesis of attack-defence scenarios. In *Proc. 29th IEEE Computer Security Foundations Symposium (CSF'16)*, pages 105–119. IEEE.

[23] Aura, T., Nikander, P., and Leiwo, J. (2000). Dos-resistant authentication with client puzzles. In *International workshop on security protocols*, pages 170–177. Springer.

[24] Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205.

[25] Aziz, B. (2016). A formal model and analysis of an iot protocol. *Ad Hoc Networks*, 36:49–57.

[26] Baier, C., Katoen, J.-P., and Larsen, K. G. (2008). *Principles of model checking*. MIT press.

[27] Basagiannis, S., Katsaros, P., Pombortsis, A., and Alexiou, N. (2009). Probabilistic model checking for the quantification of dos security threats. *Computers and Security*, 28(6):450 – 465.

[28] Baumann, H. and Sandmann, W. (2012). Markovian modeling and security measure analysis for networks under flooding dos attacks. In *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 298–302. IEEE.

[29] Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

[30] Besson, L. and Leleu, P. (2009). A distributed intrusion detection system for ad-hoc wireless sensor networks: the awissenet distributed intrusion detection system. In *2009 16th International Conference on Systems, Signals and Image Processing*, pages 1–3. IEEE.

[31] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2014). Network anomaly detection: methods, systems and tools. *IEEE communications surveys & tutorials*.

[32] Böhme, R. and Félegyházi, M. (2010). Optimal information security investment with penetration testing. In *International Conference on Decision and Game Theory for Security*, pages 21–37. Springer.

[33] Bohr, M. (2007). A 30 year retrospective on dennard's mosfet scaling paper. *IEEE Solid-State Circuits Society Newsletter*, 12(1):11–13.

[34] Bostani, H. and Sheikhan, M. (2017). Hybrid of anomaly-based and specification-based ids for internet of things using unsupervised opf based on mapreduce approach. *Computer Communications*, 98:52–71.

[35] Bray, R., Cid, D., and Hay, A. (2008). *OSSEC host-based intrusion detection guide*. Syngress.

[36] Büchi, J. R. (1990). On a decision method in restricted second order arithmetic. In *The Collected Works of J. Richard Büchi*, pages 425–435. Springer.

[37] Buennemeyer, T. K., Gora, M., Marchany, R. C., and Tront, J. G. (2007). Battery exhaustion attack detection with small handheld mobile computers. In *Portable Information Devices*.

[38] Buennemeyer, T. K., Nelson, T. M., Clagett, L. M., Dunning, J. P., Marchany, R. C., and Tront, J. G. (2008). Mobile device profiling and intrusion detection using smart batteries. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pages 296–296. IEEE.

[39] Butun, I., Morgera, S. D., and Sankar, R. (2014). A survey of intrusion detection systems in wireless sensor networks. *IEEE communications surveys & tutorials*, 16(1):266–282.

[40] Cardenas, D. J. S. and Hahn, A. (2019). Iot threats to the smart grid: A framework for analyzing emerging risks. In *Proceedings of the Northwest Cybersecurity Symposium*, pages 1–8.

[41] Carl, G., Kesidis, G., Brooks, R. R., and Rai, S. (2006). Denial-of-service attack-detection techniques. *IEEE Internet computing*, 10(1):82–89.

[42] Castro, L. N., De Castro, L. N., and Timmis, J. (2002). *Artificial immune systems: a new computational intelligence approach*. Springer Science & Business Media.

[43] Cervantes, C., Poplade, D., Nogueira, M., and Santos, A. (2015). Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 606–611. IEEE.

[44] Cheng, T.-H., Lin, Y.-D., Lai, Y.-C., and Lin, P.-C. (2011). Evasion techniques: Sneaking through your intrusion detection/prevention systems. *IEEE Communications Surveys & Tutorials*, 14(4):1011–1020.

[45] Cho, E. J., Kim, J. H., and Hong, C. S. (2009). Attack model and detection scheme for botnet on 6lowpan. In *Asia-Pacific Network Operations and Management Symposium*, pages 515–518. Springer.

[46] Chormunge, S. and Jena, S. (2015). Efficiency and effectiveness of clustering algorithms for high dimensional data. *International Journal of Computer Applications*, 125(11).

[47] Clark, J., DeRose, S., et al. (1999). Xml path language (xpath) version 1.0.

[48] Clarke, E. M., Klieber, W., Nováček, M., and Zuliani, P. (2011). Model checking and the state explosion problem. In *LASER Summer School on Software Engineering*, pages 1–30. Springer.

[49] Coppolino, L., DAntonio, S., Garofalo, A., and Romano, L. (2013). Applying data mining techniques to intrusion detection in wireless sensor networks. In *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 247–254. IEEE.

[50] Craenen, B. and Eiben, A. (2002). Computational intelligence. encyclopedia of life support sciences. *EOLSS, EOLSS Co. Ltd*.

[51] Cremers, C. and Horvat, M. (2016). Improving the iso/iec 11770 standard for key management techniques. *International Journal of Information Security*, 15(6):659–673.

[52] Dabrowski, A., Ullrich, J., and Weippl, E. R. (2017). Grid shock: Coordinated load-changing attacks on power grids: The non-smart power grid is vulnerable to cyber attacks as well. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 303–314.

[53] Dabrowski, A., Ullrich, J., and Weippl, E. R. (2018). Botnets causing blackouts: how coordinated load attacks can destabilize the power grid. *e & i Elektrotechnik und Informationstechnik*, 135(3):250–255.

[54] Damghani, H., Hosseinian, H., and Damghani, L. (2019). Cryptography review in iot. In *2019 4th Conference on Technology In Electrical and Computer Engineering (ETECH2019)*.

[55] Danda, J. M. R. and Hota, C. (2016). Attack identification framework for iot devices. In *Information Systems Design and Intelligent Applications*, pages 505–513. Springer.

[56] Deng, L., Li, D., Yao, X., Cox, D., and Wang, H. (2018). Mobile network intrusion detection for iot system based on transfer learning algorithm. *Cluster Computing*, pages 1–16.

[57] Dennard, R. H., Gaensslen, F. H., Yu, H., Rideout, V. L., Bassous, E., and LeBlanc, A. R. (1974). Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268.

[58] Di Pietro, R. and Mancini, L. V. (2008). *Intrusion detection systems*, volume 38. Springer Science & Business Media.

[59] Donatelli, S. (1994). Superposed generalized stochastic petri nets: definition and efficient solution. In *International Conference on Application and Theory of Petri Nets*, pages 258–277. Springer.

[60] Echeverria, S., Seitz, L., Klinedinst, D., and Lewis, G. (2019). ACE Clients in Disadvantaged Networks. Internet-Draft draft-secheverria-ace-client-disadvantaged-00, Internet Engineering Task Force. Work in Progress.

[61] Elrawy, M. F., Awad, A. I., and Hamed, H. F. (2018). Intrusion detection systems for iot-based smart environments: a survey. *Journal of Cloud Computing*, 7(1):21.

[62] Esquivel-Vargas, H., Caselli, M., and Peter, A. (2017). Automatic deployment of specification-based intrusion detection in the bacnet protocol. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 25–36.

[63] Etalle, S. (2017). From intrusion detection to software design. In *European Symposium on Research in Computer Security*, pages 1–10. Springer.

[64] Etalle, S. (2019). Network monitoring of industrial control systems: The lessons of securitymatters. In *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, pages 1–1.

[65] Farnell (2017). Farnell element14, calculating battery life in iot applications (2017).

[66] Feng, W., Kaiser, E., and Luu, A. (2005). Design and implementation of network puzzles. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 4, pages 2372–2382 vol. 4.

[67] Fruth, M. (2011). *Formal methods for the analysis of wireless network protocols*. Oxford University.

[68] Fu, Y., Yan, Z., Cao, J., Koné, O., and Cao, X. (2017). An automata based intrusion detection method for internet of things. *Mobile Information Systems*, 2017.

[69] Garcia-Font, V., Garrigues, C., and Rifà-Pous, H. (2017). Attack classification schema for smart city wsns. *Sensors*, 17(4):771.

[70] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28.

[71] Grieco, S. S. A. R. L. and Coen-Porisini, A. (2015). Security, privacy and trust in internet of things: The road ahead. *Computer Networks*.

[72] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*.

[73] Guillen, E., Sánchez, J., and Paez, R. (2015). Inefficiency of ids static anomaly detectors in real-world networks. *Future Internet*, 7(2):94–109.

[74] Gupta, A., Pandey, O. J., Shukla, M., Dadhich, A., Mathur, S., and Ingle, A. (2013). Computational intelligence based intrusion detection systems for wireless communication and pervasive computing networks. In *2013 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–7. IEEE.

[75] Haataja, K. M. (2008). New efficient intrusion detection and prevention system for bluetooth networks. In *Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, page 16. ICST (Institute for Computer Sciences, Social-Informatics and ....

[76] Hadžiosmanović, D., Sommer, R., Zambon, E., and Hartel, P. H. (2014). Through the eye of the plc: semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 126–135. ACM.

[77] Han, G., Jiang, J., Shen, W., Shu, L., and Rodrigues, J. (2013). Idsep: a novel intrusion detection scheme based on energy prediction in cluster-based wireless sensor networks. *IET Information Security*, 7(2):97–105.

[78] Hao, Feng and Metere, Roberto and Shahandashti, Siamak F and Dong, Changyu (2018). Analyzing and patching speke in iso/iec. *IEEE Transactions on Information Forensics and Security*, 13(11):2844–2855.

[79] Hassanzadeh, A. and Stoleru, R. (2011). Towards optimal monitoring in cooperative ids for resource constrained wireless networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–8. IEEE.

[80] Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., and Wolber, D. (1990). A network security monitor. In *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 296–304. IEEE.

[81] Hillston, J. (2005). *A compositional approach to performance modelling*, volume 12. Cambridge University Press.

[82] Hoare, C. A. R. (1978). Communicating sequential processes. *Communications of the ACM*, 21(8):666–677.

[83] Hui Suo Jiafu Wan, C. Z. J. L. (2014). Security in the internet of things: A review. *International Journal of Computer Applications*.

[84] Hummen, R., Wirtz, H., Ziegeldorf, J. H., Hiller, J., and Wehrle, K. (2013). Tailoring end-to-end ip security protocols to the internet of things. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–10.

[85] Iera, A. L. A. and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*.

[86] Jiang, T., Wang, G., and Yu, H. (2012). A dynamic intrusion detection scheme for cluster-based wireless sensor networks. In *World Automation Congress 2012*, pages 259–261. IEEE.

[87] Johanna, V., Liquan, C., Yao, Z., and Yuewei, M. (2011). Challenges in qualitative accelerated testing of wsn hardware. *Engineering*, 2011.

[88] Juels, A. and Brainard, J. G. (1999). Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*, volume 99, pages 151–165.

[89] Kai Zhao, L. G. (2013). A survey on the internet of things security. *Computational Intelligence and Security (CIS)*.

[90] Kasinathan, P., Costamagna, G., Khaleel, H., Pastrone, C., and Spirito, M. A. (2013a). An ids framework for internet of things empowered by 6lowpan. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1337–1340. ACM.

[91] Kasinathan, P., Pastrone, C., Spirito, M. A., and Vinkovits, M. (2013b). Denial-of-service detection in 6lowpan based internet of things. In *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*, pages 600–607. IEEE.

[92] Khan, Z. A. and Herrmann, P. (2017). A trust based distributed intrusion detection mechanism for internet of things. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 1169–1176. IEEE.

[93] Kim, S. W. (2015). Physical integrity check in cooperative relay communications. *IEEE Transactions on Wireless Communications*, 14(11):6401–6413.

[94] Knuth, D. and Yao, A. (1976). *Algorithms and Complexity: New Directions and Recent Results*, chapter The complexity of nonuniform random number generation. Academic Press.

[95] Kobeissi, Nadim and Bhargavan, Karthikeyan and Blanchet, Bruno (2017). Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *IEEE EuroS&P*.

[96] Krimmling, J. and Peter, S. (2014). Integration and evaluation of intrusion detection for coap in smart city applications. In *2014 IEEE Conference on Communications and Network Security*, pages 73–78. IEEE.

[97] Kruegel, C. and Vigna, G. (2003). Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 251–261. ACM.

[98] Kwiatkowska, M., Norman, G., and Parker, D. (2000). Verifying randomized distributed algorithms with prism. In *Workshop on advances in verification (WAVE'00)*.

[99] Kwiatkowska, M., Norman, G., and Parker, D. (2002). Prism: Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*.

[100] Kwiatkowska, M., Norman, G., and Parker, D. (2004). Probabilistic symbolic model checking with prism: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142.

[101] Kwiatkowska, M., Norman, G., and Parker, D. (2011a). Prism 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification*, pages 585–591. Springer.

[102] Kwiatkowska, M., Norman, G., and Parker, D. (2011b). PRISM 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan, G. and Qadeer, S., editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer.

[103] Lantz, B. and O'Connor, B. (2015). A mininet-based virtual testbed for distributed sdn development. *ACM SIGCOMM Computer Communication Review*, 45(4):365–366.

[104] Le, A., Loo, J., Chai, K., and Aiash, M. (2016). A specification-based ids for detecting attacks on rpl-based network topology. *Information*, 7(2):25.

[105] Le, A., Loo, J., Luo, Y., and Lasebae, A. (2011). Specification-based ids for securing rpl from topology attacks. In *2011 IFIP Wireless Days (WD)*, pages 1–3. IEEE.

[106] Lee, E. A. (2008). Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE.

[107] Lee, T.-H., Wen, C.-H., Chang, L.-H., Chiang, H.-S., and Hsieh, M.-C. (2014). A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan. In *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, pages 1205–1213. Springer.

[108] Leloglu, E. (2016). A review of security concerns in internet of things. *Journal of Computer and Communications*, 5(1):121–136.

[109] Liang, L., Zheng, K., Sheng, Q., and Huang, X. (2016). A denial of service attack method for an iot system. In *Information Technology in Medicine and Education (ITME), 2016 8th International Conference on*, pages 360–364. IEEE.

[110] Lin, W.-C., Ke, S.-W., and Tsai, C.-F. (2015). Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, 78:13–21.

[111] Liu, C., Yang, J., Chen, R., Zhang, Y., and Zeng, J. (2011). Research on immunity-based intrusion detection technology for the internet of things. In *2011 Seventh International Conference on Natural Computation*, volume 1, pages 212–216. IEEE.

[112] Liu, L., Xu, B., Zhang, X., and Wu, X. (2018). An intrusion detection method for internet of things based on suppressed fuzzy clustering. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):113.

[113] Liu, Y. and Yu, F. (2008). Immunity-based intrusion detection for wireless sensor networks. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 439–444. IEEE.

[114] Long, N. and Thomas, R. (2001). Trends in denial of service attack technology. *CERT Coordination Center*.

[115] Lundin, E. and Jonsson, E. (2000). Anomaly-based intrusion detection: privacy concerns and other problems. *Computer networks*, 34(4):623–640.

[116] Luo, T. and Nagarajan, S. G. (2018). Distributed anomaly detection using autoencoder neural networks in wsn for iot. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

[117] Matsunaga, T., Toyoda, K., and Sasase, I. (2014). Low false alarm rate rpl network monitoring system by considering timing inconstancy between the rank measurements. In *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 427–431. IEEE.

[118] Meier, S. (2013). *Advancing automated security protocol verification*. PhD thesis, ETH Zurich.

[119] Meier, S., Schmidt, B., Cremers, C., and Basin, D. (2013). The tamarin prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 696–701. Springer.

[120] Mell, P., Hu, V., Lippmann, R., Haines, J., and Zissman, M. (2003). An overview of issues in testing intrusion detection systems.

[121] Midi, D., Rullo, A., Mudgerikar, A., and Bertino, E. (2017). Kalis—a system for knowledge-driven adaptable intrusion detection for the internet of things. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 656–666. IEEE.

[122] Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., and Payne, B. D. (2015). Evaluating computer intrusion detection systems: A survey of common practices. *ACM Computing Surveys (CSUR)*, 48(1):12.

[123] Minakov, I., Passerone, R., Rizzardi, A., and Sicari, S. (2016). A comparative study of recent wireless sensor network simulators. *ACM Transactions on Sensor Networks (TOSN)*, 12(3):1–39.

[124] Mirkovic, J., Dietrich, S., Dittrich, D., and Reiher, P. (2004). *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[125] Misra, S., Krishna, P. V., Agarwal, H., Saxena, A., and Obaidat, M. S. (2011). A learning automata based solution for preventing distributed denial of service in internet of things. In *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pages 114–122. IEEE.

[126] Molloy, M. K. (1982). Performance analysis using stochastic petri nets. *IEEE Transactions on computers*, 9:913–917.

[127] Moya, C. and Wang, J. (2018). Developing correlation indices to identify coordinated cyber-attacks on power grids. *IET Cyber-Physical Systems: Theory & Applications*, 3(4):178–186.

[128] Moyers, B. R., Dunning, J. P., Marchany, R. C., and Tront, J. G. (2010). The multi-vector portable intrusion detection system (mvp-ids): a hybrid approach to intrusion detection for portable information devices. In *2010 IEEE International Conference on Wireless Information Technology and Systems*, pages 1–4. IEEE.

[129] Najafabadi, M. M., Khoshgoftaar, T. M., Napolitano, A., and Wheelus, C. (2016). Rudy attack: Detection at the network level and its important features. In *The twenty-ninth international flairs conference*.

[130] Nemati, M., Braun, M., and Tenbohlen, S. (2018). Optimization of unit commitment and economic dispatch in microgrids based on genetic algorithm and mixed integer linear programming. *Applied energy*, 210:944–963.

[131] Nimal, V. (2010). *Statistical approaches for probabilistic model checking*. PhD thesis, University of Oxford.

[132] Norman, G., Parker, D., Kwiatkowska, M., Shukla, S., and Gupta, R. (2005). Using probabilistic model checking for dynamic power management. *Formal aspects of computing*, 17(2):160–176.

[133] Norris, J. R. and Norris, J. R. (1998). *Markov chains*, volume 2. Cambridge university press.

[134] Novakovic, C. and Parker, D. (2019). Automated formal analysis of side-channel attacks on probabilistic systems. In *Proc. 24th European Symposium on Research in Computer Security (ESORICS'19)*, volume 11735 of *LNCS*, pages 319–337. Springer.

[135] OConnor, T. and Reeves, D. (2008). Bluetooth network-based misuse detection. In *2008 Annual Computer Security Applications Conference (ACSAC)*, pages 377–391. IEEE.

[136] Oh, D., Kim, D., and Ro, W. (2014). A malicious pattern detection engine for embedded security systems in the internet of things. *Sensors*, 14(12):24188–24211.

[137] Padhy, N. P. (2004). Unit commitment-a bibliographical survey. *IEEE Transactions on power systems*, 19(2):1196–1205.

[138] Papavasiliou, A., Mou, Y., Cambier, L., and Scieur, D. (2017). Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty. *IEEE Transactions on Sustainable Energy*, 9(2):547–558.

[139] Park, J., Iwai, K., Tanaka, H., and Kurokawa, T. (2014). Analysis of slow read dos attack and countermeasures. In *Proceeding of the International Conference on Cyber-Crime Investigation and Cyber Security*, pages 37–49.

[140] Patcha, A. and Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470.

[141] Patel, J. S. K. D. R. (2014). A survey on internet of things: Security and privacy issues. *International Journal of Computer Applications*.

[142] Paxson, V. (1999). Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463.

[143] Plateau, B. and Atif, K. (1991). Stochastic automata network of modeling parallel systems. *IEEE transactions on software engineering*, 10:1093–1108.

[144] Pongle, P. and Chavan, G. (2015). Real time intrusion and wormhole attack detection in internet of things. *International Journal of Computer Applications*, 121(9).

[145] Qi Jing Athanasios V. Vasilakos, J. W. J. L. D. Q. (2014). Security of the internet of things: perspectives and challenges. *Wireless Networks*.

[146] Rajkumar, R., Lee, I., Sha, L., and Stankovic, J. (2010). Cyber-physical systems: the next computing revolution. In *Design Automation Conference*, pages 731–736. IEEE.

[147] Rathi, A. K. and Santiago, A. J. (1990). The new netsim simulation program. *Traffic engineering & control*, 31(5).

[148] Raza, S., Wallgren, L., and Voigt, T. (2013). Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, 11(8):2661–2674.

[149] Roesch, M. et al. (1999). Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238.

[150] Roman, R., Zhou, J., and Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*.

[151] Sabhnani, M. and Serpen, G. (2004). Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. *Intelligent data analysis*, 8(4):403–415.

[152] Safavian, S. R. and Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674.

[153] Scarfone, K. and Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94.

[154] Sedjelmaci, H. and Senouci, S. M. (2013). Efficient and lightweight intrusion detection based on nodes' behaviors in wireless sensor networks. In *Global Information Infrastructure Symposium-GIIS 2013*, pages 1–6. IEEE.

[155] Sedjelmaci, H., Senouci, S. M., and Al-Bahri, M. (2016). A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

[156] Sendorek, J., Szydlo, T., and Brzoza-Woch, R. (2018). Software-defined virtual testbed for iot systems. *Wireless Communications and Mobile Computing*, 2018.

[157] Shreenivas, D., Raza, S., and Voigt, T. (2017). Intrusion detection in the rpl-connected 6lowpan networks. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 31–38. ACM.

[158] Simopoulos, D. N., Kavatza, S. D., and Vournas, C. D. (2006). Unit commitment by an enhanced simulated annealing algorithm. *IEEE Transactions on Power Systems*, 21(1):68–76.

[159] Snapp, S. R., Brentano, J., Dias, G. V., Goan, T. L., Heberlein, L. T., Ho, C.-l., Levitt, K. N., Mukherjee, B., Smaha, S. E., Grance, T., et al. (1991). Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype. In *In Proceedings of the 14th National Computer Security Conference*. Citeseer.

[160] Soltan, S., Mittal, P., and Poor, H. V. (2018). Blackiot: Iot botnet of high wattage devices can disrupt the power grid. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 15–32.

[161] Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE.

[162] Song, X., Chen, G., and Li, X. (2010). A weak hidden markov model based intrusion detection method for wireless sensor networks. In *2010 International Conference on Intelligent Computing and Integrated Systems*, pages 887–889. IEEE.

[163] Stelte, B. and Rodosek, G. D. (2013). Thwarting attacks on zigbee-removal of the killerbee stinger. In *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pages 219–226. IEEE.

[164] Stewart, W. J. (2009). *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling*. Princeton university press.

[165] Summerville, D. H., Zach, K. M., and Chen, Y. (2015). Ultra-lightweight deep packet anomaly detection for internet of things devices. In *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE.

[166] Sun, C.-C., Hong, J., and Liu, C.-C. (2016). A coordinated cyber attack detection system (ccads) for multiple substations. In *2016 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE.

[167] Suo, H., Wan, J., Zou, C., and Liu, J. (2012). Security in the internet of things: a review. In *Computer Science and Electronics Engineering (ICCSEE), 2012 international conference on*, volume 3, pages 648–651. IEEE.

[168] Swamy, N., Hriţcu, C., Keller, C., Rastogi, A., Delignat-Lavaud, A., Forest, S., Bhargavan, K., Fournet, C., Strub, P.-Y., Kohlweiss, M., et al. (2016). Dependent types and multi-monadic effects in f. In *ACM SIGPLAN Notices*, volume 51, pages 256–270. ACM.

[169] Talpade, R., Madhani, S., Mouchtaris, P., and Wong, L. (2003). Mitigating denial of service attacks. US Patent App. 10/353,527.

[170] Tan, K. M., Killourhy, K. S., and Maxion, R. A. (2002). Undermining an anomaly-based intrusion detection system using common exploits. In *International Workshop on Recent Advances in Intrusion Detection*, pages 54–73. Springer.

[171] Thanigaivelan, N. K., Nigussie, E., Kanth, R. K., Virtanen, S., and Isoaho, J. (2016). Distributed internal anomaly detection system for internet-of-things. In *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 319–320. IEEE.

[172] Thompson, H. S., Mendelsohn, N., Beech, D., and Maloney, M. (2009). W3c xml schema definition language (xsd) 1.1 part 1: Structures. *The World Wide Web Consortium (W3C), W3C Working Draft Dec*, 3.

[173] Ting, T., Rao, M., and Loo, C. (2006). A novel approach for unit commitment problem via an effective hybrid particle swarm optimization. *IEEE Transactions on power systems*, 21(1):411–418.

[174] Tritilanunt, S., Boyd, C., Foo, E., and Nieto, J. (2006). Examining the dos resistance of hip. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 616–625. Springer.

[175] Trivedi, A., Srinivasan, D., Biswas, S., and Reindl, T. (2016). A genetic algorithm–differential evolution based hybrid framework: case study on unit commitment scheduling problem. *Information Sciences*, 354:275–300.

[176] Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M., and Fischer, M. (2015). Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys (CSUR)*, 47(4):1–33.

[177] Vattakunnel, A. J., Kumar, N. S., and Kumar, G. S. (2016). Modelling and verification of coap over routing layer using spin model checker. *Procedia Computer Science*, 100(93):299–308.

[178] Vora, P., Oza, B., et al. (2013). A survey on k-mean clustering and particle swarm optimization. *International Journal of Science and Modern Engineering*, 1(3):1–14.

[179] Wallgren, L., Raza, S., and Voigt, T. (2013). Routing attacks and countermeasures in the rpl-based internet of things. *International Journal of Distributed Sensor Networks*, 9(8):794326.

[180] Wang, Q. and Megalooikonomou, V. (2005). A clustering algorithm for intrusion detection. In *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005*, volume 5812, pages 31–39. International Society for Optics and Photonics.

[181] Weaver, R., Weaver, D., and Farwood, D. (2013). *Guide to network defense and countermeasures*. Cengage Learning.

[182] Wendt, T. X. J. B. and Potkonjak, M. (2015). Security of iot systems: Design challenges and opportunities. *Informational System Frontiers*.

[183] Winkler, J. (1990). A unix prototype for intrusion and anomaly detection in secure networks. In *Proceedings of the 13th National Computer Security Conference*, pages 115–124.

[184] Wu, S. X. and Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied soft computing*, 10(1):1–35.

[185] Xu, B., Chen, M., Xing, C., and Zhang, G. (2009). A network traffic identification method based on finite state machine. In *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE.

[186] Yadav, K. and Srinivasan, A. (2010). itrust: an integrated trust framework for wireless sensor networks. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1466–1471. ACM.

[187] Yu, Z. and Tsai, J. J. (2008). A framework of machine learning based intrusion detection for wireless sensor networks. In *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*, pages 272–279. IEEE.

[188] Zarpelao, B. B., Miani, R. S., Kawakani, C. T., and de Alvarenga, S. C. (2017). A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, 84:25–37.

[189] Zeng, X., Garg, S. K., Strazdins, P., Jayaraman, P. P., Georgakopoulos, D., and Ranjan, R. (2017). Iotsim: A simulator for analysing iot applications. *Journal of Systems Architecture*, 72:93–107.

[190] Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462.

[191] Zhang, Y., Wang, L., Sun, W., Green, R. C., and Alam, M. (2011). Artificial immune system based intrusion detection in a distributed hierarchical network architecture of smart grid. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–8. IEEE.

[192] Zheng, A. and Casari, A. (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.".

[193] Zurada, J., Marks, R., and Robinson, J. (1995). Review of computational intelligence: imitating life.

Appendix A

Summary of Intrusion Detection Tools

In Chap. 3 a overview of trends in IoT IDSs in the form of heatmaps. Further details about specific tools for different types of IDSs are provided in this Appendix. Three tables are provided for the different types of IDSs, Network IDS, Host IDS, and COllaborative IDS. Each table is broken down into **Paper**, **Placement**, **Algorithm**, **Environment**, **Evaluation** and **Detection**.

Table A.1 Summary of Network Intrusion Detection Systems Proposed for IoT

Article	Placement	Algorithm	Environment	Evaluation	Detection
Arrington et al. [21]	central	immunity	smart-home	○	internal[2]
Besson et al. [30]	distributed	-	WSN	X	-
Cho et al. [45]	central	statistical	6LowPAN	○	DDoS
Danda et al. [55]	central	rule-based	testbed	X	-
Deng et al [56]	-	clustering	WSN	○, δ	Sybil
Le et al. [105]	distributed	specification	RPL	X	topology
Liu et al [112]	-	clustering	IoT	Σ	-
Stelte et al. [163]	distributed	anomaly	ZigBee	○	KillerBee
Hadžiosmanović et al. [76]	-	anomaly	ICS	Σ, δ	-
Sedjelmaci at al. [154]	central	hybrid	WSN	○	routing[3]
Sedjelmaci et al. [155]	central	game-theory	IoT	○	DoS
Haataja et al. [75]	central	rule-based	Bluetooth	X	β[4]
Yadav et al. [186]	distributed	trust-based	WSN	○	β[5]
OConnor et al. [135]	central	rule-based	Bluetooth	✓	β[6]
Thanigaivelan et al. [171]	distributed	anomaly	RPL	X	internal[2]
Luo et al. [116]	distributed	anomaly	WSN	δ	internal[2]
Wallgren et al. [179]	distributed	rule-based	RPL	○	routing[3]
Matsanaga et al. [117]	distributed	statistical	RPL	○	routing[3]
Cervantes et al. [43]	distributed	trust	6LowPAN	○	sinkhole
Misra et al. [125]	centralised	automaton	IoT	✓	DDoS
Garcia-Font et al. [69]	central	hybrid	smart-city	○	routing
Fu et al. [68]	central	automata	http	X	protocol[1]
Khan et al. [92]	distributed	trust-based	healthcare	○	routing[3]
Han et al. [77]	central	anomaly	WSN	Σ, ○	energy-DoS
Jiang et al. [86]	central	clustering	clustered	X	routing[3]
Esquivel-Vargas et al. [62]	central	specification	BACnet	δ, ✓	protocol[1]

Evaluation	X - no evaluation performed Σ - mathematical evaluation performed ○ - simulation performed δ - trace evaluation performed ✓ - execution evaluation performed
Detection	β - custom attack list [1]MITM, replay, spoofing, message dropping etc. [2]unusual behaviour by an agent within the system [3]routing attacks, e.g. Sinkhole, Wormhole, Blackhole [4]resource drain DoS, pin cracking, and spoofing [5]collision attack, Hello Flood, Selective forwarding attack [6]reconnaissance, DoS, and information theft attacks

Table A.2 Summary of Host Intrusion Detection Systems for IoT

Article	Algorithm	Environment	Evaluation	Detection
Oh et al. [136]	pattern-matching	IoT	Σ, δ	DDoS, virus signatures
Summerville et al. [165]	bit-wise anomaly	IoT	✓	worm, code injection
Liu et al. [111]	immunity	IoT	Σ	-
Kim [93]	likeliood ratio	relay coms	Σ	message integrity
Lee et al. [107]	anomaly	6LowPAN	○	battery DoS
Song et al. [162]	W-HMM	WSN	○	-

Evaluation	X - no evaluation performed Σ - mathematical evaluation performed ○ - simulation performed δ - trace evaluation performed ✓ - execution evaluation performed

Table A.3 Summary of Collaborative Intrusion Detection Systems for IoT

Article	Placement	Algorithm	Environment	Evaluation	Detection
Gupta et al. [74]	distributed	CI (multi)	wireless	X	protocol[1]
Amaral et al. [8]	distributed	rule-based	IPv6	X	-
Kasinathan et al. [91]	hybrid	rule-based	6LoWPAN	✓	DoS
Kasinathan et al. [90]	hybrid	rule-based	6LowPAN	✓	DoS
Amouri et al. [9]	cross-layer	anomaly	IoT	○	-
Hassanzadeh et al. [79]	distributed	genetic-algo.	IoT	○,✓	β[1]
Le et al. [104]	hybrid	specification	RPL	○	topology[2]
Midi et al. [121]	centralised	adaptable	IoT	✓	DoS
Shreenivas et al. [157]	hybrid	statistical	RPL	○	insider
Bostani et al. [34]	distributed	hybrid	WSN	○,✓	routing
Arolkar et al. [20]	distributed	ant-colony	WSN	X	β[3]
Coppolino et al. [49]	hybrid	hybrid	WSN	δ	β[4]
Abhishek et al. [5]	distributed	anomaly	IoT	○	β[5]
Pongle et al. [144]	distributed	anomaly	RPL	○	wormhole
Zhang et al. [191]	distributed	immunity	smart-grid	δ,○	β[6]
Yu et al. [187]	mixed	hybrid	WSN	X	routing
Buennemeyer et al. [38]	distributed	hybrid	IoT	✓,α	battery-DoS
Moyers et al. [128]	distributed	hybrid	IoT	X	-
Raza et al. [148]	distributed	statistical	IPv6	○	routing[3]

Evaluation	X - no evaluation performed Σ - mathematical evaluation performed ○ - simulation performed δ - trace evaluation performed ✓ - execution evaluation performed α - usability evaluated
Detection	β - custom attack list [1]Flooding, Port Scanning, Web Exploits [2]Sinkhole, Rank, Local repair, DIS, Neighbour [3]Sinkhole, Misdirection, passive information gathering [4]Sinkhole, sleep exhaustion [5]Compromised Gateways [6]MITM, replay, spoofing, message dropping, DoS, data leakage

Appendix B

Survey IDS Request Template Letter

Dear <INSERT AUTHOR LIST>,

I am a PhD Student working on IoT Security at Newcastle University (My Info) and I am currently working on developing an IoT security testbed. My work hopes to test out various security solutions in a range of test IoT environments in the form of embedded virtual machines hosted in various network configurations. I am contacting you in regards to your work <PAPER TITLE HERE>. <INSERT UNIQUE COMMENT ON PAPER>. I was hoping you could provide me with an implementation of your work to use as I was unable to find a publicly available repository?

Please let me know how you would prefer I cite your work in the case of publication and what license restrictions might be applicable.

If you have any questions about my work or how your implementation will be used please don't hesitate to ask.

Yours sincerely,
Luca Arnaboldi

Appendix C

Survey IDS Request Example Letter

Dear ~~*REDACTED*~~,

I am a PhD Student working on IoT Security at Newcastle University (My Info) and I am currently working on developing an IoT security testbed. My work hopes to test out various security solutions in a range of test IoT environments in the form of embedded virtual machines hosted in various network configurations. I am contacting you in regards to your work ~~*REDACTED*~~. *Your presented detection methodology is very powerful showing some excellent accuracy with low false positives, I would be interested in seeing how well it would scale in larger systems and different configurations.* I was hoping you could provide me with an implementation of your work to use as I was unable to find a publicly available repository?

Please let me know how you would prefer I cite your work in the case of publication and what license restrictions might be applicable.

If you have any questions about my work or how your implementation will be used please don't hesitate to ask.

Yours sincerely,
Luca Arnaboldi

Appendix D

Simple DoS Model on Smart Grid Power Generators

We show the energy supply trade off balance in a toy example with two scenarios, Scenario A and Scenario B in Fig. D.1, whose behaviour is as expected. In this case study we show the impact of two different factors, responsiveness to the demand and magnitude of the demand, each placed in a low demand scenario and a high demand scenario. Both scenarios are initially capable of meeting the demand, as there are plenty of available PGs; however, they react differently to the load caused by botnets' attack.

Table D.1 How the PG's interact with the attacker in Scenario A-(1/2). The States are: **A** - *Available*, **S** - *Serving*, and **D** - *Disconnected*. *Sup*, is the supply of a single PG out of four and *Att* is whether the Spike Botnet is on or not. The *Demand* is fixed at 120 units. The values represent the rate at which a state transitions from a state to another (if 0 the transition doesn't exist).

Slow	Transitions			Sup	Att
	A	S	D		
A	–	0.50	0	0	Off
S	0.50	–	0	50	Off
D	0.25	0	–	0	Off
A	–	0.50	1200	0	On
S	0.50	–	1200	50	On
D	0.25	0	–	0	On

Fast	Transitions			Sup	Att
	A	S	D		
A	–	1200	0	0	Off
S	1200	–	0	50	Off
D	600	0	–	0	Off
A	–	1200	120000	0	On
S	1200	–	120000	50	On
D	600	0	–	0	On

In Scenario A, we model two systems A-1 and A-2 under a high load. A-1 has very responsive PGs (such as gas plants), and A-2 has less responsive PGs. We showcase that in the system with faster response rate the time in which the system is in over demand is much lower. On the other hand, due to slow startup times, in the second system the time in over demand will be much higher.

In Scenario B, we model two more systems, B-1 and B-2, each of the systems shows the same number of PGs, but while the demand of B-1 is high, the demand of B-2 is low. We show that when the system is in a high demand period, an attacker is much more likely to disrupt the powergrid than if under a low demand period. The scenario matches the values for Scenario A-1, but the demand is altered between B-1 and B-2 from 50 to 100. This case highlights the different situations that can take place if a Spike botnet were to target the power grid, in our in depth experiments mimicking the power usage of real world scenario we take this a step further and examine it on various different types of PG such as gas and solar plants.

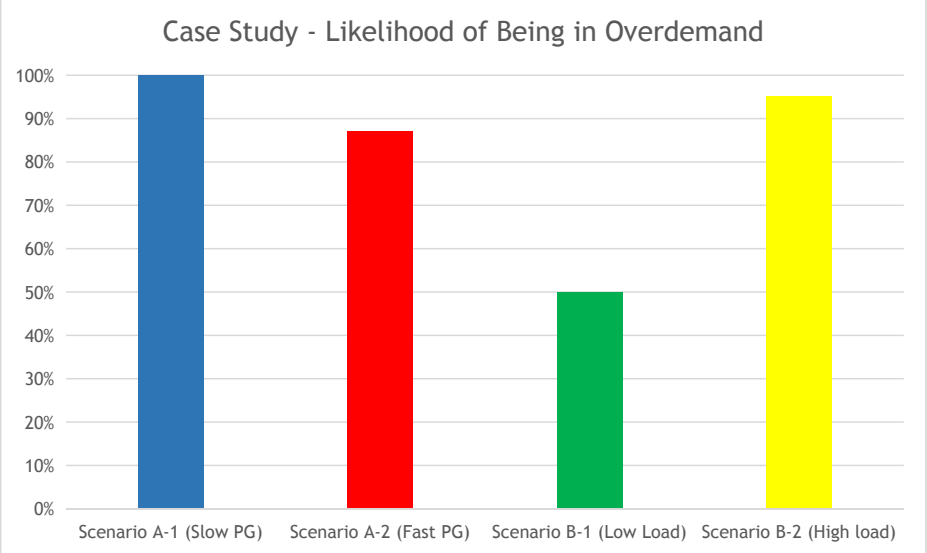


Fig. D.1 Powergrid reaction to botnet spikes in our case study scenarios

Appendix E

Anomaly Based Harm Detection

E.1 Appendix Summary

Security becomes ever more critical in the Internet of Things (IoT), as restrained resources, heterogeneity and sheer scale of these systems make them very difficult to defend. As the devices themselves often have no means to adopt many of the more standard mitigation techniques common in the web, the defence of these systems is often offloaded to a Intrusion Detection System (IDS). However, the deployment of intrusion detection onto these systems is not without complications as the unique nature of these systems presents new and unique challenges. The IoT landscape has evolved many new attack vectors and he current defence mechanisms struggle to prevent them all, to this end we propose a methodology that can work stand alone or alongside current systems to improve the attack detection rate. Our approach leverages the fact that over 60% of surveyed attacks on IoT systems aim to target certain constraints such as limited resource or dynamic routing. Based on this information we propose making use of convex optimisation theory to improve IDS detection that is able to detect any such attack. Our claim for white-box knowledge comes from the fact that through this methodology we are easily able to understand causes for anomaly, act on the alert and learn to adapt from new scenarios.

E.2 Introduction

One key thing to note is that IoT devices function very differently than the average Internet connected device such as a laptop. A key thing we consider in this paper is that some IoT devices are powered by battery and therefore could go offline at any moment. These new challenges are something current technologies and various protocols have not been built to be able to deal with [182]. These issues cause for there to be security flaws in the current implementations that can be easily exploited. A device can sometimes be hampered by the task of actually securing itself (due to resource constraints of performing techniques such as strong encryption), as has been investigated in work looking at existing solutions implemented in the IoT [17, 145]. Offloading computation away from the devices themselves can therefore be a very desirable property.

One IoT environment we focus on is IoT in a disadvantaged network [60]. In these kinds of networks there are even greater constraints in terms of communication bandwidth. Nodes in disadvantaged networks operate in what are called DIL environments (disconnected, intermittent, limited), which means that there is limited and unreliable connectivity between nodes with potentially periods of full disconnection. One approach that can ensure security without affecting the device resources is an network based Intrusion Detection System (IDS). An IDS is a monitor placed on the network that analyses incoming messages to detect attacks and/or unwanted traffic, and when paired with an intrusion prevention system can stop attacks before they affect the devices.

Standard approaches used to train IDSs include using a database of known attacks (*misuse detection*) and testing systems to create a “benchmark" behaviour and flag any anomaly as a potential attack (*anomaly detection*) [120]. An anomaly based IDS will monitor the normal behaviour within an IoT system and flag up any suspicious behaviour. The two phases of a majority of anomaly detection systems consist of the training phase (where a profile of normal behaviours is built) and testing phase (where current traffic is compared with the profile created in the training phase). Anomaly detection has the key advantage over a misuse detection in that it doesn’t require the knowledge of specific attacks and therefore can also cater to unknown attacks as they will cause the system to behave in a different

manner. However the circumstances of this critical scenario makes it difficult to detect the difference between network attacks which are attempting to take down key nodes in the networks and benign network readings. When the bandwidth becomes a constrained resource in itself optimisation of it's usage is critical. Whilst this scenario is more common in the IoT, it exists in traditional networks as well. It is usually simple to find anomalies when there is very low network usage, however, at peak usage you will encounter many more false negatives/positive. In a emergency situation these could have severe adverse effects.

In this paper we investigate whether we can detect attacks on disadvantaged networks with an anomaly-based IDSs using the expected optimal behaviour of the system, rather than the normal bench-marked behaviour of the system, as it is normally done. To answer this question we use concepts from the well studied field of system optimisation and integrate these approaches into intrusion detection techniques. There are several overlaps in optimisation theory and IoT security, as defending an IoT system becomes a combined matter of ensuring good functioning as well as attack detection. To illustrate our approach we implement a system model that mimics the behaviour of a real-world IoT system and we generate optimal behaviour traces in a model of the system, following the optimisation criteria. By collecting data of the real system we can observe and minimise the security risk as an outcome of the systems constraints. This optimised behaviour is then used to train a detection system that can spot attacks that cause long-term harm but aren't detected by current techniques. This technique has the key advantage that it is able to make use of information not normally known at the network layer. The attack is at the network layer but impacts the device at the hardware layer, however the IDS only has knowledge of the network messages. So the extra information gathered and used in the model can make up for this detriment.

This approach can be used alongside existing IDSs to improve their classification power and tailor to the specific threats affecting IoT systems. Our contributions are threefold; 1) We present a new method for attack detection that is suited to detection of harmful behaviours in IoT systems we refer to as HDS; 2) We provide a novel methodology to model system behaviour that can be used alongside real system data to improve prediction; And 3) we provide a case study and quantitative evaluation of our approach on real-world IoT network

demonstrating how our approach can be used to effectively detect several common attacks on IoT systems. To illustrate our approach we make use of case study looking at temporary COVID19 field hospitals.

Paper breakdown: the paper is split into the following sections; In section E.3 we discuss the related work; In section E.4 the problem overview is discussed; In section E.4.1 we introduce the optimisation criteria forming the basis of the harm detection; In section E.6 the model is presented alongside the optimisation formula for trace generation; In section E.7 we explain how the system data is interpreted by the IDS to compare to the model behaviour; In section E.8 we highlight our evaluation criteria assessment methodology and experiment setup.

E.3 Related Work

Intrusion Detection is a well studied area, with early work starting in the early 90s [80, 183, 159], the field is perennially evolving as the scope of the internet changes, and attacks get more varied, the detection mechanisms evolve alongside them. The state of the art when it comes to servers and a wide range of ICT infrastructures is well established, with open source and easy to use tools available for a wide range of scenarios [149, 142, 35]. However, despite the best preventative measures, we see news of successful attacks targeting organisations and individuals alike on a continuous basis. Along with the increment in attacks comes a shift in infrastructure, what we commonly call the IoT, is a term used to describe a range of distributed, heterogeneous internet systems communicating in new fashions often without user intervention. With the raise in popularity of the IoT a growing interest from the research community investigates how to successfully secure it, borrowing from existing techniques and coming up with new solutions to the problems arising from this new paradigm.

Lots of of literature investigates the problem of implementing IDSs in the IoT [171, 56, 9, 5], several approaches and techniques are discussed. However, the IoT is a diverse term describing a wide range of systems and the extant literature highlights that no single solutions is able to cover the full scope. This is partially due to the vast new avenues of

attack (often unique to IoT) that signature based schemes such as SNORT [149] struggle to detect. The general consensus in the literature is that there is no silver bullet when it comes to finding attacks and several works attempt hybrid approaches combining signature and anomaly based techniques to optimise detection rates [148, 96, 34]The need for these complicated techniques arises as an outcome of these very open systems, they can detect well known attacks such as sinkholes using signatures, whilst simultaneously being able to deal with how distributed the architecture is and covering the new angles of attack such as node compromise attacks through anomaly detection. A full description of relevant attacks is presented in Sec. E.4.

Based on our previous literature review on the topic of intrusion detection in IoT systems presented in Chapter 3, over 60% of proposed IDSs focus on attacks that are due to sub optimal behaviour. By these characterisation we refer to i. routing attacks, which are attacks targeting the way communication is exchanged in the system (representing 30% of the literature); ii. protocol attacks, which are attacks on protocol exchanges such as man in the middle (representing 15% of the literature) and; iii. device exhaustion or dos attacks, which target the availability of an IoT network (representing 20% of the surveyed literature). What is common across all these attacks is that the objective of the attack is to decrease efficiency of the network and/or completely deny it, by making devices within it behave in unexpected fashion. As there is currently no literature proposed to target this full subset of attacks we investigate a potential detection mechanism that would find this attacks through harm analysis and optimisation. Optimisation aims to find the scenarios in which a system behaves in the manner in which specific values are maximised, naturally lending itself to counteract these types of attacks.

E.4 Problem Formulation

As highlighted by our summary of the literature there are several works making use of anomaly detection in IoT systems, anomaly based intrusion detection has several key advantages over signature based, they are able to detect unknown attacks, can deal with encrypted

data and do not need to continuously update their signature database. However, the literature highlights there are significant gaps in the theory behind anomaly detection in the IoT and that by itself it may struggle to detect these attacks, or be unable to understand why the classification was made.

To this end we propose a new approach to detect *harm* occurring to IoT systems. This will draw from several advantages of anomaly based IDSs but will differ in what it is actually detecting. We would like to argue that several of the attacks currently affecting the IoT are actually ones that harm the normal functioning of the devices themselves(as a consequence of their constraints) and those around the how the IoT systems as a whole operate (e.g. open environments and with heterogeneous protocols). Since the devices themselves will operate with lossy communication, limited computation, and no security measures on the devices it becomes the job of an external tool to do ensure that the operation of the system is still working successfully and securely.

We hypothesised that by applying techniques from Optimisation theory we can successfully classify actions as harmful and non-harmful, rather than the more traditional attack/non-attack. More specifically what we propose is to identify a key set of characteristics that are common across constrained IoT Systems and that may represent new avenues of attacks, attempt to classify behaviour that might lead to the negative increase in this qualities, minimise it (minimisation properties), then aggregating them with a set of known desirable properties of good system functioning, identifying which behaviours will improve these properties (maximisation properties). Through these characteristics we build an AB-IDS that can classify harmful and non harmful behaviour for any IoT system, real-time.

This requires for the system administrator to have knowledge of the system it is protecting, such as battery consumption of the devices actions, and consequently will require a complicated setup phase. However, once these measurements are in place the relative strain on the system due to the IDS will be minimal. This approach is very flexible and can be used as a stand alone IDS or alongside existing techniques to improve the accuracy of detection. We evaluate this approach on a IoT scenario and investigate its effectiveness in finding these attacks that harm the core aspects of IoT systems.

E.4.1 Attack Specification

The IoT and its many use cases suffer from *new kinds of attacks* which are often not pertinent to standard systems. Consequently arising the need for the IDS detection to be expanded. We propose a list of new vulnerability vectors in the IoT that would impact the ability for an IDS to detect intrusions as the following:

i. *Node compromise attacks*, devices in IoT systems are often vulnerable to physical takeover, allowing for malicious behaviour from previously benign members of the network;

ii. *Communication errors*, due to deployment in disadvantaged environments, IoT is very prone to network errors which can be a source of irregular traffic;

iii. *Battery drainage attacks*, the constrained resources may lead to selfish device behaviours, due to device self preservation constant polling may lead to battery drainage so devices will stay offline for large periods of time, leading to difficulty in behavioural patterning and harder network diagnostics;

iv. *Routing attacks*, the IoT is often deployed as an open network through WiFi or similar technologies, this opens up the network to external connections as well as new attacks to do with routing, such as sinkhole attacks;

v. *Compromised communication*, as a consequence of low bandwidth it is difficult for devices to reach far away destinations,so the network configuration becomes very important, therefore, if a node is identified as a key connector between two of the devices and is consequently compromised, communication in the network is broken;

A malicious intruder may, for whatever reason, wish to disrupt the availability of parts or the full sensor array. In order to do so he wishes to drain the battery of some sensors. Our scenario makes use of cryptographic authentication to allow for the data to only be collected by trusted users. One of the more energy consuming aspects of any message exchange is cryptography [182], by attempting authentication to various sensors in the system he

can systematically drain the battery of the whole system. Whilst this behaviour might be suspicious under very low system load, it would be much harder to detect in an emergency scenario such as this with several first responders on the field. Since the attack behaviour cannot be easily spotted, we instead can keep track of the well being of the system, whilst the attacks are hard to spot the harm is measurable and can be used to trace back the attacks. We propose a technique to generate traces of *optimum* behaviour that will minimise harmful actions within a system, in combination with maximisation of positive aspects that increase the system well being.

Whilst generating optimum traces is very difficult in the real world, it becomes much more feasible when used in models. The model abstracts to only include the characteristics that are of interest and the problem space is therefore greatly reduced. Through a model we can simulate large quantities of behaviour to observe the outcomes of different routing strategies and different behaviours. Using Markov Decision Processes (MDPs) we can find the behaviours that will maximise the system efficiency according to our optimisation strategy. MDPs have some key advantages: they have substantial tool support such as PRISM Model Checker [102], they rely on probabilities and non-determinism to recreate systems and they provide the ability to find the optimum paths through the system using the reward function. Through the reward function we generate traces that mimic potential system behaviour. Taking advantage of the relatively simplistic set of behaviours that an IoT device can perform, we can implement a variety of systems focusing solely on their network traffic. When the IDS system is deployed in the real world it compares the network traffic with the possible optimum behaviour traces generated from the model and then labels any deviating sequence as harmful. This approach allows to detect a variety of potentially dangerous attacks and can be extended for different scenarios. The model relies on several information being collected for a specific device and must therefore require for several measurements to be taken regarding the devices and their functioning. We first describe the set of optimisation criteria in sec. E.5 and then discuss the model implementation in sec. E.6.

E.5 Optimisation Theory for Attack Detection

The harm detection strategy takes the form of a function F, which takes as input the set of IoT devices $N = \{n_1, ..., n_z\}$, and for a given run of the system calculates the output of each optimisation variable $\omega_i \in \Omega$, minus the set of minimisation criteria $\lambda \in \Lambda$ for each device $n \in N$ and sums it up, with the optimal strategy being the one with the highest value. As the system is probabilistic and continuous the optimal path may vary for a given scenario and there might be multiple optimum traces. We also introduce a level of uncertainty, as IoT systems are often distributed it can sometimes be the case that an IDS, may have to function under incomplete knowledge and therefore account for the known information denoted by $KI(n)$. Whilst incomplete knowledge may reduces the accuracy of the detection, one advantage of the approach is that it is able to look at more than just a single packet and its effect on a device, as it will still effect the remainder of the system.

$$F(N) = \sum_{x=1}^{\Omega} \left(\sum_{n=1}^{N} \cdot \omega_x - \sum_{n=1}^{N} \cdot \lambda_x \right) \in KI(n) \tag{E.1}$$

Different weightings $\psi_i \in \Psi$ are assigned to each ω_i and λ_i depending on system choices and prioritisation, using this value we can get different paths that are critical to a specific system or to the choices of the system administrator.

Battery Power

We focus on a restricted set of IoT Devices, namely battery powered sensors. A device in this scenario will have a restricted battery life and certain operation will cause high strain on its duration. As such standard system behaviour should aim to minimise these operations. A device will spend large periods of time in sleep mode and targeted attacks may aim to disallow this behaviour and consequently drain large amounts of battery. One of the key difficulties in this field is that the devices themselves will have no way to self regulate their own battery usage, this is is due to the device being very simplistic scope and costs. Furthermore, whilst a host based IDS would sometimes observe battery drain, it is not possible to do for a NIDS

without additional knowledge. By taking readings of battery drain following the methodology proposed by Farnell [65] we calculate a drain per message as:

$$\omega_{bp} = \left(Capacity(mAh) Voltage \cdot Consumption(mAh) \cdot 0.7 \right) \cdot \psi_{bp} \tag{E.2}$$

Bandwidth

The bandwidth is the amount of data that can be transmitted in a fixed time, or rather it is characterised as the maximum rate of possible data transferred across a network. This metric is specifically important in IoT systems as it sets them apart from Wireless Sensor Networks (WSN), whilst a WSN is often a closed system the IoT is fully open and wireless and therefore needs to cater to a much higher rate of traffic. It is important that the bandwidth is not flooded by “bad" data and is instead used to improve system operation. To measure bandwidth in a system we use the standard equation:

$$\omega_{bw} = \left(\frac{Return\ Window}{Time\ to\ Return\ Round - trip} \right) \cdot \psi_{bw} \tag{E.3}$$

Coverage

A lot of IoT systems are built on lossy and often unreliable communication channels, often using UDP over more traditional TCP. This causes the coverage of the system to be a common issue, with packets dropping or not reaching destination. This feature makes it hard to make short term decisions as to whether the system is being attacked or whether there is a problem with the connections. By accounting for lossy channels in our model we can observe what a probabilistic outcome of lossy communication looks like and then differentiate it from attacker behaviour. The formula for coverage in networks is:

$$\omega_{cov} = \left(\frac{Completed\ Message\ Exchanges}{Conversations\ Initialised} \right) \cdot \psi_{cov} \tag{E.4}$$

Throughput

IoT Devices can be easily tampered with, or taken offline. Throughput is defined as the number of messages processed over a given time interval (cumulative messages sent/current time). By definition if a message takes longer to send the throughput will decrease, hence an attacker targeting computationally intensive tasks that delay the transmittance of messages is going to cause a decrease in the systems throughput. This is one of the key metrics of a systems efficiency and is calculated by:

$$\omega_t = \left(\frac{Messages\ in\ System}{Time} \right) \cdot \psi_t \tag{E.5}$$

E.6 Model Based System Optimization

A model has a key advantage over a real system in that it can be used to extrapolate key data of interest and quickly observe how the components react. The intent of the modelled system is in essence to produce traces of behaviour that correspond to the behaviour of real devices at the network level. This means that by comparing the model output to the captured data from the system, a direct comparison can be made. The traces can be used by the HDS as markers of optimal system behaviour, by matching what is going on in the system to the traces any deviating behaviour can be found. As discussed in previous work [17], there are several process algebras that aim to capture actions taking place in the system. However, the purpose of these semantics is very different, whilst common methods are used to identify potential bottlenecks or deadlocks to reason about a system, we wish to recreate network behaviour to find desirable traces. Taking this into accounts we extend the semantics to capture routing behaviour, message exchanges, timings and the impact of these exchanges on the devices themselves. We expand on our previous model to focus on how messages travel through the system and how to capture the optimum variables, and deviate from the concept of attack behaviour.

Given a global set of protocol messages γ, a device D is a tuple (A,T,L,P), where $A \subseteq \gamma \times [0,1]$ is the set of potential protocol actions valid for the IoT System; T is the set of

timings to process different actions where each action γ when performed will add a a value t to global value *clock* until a full exchange is completed and the value is reset; L is the target destination for the message in the form $L \subseteq N \times [0,1]$ where N is the set of devices in the network and where $(s,n,p) \in L$ means the device $D_1 = s$ source chooses destination n with probability p, mimicking ad-hoc behaviour. Finally, P is used to model the likelihood of the message being received, where $(a,p) \in P$ means the process successfully performs action a with probability p, and such that $\sum\{p \mid (a,p) \in A\} = 1$ We make the choice to include a local time in the form of a clock as the core important factor is the time it takes for a message to propagate through the system. A monitor keeps track of when the message has reached the destination (or if the message doesn't) and resets the clock. Another feature to note is that there is a probability of the message not being sent to mimic a lossy channel which is common in IoT infrastructures.

The second component in the system is a *monitor*(M), the purpose of monitor in our model is to find optimum pathing and keep track of device behaviours. It leverages on it's knowledge of the devices battery, memory and current actions to find the optimum pathing for any given behaviour. It's job is also to update the clock and reset the actions when a message has reached destination (as the original sender is not aware of this). A monitor M controls values λ and ξ, where λ is the remaining battery of the device and ξ is the current memory value. Given a global set of battery drains Ω the λ is measured as a quantity that is linearly drained by a ω_a where $\omega_a \in \Omega$ is a constant battery drain of an action, the monitor will update its λ value to λ' after each corresponding device action. Conversely a $\xi \in \Xi$ starts at zero and is filled whenever a device receives a message (to simulate a open thread or data on RAM), the value ξ is linearly decreased by a value $\theta \in \Theta$ in the form $(a,D) \in \theta$ where each theta is associated to a single device performing an action after it forwarded the received action a.

Rule 4. Given a device $D_1 = (A_1, T_1, L_1, P_1)$, a communication initiated by device D_1 in synchronization with monitor $M_1 = (\lambda_{D_1}, \xi_{D_1})$ on an action a with an associated timing t,

and destination n with a success probability p, takes the form:

$$\frac{(a,p) \in A_1 \; n \neq D_1 \;\; t_a \in T \;\; p > 0 \;\; n \in L \;\; T_1 + t_a \;\; n = 1 \;\; \lambda_{D_1} - \omega_a \;\; \xi_{D_1} - \theta_{D_1}}{(A_1,T_1,L_1,P_1)||(\lambda_{D_1},\xi_{D_1}) \xrightarrow{(s,a,t'_a,n')} (A_1,T'_1,L'_1,P_1)||(\lambda'_{D_1},\xi'_{D_1})}$$

Rule 5. Given a device $D_1 = (A_1,T_1,L_1,P_1)$, a communication received by device D_1 from $D_2 = (A_2,T_2,L_2,P_2)$ on an action a originally, with an associated timing t, and destination D_1 resets the clock for the communication and unlocks the destination, taking the form:

$$\frac{(p_a,a) \in P_1 > 0 \;\; n = D_1 \;\; n \in L \;\; t_a \in T_2 = 0 \;\; s_{D_2} \in L_2 = 0}{(A_1,T_1,L_1,P_1)||(A_2,T_2,L_2,P_2) \xrightarrow{(s,a,t'_a,n')} (A_1,T_1,L_1,P_1)||((A_2,T'_2,L'_2,P_2))}$$

A full protocol exchange can involve several transmission of messages by different devices until it has reached the intended destination. Once the destination is reached the monitor will update the clock and pathing, there is also a probabilistic possibility that a message is dropped and doesn't get to the destination. The routing through the system is achieved using MDP *optimum* path. Theoretically the quickest path would be to send directly to a device, or route it through to another subnet the shortest number of bounces. However, the monitor might instead choose to route it through other devices to optimize the systems well being instead (as the quickest path might cause devices to be drained). The non-determinism in combination with the reward structures to monitor messages, time and drain are used to find the optimum behaviour strategy, or the most rewarding traces through the system. Implementing the model in a tool like PRISM allows us to make use of Probabilistic Computation Tree Logic (PCTL) [26] to calculate various conditions of pertinence to the system, to compute the optimal behaviour, and to simulate traces of the model.

E.6.1 Trace Generation

In order to recreate the optimal traces that match the optimisation scenarios described through the PCTL we use formal verification. In formal verification, finite state model checking needs to find an automaton equivalent to a given PCTL formula, i.e., such that the PCTL

formula and the automatom recognise the same ω-language. In the case of PRISM Model Checker, the PCTL is specifically matched by a Büchi Automaton [36]. In this context this automaton represents the full set of actions resulting from the discovery of the optimal system property of the MDP. Each valid path in the Büchi automaton can be traced back to the labelled transitions of the system model enabling for easy understanding about behaviours and consequences of action sequences. Whilst the states of the Büchi automaton do not correspond to the states of the MDP, a easy matching allows to see the current state in a system walk and to see consequent outcomes of particular behavioural choices in a system. This relation provides insight into how systems behave giving the user understanding about consequences of actions and outcomes of design decisions; allowing to use this for system customisation and evaluation.

E.7 Harm Detection System

The HDS needs to be able to compare the behaviour in the system to the behaviour exhibited by the model. To do this we put in place a tool that can interpret the network traces to match the model output. This is based on having existing knowledge of the network as the various devices need to be bound to a fixed identifier. We use WireShark as the state of the art tool to collect network data. The HDS translator then converts the data to match the traces of the system, it performs a pattern matching on the data and matches it to the values in the model. The matching identifies a starting state, and follows the path of the trace step by step each until the packet destination is reached. Of course there will be various valid paths achieved through the simulations. Deviating behaviour is considered an attack on the system as it will be a behaviour that will have cause harm. To improve the performance of the comparing algorithm, the traces of the system are constructed as a state diagram, so the algorithm can follow each option quickly. This method allows for efficient on the fly intrusion detection. As this approach can be restrictive we also propose a potential extension, we introduce uncertainty value in the pattern matching as this allows from some variation to

account for non-optimal behaviour happening naturally. This can be observed graphically in Fig. 5.6.

E.7.1 From System Data to Model Behaviour

Whilst model traces take the form of a series of actions, network traces are much more complex. This requires transformation into more focused behaviour before they can be used for anomaly detection. Based on the modelling described in Sec. E.6, we propose a network traffic characterisation method. Each state in the model has specific characteristics related to where the packet is coming from, type of message, time message was sent and time it takes to reach destination. These are all characteristics that can be extracted from network traffic of an IoT system. Furthermore we wish to deduce the state transition system of the corresponding LISA model, based on the flow of the model. This can then be compared to see if it corresponds to a traffic flow of an HDS model.

The first step becomes identifying the state of the network Δ. This is done on a per package basis for each package within a window N, leading to state space $\Delta = \{\delta_1, \delta_2, \ldots, \delta_N\}$. This window is flexible, based on the type of threats, speed of computation, and length of traces one has for their system. Using the Δ, message flows Ψ are identified within the current window. A message flow ψ, is composed of a series of s, a, t_a, n transitions, each component of which is gathered from the network package. Using UDP/COAP as an example, UDP contains Source and Destination, which correspond to s and n respectively, the UDP message type represents the a and t_a is gathered as part of the environment data. We note that since our scenario involves to find optimal network usage, we do not observe additional features such as package length, as it would exacerbate the state space of the system needlessly. A final transformation occurs for the sake of the time windowed gathering approach, the time for action a as t_a becomes a cumulative increment from the initial time gathered at state δ_1 which each new time being added on to the next state to mimick the behaviour of the model. Using the characteristics of all the flows within the network capture a transition system is constructed. On account of noise or dropped packets a certain threshold variation α is allowed from the similarities of traces, which can be customised for strictness. Finally

for efficiency only the first y packages are compared before a full comparison is made to quickly sort through the current scenarios.

E.8 Experiment Setup

We evaluate the approach to test it's effectiveness on a real system. We set up a virtual network of devices to mimic an IoT system set up in different topologies and setting up various routing paths. We take measurements from real IoT devices to extrapolate to the VM as we are not able to virtualise battery. Beyond this assumption we calculated time to send a message/log a message, baseline battery usage (from the IoT device), percentage increase in battery usage and battery drain per action. We then quantitatively evaluate the HDS performance on a series of metrics described in E.8.1.

The system was setup using VirtualBox and mimicking IoT devices functionality, it consisted of three sub-nets composed of different devices, mimicking a e-health station in Fig E.1. To mimic IoT devices we made use of tinycore linux a very minimal (12 MB) modular linux system that we used to implement basic IoT functionality of message exchanges. The traffic was routed in between sub-nets by routing nodes (also containing the IDSs) that could alter message flow and control traffic, each of them have HDS capabilities, and represent the fog layer of the diagram. Finally a central server mimicking a cloud controller in implemented, to ensure correct routing and processing of information. The Routers where implemented using Ubuntu Servers and we implemented quagga routing suite and set it up to direct the messages through the system. To mimic sensor readings, real e-healthcare datasets were used [1].

Case Study - System Setup: Due to the global COVID19 pandemic, on 3 April 2020, the government of the United Kingdom announced the construction of the NHS Nightingale hospital [2]. These temporary facilities were designed as state-of-the-art e-hospitals with internet connectivity and means to quickly expand if needed. The temporary field hospital could house 5000 beds,each integrated with sensors and Internet connectivity. This one of

[1] Available at : https://physionet.org/about/database/

[2] "Coronavirus: Nightingale Hospital opens at London's ExCel centre". BBC News. 3 April 2020

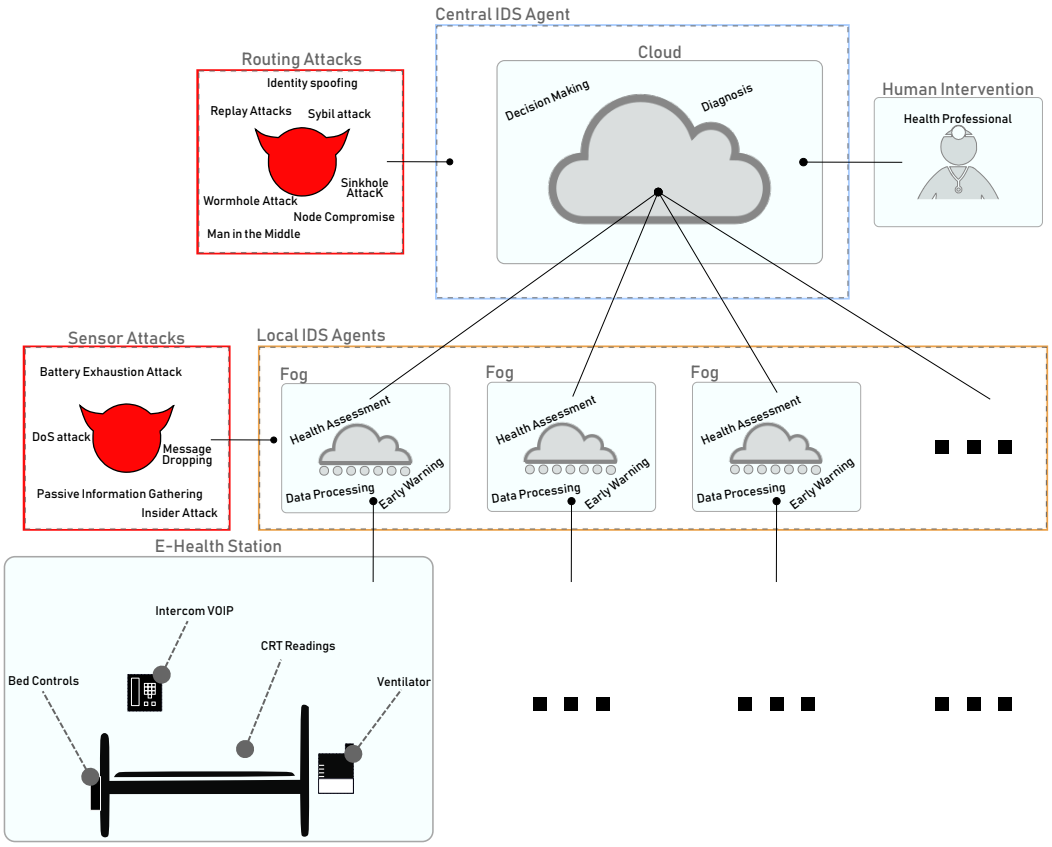


Fig. E.1 E-healthcare facility used for IDS Case Study, including attacks and IDS setup

a kind IoT scenario, where the effects of an attack could be human lives, creates a unique security problem.

A configuration of the network used for the experiment is presented in Tab. E.1, *Agent 1*, represents a local agent with direct descendants whilst *Agent 2* has a hierarchical structure from cloud to end devices with both local and central IDS agents. A diagramatic visualisation is provided in Fig. E.2.

E.8.1 Evaluation Criteria

In the specific scenario we are investigating there is going to be a very high level of bandwidth usage. An attack that has risen in popularity that is particularly effective in this scenarios, is an attack known as *Slow DoS* [139]. Unlike traditional DoS attacks which often spam to exhaust resources, slow dos use smarter tactics, often targeting specific functionality without

Table E.1 Network table case study scenario

Device	Interface	IP (enp0s3 NAT)	Manages	Interface	IP (enp0s3 NAT)
Agent 1	enp0s8	192.168.1.254	IoT 1.1	enp0s8	192.168.1.1
	enp0s9	192.168.100.1	IoT 1.2	enp0s8	192.168.1.2
	enp0s10	192.168.101.1	IoT 1.3	enp0s8	192.168.1.3
			IoT 1.4	enp0s8	192.168.1.4
Agent 2	enp0s8	192.168.2.254	Local IDS 2.1	enp0s8	192.168.2.1
	enp0s9	192.168.100.2	IoT 2.1.1	enp0s8	192.168.2.11
	enp0s10	192.168.102.1	IoT 2.1.2	enp0s8	192.168.2.12
			Local IDS 2.2	enp0s8	192.168.2.2
			IoT 2.2.1	enp0s8	192.168.2.21

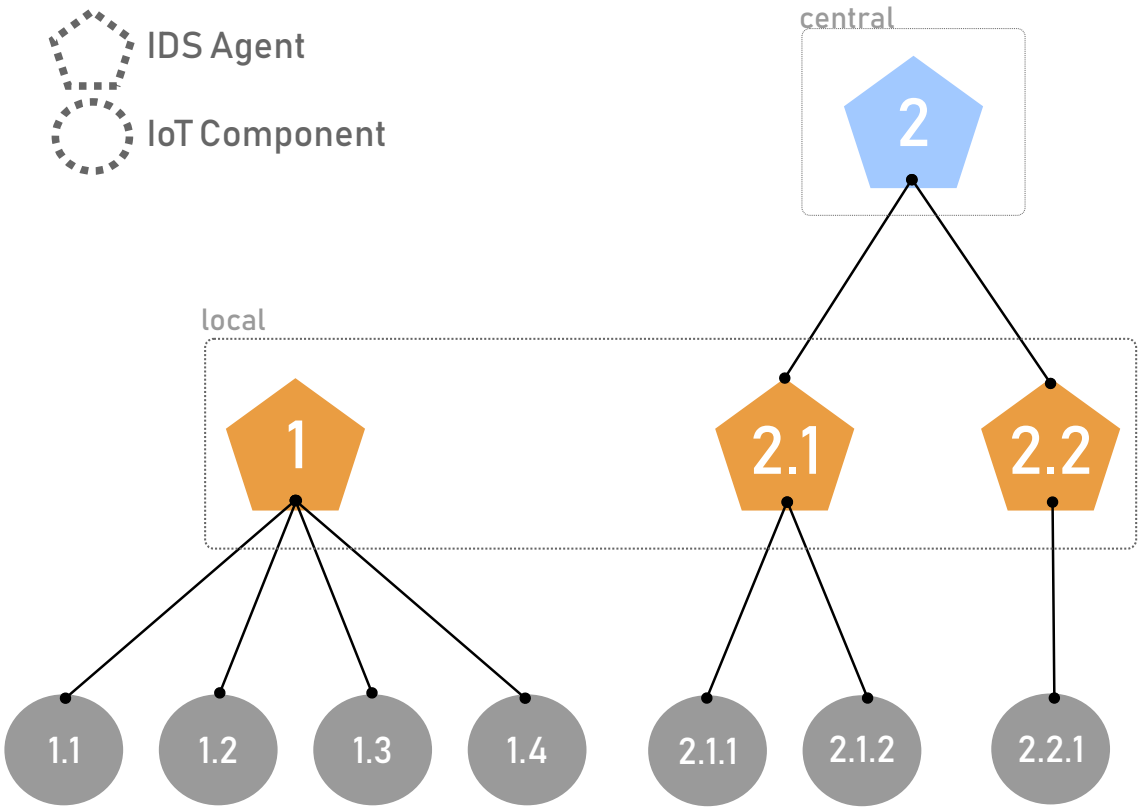


Fig. E.2 Diagramatic representation of case study scenario

necessarily flooding the system to achieve the same result. Consequently, it is much harder to detect, especially under high system load. No current slow DoS tools exist specifically for IoT, so to test the effectiveness of our HDS in finding slow DoS attacks we tested three of the most common slow DoS attacks and used them on our experiment IoT setup:

1. Pyloris [3], a configurable tool that aims to exhaust a services resources by opening multiple TCP connections.

2. R.U.D.Y. (R-U-DEAD-YET?) [129], an open source tool on google code, performs a slow DoS using POST messages with unusually long content length, allowing to consequentially fill the form slowly and in small chunks causing a long backlog of server threads.

3. Sockstress [4], works by using RAW sockets to establish many TCP connections to a listening service. Because the connections are established using RAW sockets, connections are established without having to save any per-connection state on the attacker's machine.

For the second phase of evaluation we focus on the routing based attacks. This is slightly harder to evaluate in a live environment as there is no tool available to easily simulate these scenarios. However, fortunately this can be artificially in our virtual network, by simulating the impact of these attacks and rerouting through the quagga router nodes. The HDS was then evaluated for accuracy in detecting the three attacks using F1 score. We assess the performance overhead of the approach as this is an important aspect to consider in IoT systems. And finally show correlations between scalability and the results of the HDS and discuss advantages and limitations.

[3]PyLoris, Motoma and PyLoris and Python, available at: https://motoma.io/pyloris
[4]Sockstress Tools & Source Code, Jack C. Louis available at: https://defuse.ca/sockstress.htm

Appendix F

From Secure Protocols to Secure Systems

F.1 Chapter Summary

As our daily lives become ever more connected, the need for security becomes more important. Due to the constraints of the IoT, the protocol designer must make careful considerations when making IoT security design decisions. To advance the IoT towards a more secure and interconnected future, new protocols are being designed. It is therefore desirable to verify their security, as doing so has been proven to be a non trivial task. We explore this process by creating the first full formal verification of ACE-OAuth a new IoT authentication protocol. Noting the difficulty in verifying such complex protocols we design a methodology able to observe different implementation options and verify the different scenarios. Expanding on this even further, a new protocol design tool MetaCP is devised and presented in F.3. With a modern graphical interface, the cryptographer can design a protocol with MetaCP and simultaneously export it into multiple target languages for formal verification. A protocol dictates how communication is done over the internet and therefore serves as a cornerstone for observation of device behaviour. Once we formally verify whether a protocol is vulnerable or secure from attackers, its formal model may be used as the basis for our modelling approach of attack behaviour in Chap. 5. Attacks are used for attacker behaviour and the formal protocol specification is used to model device interactions.

F.2 Chapter Introduction

As discussed in the previous chapter, the highest complexity within the IoT is their interactions. The way devices interact is dictated by protocols. Protocols are the underlying foundations of the internet, dictating its operation. As the internet ever progresses so must the protocols that define its functioning. With great advances in interoperability and sheer breath of what can be achieved in new internet scenarios such as IoT and Cloud, so do the rewards of breaking such protocols increase. A malicious attacker has a whole lot to gain, and what better avenue of attack than protocols to attack it's targets - break the foundations and the whole infrastructure crumbles down. To this end it is of paramount importance to make these foundations air tight, and this is where formal methods come in. If the attackers are getting smarter protocols too need to make advances. As formal protocol verification tools have become more mature they are rising in popularity especially amongst academia, they have proven effective in finding and solving vast arrays of security breaches [118, 78]. It has come to the point however where it is not simply good enough to find the problems after the protocols have already been designed, we must prevent the problems taking place in the first place.

Secure protocols lead to less attacks. This in itself makes it easier to defend the IoT as we can use these protocols to design more secure systems. However, not all deployments are easily updated to be secure and we cannot possibly verify every single protocol implementation. We therefore still need to make use of IDSs as an extra layer of security. Most attacks in the IoT, are due to protocols and a lot of IDSs are consequently focused around this. An attack may not always require for the protocol to be insecure, as legitimate behaviour may still cause adverse effects, or may cause adverse effects in a specific system only. We therefore use this system to aid the IDS in detecting attacks in the following way: 1) If an attack is found on the protocol currently implemented in the system, we can use the attack signature in the IDS to detect intrusions; 2) If the current version of a protocol is found to be secure, bu an old version of a protocol is known to be insecure, it is likely that an attacker may attempt to use the old attacks on the system, and so a intrusion can be found; and 3) Even if the protocols in the system are secure, they still may be implemented incorrectly, and

consequently be vulnerable, so we can use the correct expected and secure behaviours as the benchmark of what should be taking place.

The work presented in this chapter was inspired by a collaboration with my PhD industrial partner ARM ltd, Arnaboldi & Tschofenig [19]. The work presents a framework to generalise a formalisation of protocols making it suitable to formally verify several different implementation options, the full paper is availableat Arnaboldi & Tschofenig [19]. ACE-OAuth is a novel IoT protocol which is currently in the process of standardisation, my co-author H. Tschofenig is one of the designers of the protocol and asked for my expertise in protocol verification to conduct the analysis of the protocol. After reading over 150 pages of specification, I discovered that this kind of document would allow for several different implementation options. Therefore, verifying the protocol would require multiple models of the of several implementation options. This spawned the idea of making a flexible verification framework where different implementations could be tested, as well as the concept of minimal requirements. This idea allowed the discovery of the minimal implementation that would fit the specification whilst maintaining the security objectives. This work was all done manually, which was not an easy task as dealing with formal languages is tricky and prone to error. Discussing this with a colleague who also specialised in protocol verification we came up with the idea to aid the protocol designer through a graphical interface. The interface would mimic the process used to currently design protocols and then automated the translation into a formal language. In collaboration with colleague R. Metere we developed the tool MetaCP, Arnaboldi & Metere [15], this is discussed in the second section of the chapter. This flexible approach allowed to convert a protocol specification not only into protocol verification languages. but into our bespoke device behaviour language as per Chap. 4.

The focus on formal verification of protocols is directly useful in attack detection as it allows us to predict potential vulnerabilities and prevent them. If we model device interactions we need to consider protocols, correct device interaction are a stateful representation of communication dictating good behaviour. The added benefit of our design approach is that we can use the already created protocol formalisation to act as the basis for modelling device

interactions as per Chapter 4. However going from a protocol specification to a system model is a non trivial process. So we leverage our data centric approach that allows for the automated conversion from Protocol Specification and Verification (PSV) data structures to a target language of choice. A PSV file collects information about the protocol as a data structure in an XML language whose constraints are defined in a Document Type Definition (DTD) file. From this base language we initially convert to formal protocol verification languages for security analysis, however, we can then use this same approach, described in Arnaboldi & Metere [15], to translate into *gentleman/rude* device semantics automatically. We need to to have secure protocols to defend IoT systems, this method allows for the design of secure protocols; and in the case where they are insecure this same approach can be used to preempt and discover attacks on systems. The latter approach will be further discussed in Chapter 5.

F.3 MetaCP: Cryptographic Protocol Design Tool for Formal Verification

Formal security analyses of communication protocols base their model and reasoning on the interpretation of their specification. In this context, machine-aided verification tools have become popular in the past three decades reducing the complexity and improving the repeatability of the security analysis. However, the state of the art tools focus their automation on the last part of the verification process, i.e. *after* the model has been developed. MetaCP proves itself a promising first effort to automatise the process of formal verification, focusing on the modelling part [15]. We plan to let it be freely available at metacp.eu.

The model interpreting the design of a protocol is the first mathematical artefact in the process of formal verification, but the interpretation process itself is not mathematical. Therefore, a copious amount of papers are written to convince the community of security experts that the models indeed capture the relevant security aspects of the specification. Those discussions may sometimes overlap or repeat across different papers. MetaCP enhances the reusability of any exporting language to any protocol of interest.

The automated tools for formal verification of security protocols are based on ad-hoc languages that allow for reasoning in some cryptographic model to prove (or disprove) mathematical theorems capturing the security properties desired. Mechanised formal verification is a very complex task and, unfortunately, the languages used are not developed to be understandable or usable, without clear syntax and semantics. Hence, they are quite obscure and very difficult to use or learn by anyone who does not happen to be directly involved with the related projects. Even knowing the languages very well, writing the code is reduced more or less to writing pure text files without any aiding framework. Moreover, formal models written in two different formal languages may capture different aspects of the protocol, and what is verified in one tool may be seen as an attack from the other tool. Conversely, with the intuitive graphical interface of MetaCP, one can design a protocol and automatically export it to multiple other target formats: we currently provide exporting plug-ins to Tamarin [119], the applied pi-calculus of ProVerif [2], and LaTeX illustrations.

We show how the data-centric approach brought by MetaCP entails many benefits to the process of formal verification of security protocols. Nevertheless, analogous concepts apply to *any* domain, and can be seen as future extensions of MetaCP. In particular, MetaCP allows for (i) reducing the amount of ambiguity typical of specification written in natural languages[1]; (ii) making the design of a protocol much more intuitive through a modern graphical interface; (iii) including a corpus of reusable exporting plug-ins; (iv) kicking off the work of both security experts and learners by systematically and automatically exporting from the design or specification; and last but not least, (v) quick fixing of mistakes in the design, as they will automatically reflect to target languages when re-applying the relevant plug-ins. It is not difficult to see how part of those benefits also apply to other disciplines, e.g. in the didactic field. As a running example, we re-create the Diffie-Hellman key exchange protocol in MetaCP, exporting its design to ProVerif and a LaTeX illustration in just a couple of minutes of effort.

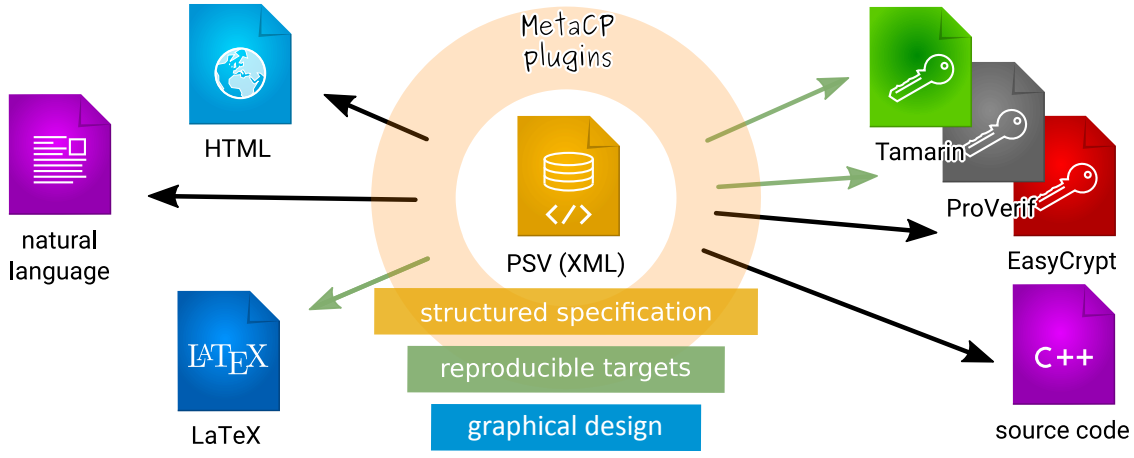
[1] This is a common concern when one models from specification [51, 119].

F.3.1 Architecture

The crucial innovative aspect of MetaCP is its data-centric approach, where the protocol specification is stored in a structured way, as shown in Fig. F.1. The benefits of this approach are manifold and enable for unprecedented little effort in going from the design to formal verification of security protocols.

The MetaCP architecture is composed of three kinds of components: **design**, **specification**, and **export**. At the design level, we provide an intuitive Graphical Design Editor (GDE) that allows for creating, dragging and dropping elements that will be later saved into the specification. The GDE is written in a modern web application framework, using ReactJs, Bootstrap, NodeJs, and Redux. At the specification level, we provide a data structure written in XML language meant to collect the information required to fully describe a security protocol. Such structure can later be interpreted by means of a plugin. We provide two plugins towards formal verification languages, one for Tamarin and one for ProVerif, that automatically interpret the protocol described in their syntax. Furthermore, a third plugin exports into LaTeX code for illustration purposes. We found it comfortable to write the above plugins in XSLT, but they can be written in any language of choice. All these components are completely independent and their details are explained in the following subsections.

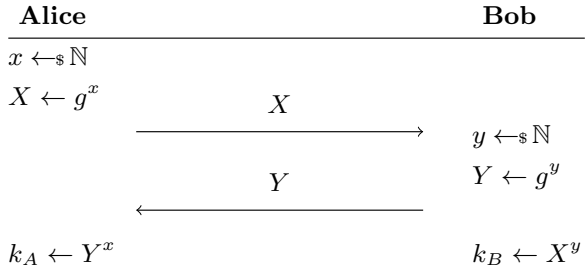
Fig. F.1 MetaCP supports a data-centric approach where the specification is stored as structured information. The green arrows point to the currently supported target tools. Original diagram from Arnaboldi & Metere [15]



Protocol Specification and Verification Data Structure

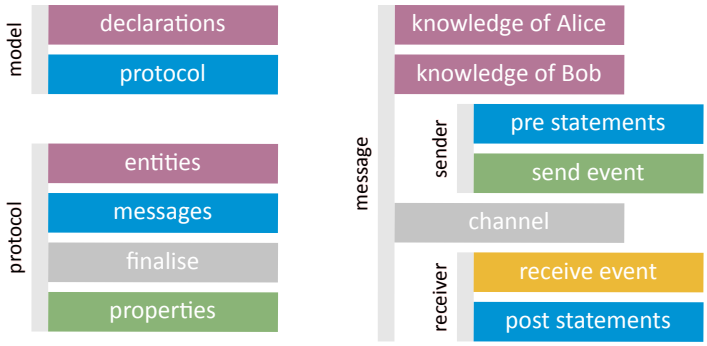
The ability for MetaCP to automatically translate into multiple verification languages resides in its description language, denoted as Protocol Specification and Verification data structure (PSV). A PSV file collects information about the protocol as a data structure in an XML language whose constraints are defined in a Document Type Definition (DTD) file. PSV is suitable to be easily extended and enjoy multiple interpretations. DTDs merely enforce the structure without adding strong constraints to the semantics, thus not breaking the flexibility required by our approach. Our approach is sensibly different from all previous approaches, where researchers struggled to find a single semantics capable of embracing the semantics of all desired target languages. The *single generic semantics* approach could work well for a few languages whose semantics were not too far apart, but would either fail or find it very difficult to capture the requirements of other languages. So, the very successful projects AVISPA [13], AVANTSSAR [12], and ProScript [95] are all not suitable to make a comprehensive multi-language translation tool due to the single generic semantic bottleneck. We illustrate how MetaCP is suitable for being a multi-language translation tool, through an example, the Diffie-Hellman key exchange (DHKE), as illustrated in Fig. F.2.

Fig. F.2 The Diffie-Hellman key exchange protocol as exported by MetaCP with the LaTeX exporting plugin, directly from the design. Original diagram from Arnaboldi & Metere [15]



The standard flow a PSV file is shown in Fig. F.3 and includes the following sections: declarations and the protocol.

Fig. F.3 High level description of the PSV data structure to specify protocols. Original diagram from Arnaboldi & Metere [15]



Declarations. To allow for a type system over all the structures used within a protocol specification, e.g. variables, constants and functions, the declarations of the corresponding membership sets are mandatory beforehand. Each subsequent declaration needs to refer to an existing set identifier.

Protocol. A protocol is composed of entities, messages, a final elaboration step after the messages, and finally the desired properties. The entities are the participants of the protocol that exchange messages whose directives affect their knowledge. The final elaboration step can include statements; for example, at the end of a key exchange protocol, the parties may reconstruct the key at that stage. Security properties that can be currently specified are correctness, authentication and secrecy. We do not discuss the security properties on this paper, and we reserve to comprehensively study them in future works.

The messages are structured in four parts: the *knowledge*, the *sender*, the *receiver* and a communication *channel* in the between. The **knowledge part** is per entity and lists all the known variables and constants by the entity *before* either sending or receiving the message. The knowledge is beneficial to detect or restrict the designer not to use unknown structures. The **sender part** shows two sub-parts: the first can include statements required to construct the message to send, and the second is the message as it is pushed to the channel. Similarly, the **receiver part** shows two sub-parts but, in this case, they are inverted: the first is the incoming message, while the second are statements manipulating variables in the knowledge of the receiver, which has been just augmented with the received message. We remark that

the received message may not be the same as originally sent by the sender. Any manipulation to the message can be done in the **channel part**. This structure has the benefit of allowing the designer to model different scenarios of interest. In particular, (i) systematically biased channels can be implemented with a function in the channel, (ii) man in the middles may be embedded inside the message, without creating additional messages and simplifying the design of attacks, and (iii) faults can be implemented either as empty received messages or probabilistic functions in the channel. The above listed scenarios are merely examples, and other scenarios can benefit from this particular structure of the message. Using the DHKE running example, we cherry-picked the first message sent in the protocol. We replaced the details of its content with a brief summary. The extended content will be later explained is shown in the right hand of Fig. F.4.

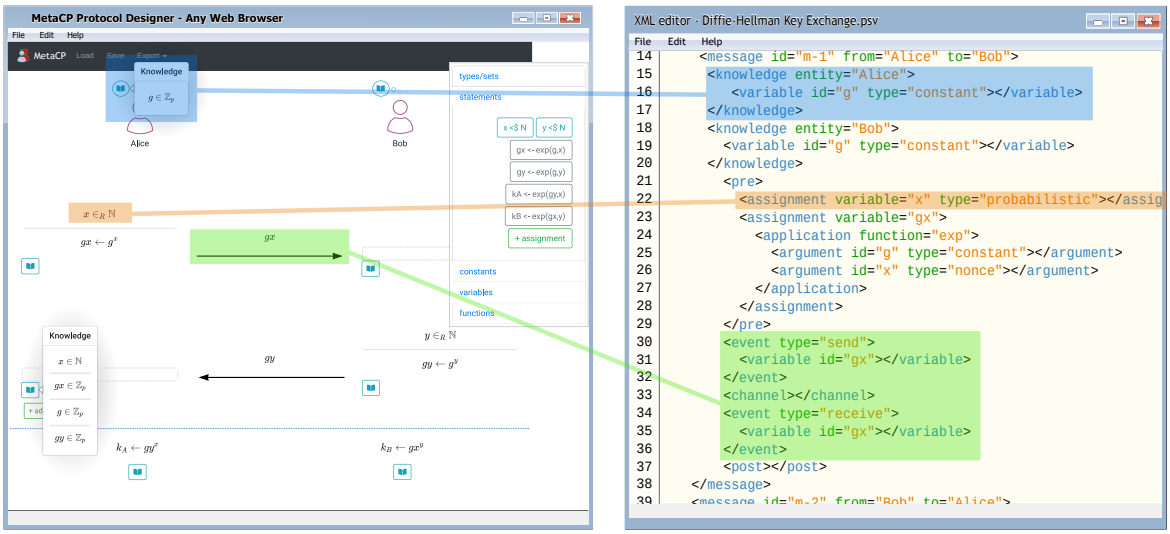
```
<message [...] from="Alice" to="Bob">
  <knowledge entity="Alice"> [...knows g] </knowledge>
  <knowledge entity="Bob"> [...] </knowledge>
  <pre>[...elaborates gx]</pre>
  <event type="send">[...sends gx]</event>
  <channel></channel>
  <event type="receive">[...receives gx]</event>
  <post></post>
</message>
```

Graphical Design Editor

Whilst we see the XML as intuitive and in line with how you would describe a protocol in a specification, in practice, MetaCP is equipped with a modern Graphical Design Editor (GDE) as illustrated in the left part of Fig. F.4, to aid the user with the design of the protocol rather than focusing on a formalisation i.e. the PSV. The GDE mimics the standard drawing process most familiar to any protocol designer, and it lets the user specify variables, functions and message flow. It does so through a smooth drag and drop design, making it easy to piece together the protocol. The GDE is intended to guide a user through the coherent definition of the PSV, automatically providing the following relationships in the data structure: first, the knowledge is automatically augmented as the protocol is constructed, and second, the GDE will enforce correct typing across the protocol and functions. The ability to store further

relations and information about the protocol is a significant aid that modern frameworks for programming languages usually incorporate as the basics. Once a desired protocol is drawn out it can be saved as PSV. The Fig. F.4 shows how some parts of the design reflect to the saved PSV.

Fig. F.4 The graphical design of MetaCP (left) is saved as the PSV format (right). Original diagram from Arnaboldi & Metere [15]



Exporting Plugins

A plugin provides a fully automated protocol-agnostic interpreter from PSV code to the desired semantics of the target language. We remark that our plugins are examples of interpretation of the target semantics: additional plugins targeting the same language are allowed.

The combination of the benefits of the GDE and the exporting plugins can sensibly improve the experience of protocol designers, even if they are expert in a specific language. To the best of our knowledge, the languages used in formal verification for protocols do not enjoy advanced frameworks. Therefore, their source code is more prone than other languages to subtle and hard-to-spot bugs[2]. Therefore, MetaCP enhances the quality of the work of

[2]Some tools work in untyped languages making even small typos hard to detect.

experts in those languages, both speeding up their work and mitigating (when not removing) many subtle bugs at the design level.

Plugins can be called in the GDE directly, as well as natively, e.g. as scripts in a shell, once the PSV is available. The architecture of MetaCP is such that all the components, PSV (with DTD), GDE and exporting plugins, are independent. So when a plugin is called in the GDE, it automatically and transparently generates a PSV as input to the plugin. The original paper contains a conversion from PSV to ProVerif as it was more relevant to formal verification of protocols. The current chaspter presents the conversion to our behaviour modelling technique as it is more relevant to the IDS work we present.

As the PSV is a structured container of the specification of the protocol, interpreting plugins confer semantics to that specification from the point of view of their target language. Hence, a plugin can be seen as the effort of applying the semantics of the target language to the structure of the source PSV. To do that, the plugin translates the PSV into the target language grammar. If they were two languages with their own semantics, some sort of bisimulation certifying that semantics are preserved in the translation would be expected. Differently in this case, we illustrate how the methodology of our plugin does not introduce errors in the target code upon certain conditions. This is used to then automatically construct a LISA model as per the previous section. A translation rule from PSV to MDPs in PRISM, following the formalism of the Prism system description language presented in the original work by Kwiatkowska et al. [98] with further details on concurrent composition available online [3], is presented. Similarly for the xPath directives, we refer to Clark et al. [47].

The key to read those rule is as follows: at the conclusion (bottom or bottom-right) we have the grammar of PRISM Model Cheker, which is inferred by the parts above its line or at the left of the corresponding inline symbol $\vdash$, while the lines above are the interpretation reading from the PSV whose notation uses xPath directives. Additionally, we use the notation explained as follows. We refer to the (ordered) sets of elements generated by an application of an xPath directive d within angle brackets, i.e. $\langle d \rangle$. If the result set is a singleton, we also refer to the single element with the same notation. Some rules are parametric, the parameter

[3] http://www.prismmodelchecker.org/doc/semantics.pdf

passed to them is superscripted after their name, so the notation $[\mathrm{r}]^p$ is for the rule named "r" with parameter p. To read attributes from tags, we use square brackets notation, so we denote the attribute `type` from the tag in e as $e[@type]$. Unlikely other common rules, they have to be explicitly called. The notation we use to apply a rule to all elements of a set of elements is a vertical bar with the application domain as subscript, e.g. to apply the rule [r] to all elements in $\langle d \rangle$, we write $[\mathrm{r}]\,|_{\forall e \in \langle d \rangle}$. As a short notation, if the rule to apply has the same name as the xPath directive of the set, we omit it leaving only the $\forall$ symbol, e.g. $[\mathrm{r}]\,|_{\forall}$ is short for $[\mathrm{r}]\,|_{\forall e \in \langle r \rangle}$. Finally, we shorten the call of two rules applying to set of diverse elements, e.g. el1 el2 with the vertical bar |, e.g. [el1|el2] will apply to elements whose tag is either el1 or el2 in the order they appear in the application domain. As the reader may already have noticed, the rules in Fig. F.5 and Fig. F.6 rely on some assumed relationships between the elements in the PSV. These relationships cannot be enforced by the DTD: for example, the rule [variable] `"typed"` assumes that the type will be actually found in the declarations. The DTD can only guarantee that the variable specified appears as an *identifier* in the past, but it cannot guarantee that the identifier was actually defined for the desired element. By designing the protocol with the GDE, MetaCP is able to generate a PSV where these relationship are always valid. To enforce such relationships in the PSV itself, we reserve ourselves to upgrade the DTD to the more powerful XML Schema Definition [172].

As introduced before, the GDE currently confers a type system to the functions, variables and statements to respect across the whole protocol, and manages the knowledge automatically at each step of the protocol. In particular, the extra properties provided by using the GDE are that all messages are exchanged between intended parties, and all statements can refer only to corresponding pre-declared sets, variables and functions, according to the knowledge of the party at that specific point of the protocol. An illustrated example for the rule [application] can be seen in Fig. F.5.

As opposed to much more rich syntax available in formal verification tools, in PRISM there is no such things as mathematical function applications such as encryption. Instead

Fig. F.5 Rules applied by the LISA plugin in MetaCP for **variables**, **arguments** and function **applications**. From the top to the bottom of the rule, they show how the PSV elements are interpreted to the grammar of the PRISM Model Checker.

$$[\text{variable}]^{\text{m}},[\text{argument}]^{\text{m}}\text{:}\ \frac{v \in \langle variable|argument\rangle \quad m = \bot|m = \texttt{"typed"} \quad l \in \langle comment\rangle := l \in \mathbb{Z}}{x \leftarrow v[@id] : [0..l]x \vdash Var_m \in M_m(Var_m, C_m)}$$

$$[\text{variable}]^{\text{m}},[\text{argument}]^{\text{m}}\text{:}\ \frac{v \in \langle variable|argument\rangle \quad m = \bot|m = \texttt{"typed"} \quad l \in \langle comment\rangle = \bot}{x \leftarrow v[@id] : [0..1]x \vdash Var_m \in M_m(Var_m, C_m)}$$

$$[\text{application}]\text{:}\ \frac{a \in \langle application\rangle}{i \leftarrow a[@id] : [0..1]i \vdash Var_m \in M_m(Var_m, C_m)}$$

a Boolean variable is used to denote whether it is e.g. encrypted (1) or not (0), for any such function. The usage of these rules is recursive and we can observe the usage of the [application] rule several times during rules for assignment of variables in Fig. F.6.

Fig. F.6 Rules applied by the LISA plugin in MetaCP for messages in the protocol along with depending rules.

$$[\text{entity}]\text{:}\ \frac{e \in \langle entity\rangle}{M \leftarrow [\text{message}]^{\,e}|_{\forall}.end \vdash P}$$

$$[\text{message}]^{\text{e}}\text{:}\ \frac{m \in \langle message\rangle \quad m[@from] = e}{M \leftarrow [\text{pre}]\,|_{\forall}.\,[\text{event}]\,|_{\forall e \in \langle event[1]\rangle} \vdash M}$$

$$[\text{message}]^{\text{e}}\text{:}\ \frac{m \in \langle message\rangle \quad m[@to] = e}{M \leftarrow [\text{event}]\,|_{\forall m \in \langle event[2]\rangle}.\,[\text{post}]\,|_{\forall} \vdash M}$$

$$[\text{pre}],[\text{post}]\text{:}\ \frac{M \in \langle pre|post\rangle}{M \leftarrow [\text{assignment}]\,|_{\forall} \vdash M}$$

$$[\text{assignment}]\text{:}\ \frac{\begin{array}{c}a \in \langle assignment\rangle \quad a[@type] = \texttt{"probabilistic"} \\ \hline x \leftarrow a[@variable] \quad M \leftarrow [\text{y}]\,|_{application} \quad M \leftarrow [\text{application}]\,|_{\forall} \\ \hline \{x,y\} \vdash Var_e \in M_e\end{array}}{}$$

$$[\text{assignment}]\text{:}\ \frac{\begin{array}{c}a \in \langle assignment\rangle \quad a[@type] = \texttt{"deterministic"} \\ \hline x \leftarrow a[@variable] \quad M \leftarrow [\text{application}]\,|_{\forall} \\ \hline \{x\} \vdash Var_e \in M_e\end{array}}{}$$

$$[\text{event}]\text{:}\ \frac{\begin{array}{c}e \in \langle event\rangle \quad e[@type] = \texttt{"send"} \quad M = M_{sender}[\![e]\!]|M_{receiver} \\ \hline c \in M_{sender} \quad v \leftarrow [variable]\forall v \in e \quad \Delta = \bot \\ \hline c \vdash [e]true \rightarrow \Delta : v'_1 + \ldots + \Delta : v'_n \\ \hline c \vdash C_{sender} \in M_{receiver}\end{array}}{}$$

$$[\text{event}]\text{:}\ \frac{\begin{array}{c}e \in \langle event\rangle \quad e[@type] = \texttt{"receive"} \quad M = M_{receiver}[\![e]\!]|M_{sender} \\ \hline c \in M_{receiver} \quad v \leftarrow [variable]\forall v \in e \quad \Delta = \bot \\ \hline c \vdash [e]true \rightarrow \Delta : v'_1 + \ldots + \Delta : v'_n \\ \hline c \vdash C_{sender} \in M_{sender}\end{array}}{}$$

The sender and receiver actions in our MDP take the form of two concurrent Module actions $M_{sender}[\![e]\!]|M_{receiver}$ following the PRISM concurrent process syntax, leading to two simultaneous transitions in the two modules updating the respective values in the event. As

can be observed the PSV doesn't support probabilistic behaviour and therefore there are no update probabilities Δ, however this suits the structure of an MDP in prism, not requiring probabilities to be specified.

F.3.2 Related Work

As it stands, it is very difficult for the casual user or even for the security professional, to ascertain the truthfulness of a formal protocol analysis, and how it relates to the original protocol. Witnessing the sensibility of researchers about this problem, namely the project "Automated Validation of Internet Security Protocols and Applications" (AVISPA) [13] has attempted to unify the verification, by presenting a single language of specification and automating the translation into various back-end tooling. Even with the integration of multiple verification options, research shows that protocols found to be secure by AVISPA were later found flawed in different formal verification languages [78], and no new extensions or backends were integrated.

Another attempt that the literature proposes to standardise the way in which protocols are designed is ProScript [95], here, a new high level language for the specification of security protocols is proposed. ProScript is able to automatically interpret from the high level specification to applied PI calculus, verifiable in ProVerif and CryptoVerif.

Another very successful case study in the area of general purpose languages is F* [168]. F* is a general purpose programming language based on monadic semantics, which supports semi automated proving and automatic translation into F# and OCaml. Like MetaCP, F* is able to represent aspects from programs and implementations in a way that can be easily extrapolated to other languages.

F.3.3 Future Work & Limitations

MetaCP is a first-of-its-kind tool aiding the protocol expert through the process of design and formal specification. It is based on a data-centric approach where the protocol specification is described as a data structure containing all the information that must currently be interpreted

from many pages written in natural languages. We showed its very promising potential with the example of formal verification languages for Tamarin and ProVerif, as well as illutrative LaTeX, and there is still a lot of space left for future expansions. Both ProVerif and Tamarin reason in the symbolic model, we have yet to research into the translation into tools reasoning in the computational model.

After its promising entrance, MetaCP is far from a mature solution. Our ideas of the key areas where MetaCP can grow and be extended lie in its limitation and are as follows.

- Focusing on the high flexibility of the PSV, we constrain its structure through a DTD; however, further investigation on a potential migration to the more powerful XML schema definition language (XSD) [172] may show significant improvements to the PSV's internal coherence.
- We also note that the GDE does not support all the feature provided by the PSV, e.g. it only allows to design two-party protocols and no security properties can be specified yet.
- The PSV itself is expected to be extended and improved to to contain all the (useful) information that can be found in current protocol specifications.
- Additional plugins exporting to different languages are obvious and required, in particular to generate executable code, e.g. C++.
- We aim to apply this tool to specific fields of increasing interest, i.e. Intelligent Transportation Systems and Vehicle to Grid. These fields involve experts in the many disciplines, and security is of paramount importance; MetaCP can bridge the communication gap between them.

F.4 Chapter Conclusion

Protocols are core components of any IoT deployment, and represent a key factor of consideration for any system designer. Both if a bespoke protocol is designed for a system or

an existing protocol is chosen, they need to be verified for security, and it is consequently important to be able to do so with ease; MetaCP helps accomplish this goal. Upon the successful design of a protocol, the user may now automatically use the design created by MetaCP (in PSV) to model the interactions of their system, and analyse its behaviours, aiding the possibilities for decision making and system evaluation. Furthermore this modelling representation can be used to specify system behaviour within an IDS. We showcase the translation rules than can go from the PSV to the LISA models discussed in the previous sections. Whilst this is far from a perfect interface it allows for some initial models to work from the decrease the amount of work required.