On Reducing Underutilization of Security Standards by Deriving Actionable Rules: An Application to IoT

Md Wasiuddin Pathan Shuvo

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Information Systems Security) at

Concordia University

Montréal, Québec, Canada

May 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Md Wasiuddin Pathan Shuvo**

Entitled: **On Reducing Underutilization of Security Standards by Deriving Actionable Rules: An Application to IoT**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Mohsen Ghafouri

_____ External Examiner
Dr. Paria Shirani

_____ Examiner
Dr. Mohsen Ghafouri

_____ Supervisor
Dr. Suryadipta Majumdar

Approved by _____
Dr. Abdessamad Ben Hamza, Chair
Department of Concordia Institute for Information Systems Engineering

_____ 2023

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

On Reducing Underutilization of Security Standards by Deriving Actionable Rules: An Application to IoT

Md Wasiuddin Pathan Shuvo

Even though there exist a number of security guidelines and recommendations from various worldwide standardization authorities (e.g., NIST, ISO, ENISA), it is evident from many of the recent attacks that these standards are not strictly followed in the implementation of real-world products. Furthermore, most security applications (e.g., monitoring and auditing) do not consider those standards as the basis of their security check. Therefore, regardless of continuous efforts in publishing security standards, they are still under-utilized in practice. Such under-utilization might be caused by the fact that existing security standards are intended more for high-level recommendations than for being readily adopted to automated security applications on the system-level data. Bridging this gap between high-level recommendations and low-level system implementations becomes extremely difficult, as a fully automated solution might suffer from high inaccuracy, whereas a fully manual approach might require tedious efforts. Therefore, in this thesis, we aim for a more practical solution by proposing a partially automated approach, where it automates the tedious tasks (e.g., summarizing long standard documents, and extracting device specifications) and relies on manual efforts from security experts to avoid mistakes in finalizing security rules. We apply our solution to IoT by implementing it with IoT-specific standards (NISTIR 8228) and smart home networks. We further demonstrate the actionability of our derived rules in three major applications: security auditing, Intrusion Detection systems (IDS), and secure application development.

Acknowledgments

I would like to express my deep gratitude to my supervisor, Dr. Suryadipta Majumdar, for his invaluable guidance, unwavering support, and endless encouragement throughout this research journey. I am incredibly lucky to be able to work under the close guidance of my supervisor, who inspired me with bright ideas, helpful comments, suggestions, patience, and insights that have contributed to the improvement of this work. His vast knowledge and expertise have helped me develop a better understanding of the subject matter, and his mentorship has been instrumental in shaping me into the researcher I am today.

I would also like to extend my appreciation to Dr. Paria Shirani, whose insightful comments and suggestions have greatly contributed to the quality of this work. Her feedback and recommendations have been immensely helpful in refining my ideas and improving the overall structure of this thesis.

I would like to acknowledge my family and friends for their unwavering love, support, and encouragement throughout this journey. Their constant support and motivation have been a source of strength for me, and I could not have achieved this without them.

Finally, I would like to express my gratitude to the department and university for providing me with the resources and opportunities that made this research possible. Thank you all for your invaluable contributions to my academic and personal growth.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Recent cyberattacks are typically caused by various safety and security threats that result from implementation flaws and insecure default configurations [2–8]. For instance, the Mirai botnet infecting millions of devices and conducting massive Distributed Denial of Service (DDoS) attacks on major services, e.g., Amazon, GitHub, and Netflix, mainly resulted from not following the best practices (e.g., latest versions of libraries and protocols, no weak passwords, etc.) [9]. Due to similar issues in implementing the best practices, several other recent attacks also lead to severe security and safety consequences, such as unauthorized access to smart homes [7], injecting fake voice commands to smart home devices and hubs [10], and health hazards to infants in a smart home [11]. As a result, the accountability and transparency of those devices and their operations often become questionable. This might be due to the fact that most security solutions (e.g., [11–14]) are not using standards as a basis for their security evaluation.

Several works (e.g., [11–20]) are addressing different security issues such as intrusion detection, device fingerprinting, application monitoring, and access control. However, none of those works focus on developing a generic approach to automatically define actionable security rules for verifying different system and device security. Moreover, none of them choose existing security standards as the basis of their security evaluation, which as a result endangers billions of devices and systems against many severe security threats (e.g., Mirai [9]).

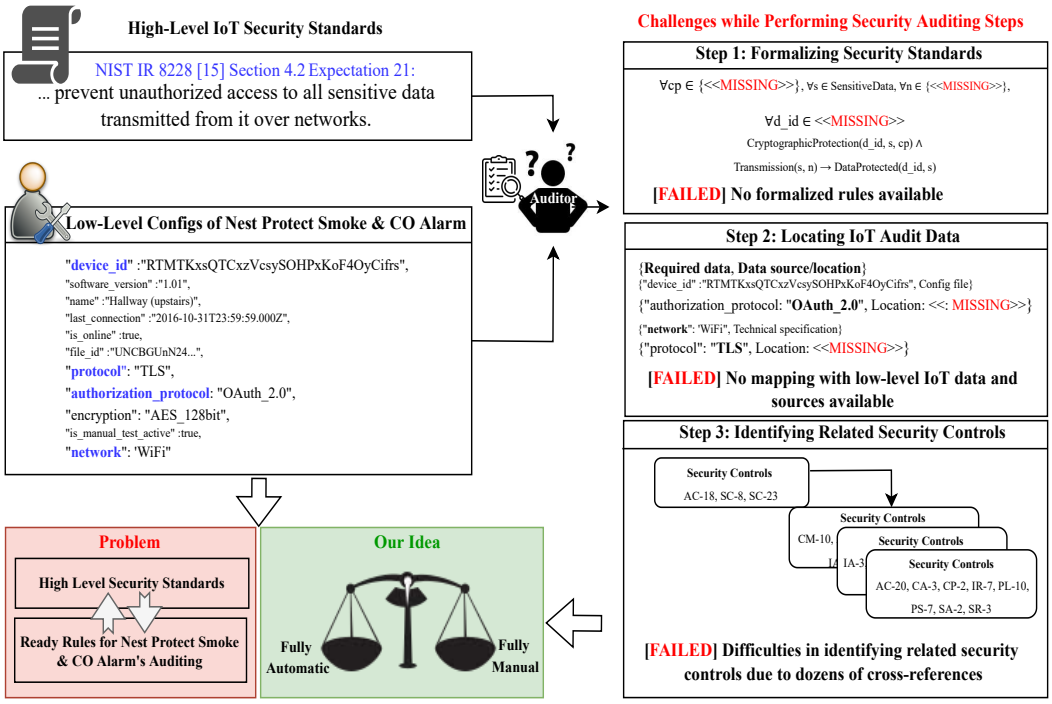

Figure 1.1: Motivating example depicting major challenges in converting high-level security standards to actionable security rules for security auditing of low-level IoT system implementation

One of the main reasons behind this under-utilization might be due to the high-level nature of most of those security standards (e.g., [1, 21, 22]) which renders additional overhead to adopt them in different security applications that typically operate on system-level data. Interpreting high-level recommendations and deriving actionable security rules for low-level system implementations becomes infeasible using any extreme solutions, i.e., a fully automated solution (which is less accurate) and a fully manual approach (which is tedious and error-prone).

1.2 Motivating Example

A motivating example is shown in Figure 1.1 where an Internet of Things (IoT) security standard, known as the National Institute of Standards and Technology Internal Report 8228 (NIST IR 8228), is directly used to perform security auditing (but failed, as explained later) of a smart home device (Nest Protect Smoke and CO Alarm [23]). Particularly, the left side of the figure shows typical inputs to an auditing tool: a "high-level" recommendation from NIST IR 8228 (top) and

"low-level" configurations from a Nest Protect Smoke and CO Alarm (middle). The right side depicts the challenges encountered while performing different security auditing steps (Steps 1-3). On the bottom left, we briefly illustrate the problem and our idea to solve it.

Specifically, this example depicts a scenario where an auditor aims at auditing a Nest Protect Smoke and CO Alarm device against the `Expectation 21` in Section 4.2 of NIST IR 8228 [1]. The expectation states: *"a device can prevent unauthorized access to all sensitive data transmitted from it over networks"*. On the other hand, configurations from a Nest Protect Smoke and CO Alarm include information about `device_id`, `software_version`, `protocol`, `network`, etc. While performing auditing using these inputs, an auditor encounters several challenges, as follows. (i) During Step 1 (for formalizing security standards), the allowed list of cryptographic protection (`cp`) methods, networks (`n`), and device IDs (`d_id`) are missing from the `Expectation 21` description in NIST IR 8228. During Step 2 (for locating audit data), the auditor cannot easily find the source of `authorization_protocol` and `protocol` in a Nest Protect Smoke and CO Alarm, even if she can locate others (e.g., `device_id`, `network`) from its configuration files or technical specifications. During Step 3 (for identifying related security controls), the auditor might struggle to link between various controls, such as the `AC-18` control refers to nine other controls: `CA-9`, `CM-7`, `IA-2`, `IA-3`, etc. Therefore, very likely, most of those auditing steps might fail, if not all.

The main problem is to address those challenges and allow interpreting high-level security standards and defining ready rules for auditing Nest Protect Smoke and CO Alarm. To that end, both fully automated and fully manual solutions might also fail the auditing process because full automation might change the semantics of the original recommendations, and relying only on manual effort would be time-consuming and error-prone. Therefore, our idea is to balance between those two extreme approaches and find a practical solution to derive actionable rules for IoT devices by proposing a semi-automated approach.

1.3 Problem Statement

There are several challenges in deriving actionable rules from security standards.

- Firstly, most existing security standards are provided at high-level without any clear mapping between those recommendations with the actual design and implementation of IoT products available in the market. Thus, it becomes almost infeasible to use them to conduct in security applications (that require more low-level granular security rules).
- Secondly, those standards significantly differ from each other in terms of scope, objective, and level of descriptions. Therefore, interpreting the security recommendations from each standard for deriving rules becomes non-trivial.
- Thirdly, among those standards, there are conflicting recommendations. As a result, a systematic analysis of those high-level recommendations is required to interpret them and resolve their conflicts, before deriving actionable rules.
- Fourthly, the knowledge and expertise required from a target audience of these security standards is not explicitly specified, and the guidelines are not crafted as actionable for the target audience [24].
- Lastly, security standards contain too many different types of interrelated guidance on a single subject [25], which are frequently cross-referenced to dozens of other security documents. This can make the recommendations challenging to follow and fully utilize at times.

We will address these challenges in Chapter 4.

1.4 Thesis Contribution

In this thesis, we propose a partially automated approach (which appears to be more practical) to derive actionable security rules from various security standards and show its application to IoT. More specifically, first, we conduct a study on major security standards such as [1, 21, 22, 26–28]. Second, we extract IoT device-specific information from product specifications, API documentation, and configuration files to build a knowledge base. Third, we leverage Natural Language Processing (NLP) techniques for summarizing and a fine-tuned Named Entity Recognition (NER) model to extract key recommendations from those security controls. Fourth, we instantiate each recommendation as a security rule, expressed in formal language, on various IoT systems by collecting

IoT log data, API, and configuration files. Due to the criticality of security applications, our derived security rules are preferred to be examined by a security expert to assure their correctness. Finally, our derived rules are applied to various security applications, such as security auditing, Intrusion Detection Systems (IDS), and secure application development.

The main contributions of this thesis are as follows.

- This thesis work is the first effort to reduce the underutilization of security standards for emerging technologies (e.g., IoT). Our effort can potentially help to safeguard those new technologies in various aspects (e.g., enabling security mechanisms, allowing secure development) as well as improve the existing security standards by identifying their inconsistencies and discrepancies with the target technologies.

- As per our knowledge, this is the first effort to derive actionable security rules from IoT security standards. This actionability of derived rules is demonstrated by integrating our approach in a smart home ecosystem and applying those rules to various security applications, i.e., security auditing, IDS, and secure application development.

- Our experimental results further show the effectiveness of our solution in reducing the manual efforts (e.g., 50% effort reduction on average) and adaptability of our derived rules for security auditing (where 5,000 smart home devices can be audited within ten seconds).

The thesis is organized as follows. Chapter 2 reviews related works and compares them with our approach. Chapter 3 describes the preliminaries and challenges. Chapter 4 details our methodology. Chapter 5 provides the application to different security mechanisms. Chapter 6 shows the implementation details, and Chapter 7 presents the experimental results. Chapter 8 discusses different aspects of our approach. Chapter 9 concludes the thesis by providing future research directions.

Chapter 2

Related Works

In this chapter, we compare our work to existing IoT security approaches of varying scope and objectives, concluding that it is complementary to those efforts.

2.1 Rule-based IoT Security Solution

Recent research has looked into various aspects of IoT security and safety. Fung et al. [29] introduce a policy-sharing model for the IoT environment where users can define different policies and share them with their friends through their social connections. They also track the reputation of policies based on the feedback of different policies adopted by users without the need for central facilities. PFIREWALL [14] automatically generates data minimization policies to control the data flow by filtering communication between devices and platforms of IoT systems to protect users' privacy. They implement their solution on IoT wireless communication protocols, and results show that it reduces data leakage significantly without modifying home automation. IoTSafe [11] detects runtime physical interactions of the IoT environment by performing static analysis and dynamic testing to enforce security policies in real-time and prevent unsafe states with high accuracy. Soteria [18] verifies the safety and security policies of IoT platforms by performing static code analysis of IoT applications. On the other hand, IoTGuard [13] is a dynamic safety and security policy enforcement system which prevents unsafe states in IoT systems by implementing code instrumentation to

extract the app's run-time information and building a dynamic model. Both of them utilize user-defined policies to detect violations in the IoT environment. Nespoli et al. [19] propose a system that detects existing vulnerabilities in IoT systems as well as dynamically adapts to surrounding IoT devices and services based on rules to protect against malicious network attacks. Dome et al. [20] propose an IoT rules generation framework that collects IoT sensor data as training data to find patterns and meaningful features. Then they utilized the Random-Forest model to convert those patterns into rules, build a threat prediction model, and monitor rules violations. Majumdar et al. [30], [31] conduct security auditing leveraging formal techniques in cloud platforms. Madi et al. [32] also carried out security auditing on a cloud platform by proposing an auditing framework for OpenStack. Instead of adopting rules and policies from security standards, most of these works develop their own policies.

Another line of research focuses on access control, monitoring, and intrusion detection of IoT systems. Jia et al. [17] perform a study on all the existing attacks on IoT platforms by reproducing them and categorizing them based on the lifecycle and adversary techniques. Following that, ContexIoT is being created as a context-based permission system to ensure contextual integrity for various IoT platforms. SmartAuth [33] performs static analysis to extract security-related information from IoT applications to devise authorization mechanisms to mitigate over-privileged problems in IoT systems. HoMonit [12] extracts Deterministic Finite Automaton (DFA) from the source code of SmartApps and utilizes side-channel techniques to monitor encrypted traffic of IoT systems to detect anomalies in the SmartThings platform. IoTArgos [15], is a multi-layer security monitoring system aided by supervised and unsupervised machine learning algorithms utilizing the network data collected from various IoT devices through programmable home routers to detect intrusions and anomalies in IoT platforms. Anthi et al. [16] develop a multi-layer intrusion detection system for IoT devices that can detect various network-based attacks. Their system profiles normal IoT behaviour and classifies them, then detects attacks against the IoT network as well as classifies attack types. In contrast, our work can complement their works by providing actionable security rules which can be utilized in various security applications, particularly the ongoing research on IoT security auditing.

2.2 Association Rule Mining based Works

P. Lou et al. [34] propose a method for obtaining association rules on multi-source logs based on the Adaptive Miner algorithm to provide critical information for cyber intrusion detection and assist non-experts in conducting security problem investigations in cloud computing platforms. The experiments with 12 million alerts from 34 intrusion detection systems in three organizations show that their predictive blacklist minimizes the alerts of events (3%) of the raw data. Ozawa et al. [35] use association rule mining to discover the regularities in darknet data. An experiment on the darknet analysis of a large set of TCP SYN packets collected from 1 July 2016 to 31 July 2018 with the NICT /16 darknet sensor shows that this analysis can be used to track the activities of the attacker. They are able to convincingly demonstrate that Mirai and Hajime are in direct competition with one another in timelines with generated or disappeared rules. Husak et al. [36] use sequential rule mining to analyze intrusion detection alerts and to predict security events for creating a predictive blacklist. Safara et al. [37] use an association rule mining algorithm to extract appropriate features from the raw data and then use the features for classifying the data and detecting anomalies in communication networks. They use the combination of artificial neural network and AdaBoost classification algorithms to classify anomalous data in KDD Cup 99 dataset. They also compare their result with existing models, and their models showed better performance than others.

Xu et al. [38] propose an Attribute-based access control (ABAC) policy mining algorithm which improves the policy by merging and simplifying candidate rules, then selects the highest-quality candidate rules for inclusion in the generated policy. They show the effectiveness of their proposed algorithm by experimenting with both the sample policies and synthetic policies. Sanders et al. [39] use the rule mining approach to analyze systems' audit logs for automatically generating ABAC policies that minimize both under-privilege and over-privilege and propose a policy scoring algorithm for evaluating ABAC policies. They experiment with Amazon Web Service (AWS) audit log events data and show the effectiveness of their algorithm.

Table 2.1: Comparing existing solutions with ours. (●), (○), (-), and (NA) mean supported, partially supported, not supported, and not applicable, respectively.

Proposals	**Methods**	**Coverage**		**Features**						
		Environment	**Objective**	**Knowledge base**	**First Order Logic**	**Automation in Rule Derivation**	**Expressiveness**	**Automatic System**	**Runtime Enforcement**	**Security Standards**
ContextIoT [17]	Custom Algorithm	IoT	Access Control	NA	–	–	NA	○	●	–
Soteria [18]	Static Analysis	IoT	Intrusion Detection	–	–	–	NA	○	–	–
IoTGuard [13]	Dynamic Analysis	IoT	Intrusion Detection	–	–	–	NA	●	●	–
Majumdar et al. [30]	Formal Method	Cloud	Auditing	–	●	–	●	○	–	○
Madi et al. [32]	Formal Method	Cloud	Auditing	–	●	–	●	NA	–	○
Majumdar et al. [31]	Formal Method	Cloud	Auditing	–	●	–	●	NA	●	○
Homonit [12]	Custom Algorithm	IoT	Monitoring System	NA	–	–	–	–	–	–
IoTSafe [11]	Static and Dynamic	IoT	Intrusion Detection	NA	–	–	–	●	–	–
PFIREWALL [14]	Custom Algorithm	IoT	Access Control	NA	–	○	○	●	●	–
This Work	Formal Method	IoT	Auditing, Secure development, etc.	●	●	○	●	○	○	●

2.3 Comparative Study

Table 2.1 summarizes a comparative study of existing works. The first two columns enlist existing works and their methods, respectively. The next two columns compare the coverage, such as the supported environment (IoT, cloud) and main objectives (auditing, intrusion detection). The remaining columns compare these works based on different features, i.e., knowledge-base, first-order logic, automatic rule derivation, expressiveness, automatic system, run-time enforcement, and utilization of security standards. In our study, four different symbols are used to show the amount of support for a particular feature or functionality. These symbols are a complete circle (●), an empty circle (○), a hyphen (-), and (NA), which stands for supported, partially supported, not supported, and not applicable. These symbols are used to classify the amount of support for various system capabilities or functionalities. A (●) symbol indicates that the work fully supports the feature, a (○) symbol indicates that the work only partially supports our feature, a (-) symbol indicates that the work does not support any specific feature, and a (NA) symbol indicates that the work is not

applicable to that feature. In summary, our work mainly differs from other works as follows. Firstly, we only propose an approach to derive actionable security rules from existing IoT security standards. Secondly, we build a knowledge base for both IoT standards and IoT devices that can be utilized in other related research. Finally, our derived security rules can directly be used for various security applications.

Chapter 3

Preliminaries

To keep our discussion more concrete, the rest of the thesis will be on the scope of IoT standards and smart home networks. In this chapter, we provide background information on security standards, discuss how the IoT differs from traditional Information Technology (IT) devices and what that means for security, and give an overview of the major security standards. Finally, we present the preliminary results of our research that are later utilized to decide design choices.

3.1 Background

In this section, we provide an overview of security standards and explore how the IoT differs from traditional IT devices.

3.1.1 Security Standards

Security Standards describe the best practices from several security documents, organizations, and publications. A security standard is designed as a framework for an organization requiring stringent security measures. These standards can be developed by a variety of organizations, including governments, industry associations, and standards bodies. The procedure for developing security standards is often a collaborative one that makes use of the knowledge of many different stakeholders. These stakeholders may include security professionals, business executives, public servants, and academic researchers. Each security standard contains several *security controls*, which describe

the protection capabilities for particular security objectives of an organization and reflect the protection needs of organizational stakeholders. *Security expectations* are the expected outcomes from a security control to ensure the secure operation of a system.

3.1.2 IoT Devices vs. Conventional IT Devices

In the realm of technology, the Internet of Things (IoT) and conventional IT/computer devices represent two distinct paradigms. IoT is a very new concept that is quickly gaining popularity. While conventional IT devices have been available for many years and are commonly utilized, IoT is a relatively new phenomenon. As a result, managing risks for IoT devices as a whole, including consumer, enterprise, and industrial IoT devices, differs from managing risks for traditional IT devices [1].

The IoT is a network of interconnected physical objects, such as cars, appliances, and other household things, that are equipped with electronics, software, sensors, and connectivity. IoT aims to make it possible for these devices to connect with one another and carry out activities autonomously. The objective is to build a network of intelligent devices that can track, assess, and react to changes in the environment and user behavior. According to the NIST IR 8228 security standard [1], there are several distinctions between IoT and traditional IT devices in terms of security, and we have listed the most significant ones below.

Cost Effective. The price of IoT devices compared to conventional IT devices is one of their main disparities. IoT devices are made to be inexpensive, compact, and effective, as opposed to traditional IT equipment, which is typically bigger, more complicated, and more expensive. The components utilized in each device, the manufacturing method, and the volume of the production are only a few of the reasons for this pricing disparity. It is crucial to take security considerations into account when comparing the cost of IoT devices to traditional IT devices. IoT devices often cost less than traditional IT equipment, but they could also be less secure because of their reduced cost and more straightforward design. Manufacturers may choose generic hardware or open-source software to cut costs while still including the necessary functionality in their products.

Lack of Administrative Capabilities and Interface. The absence of administration capabilities

and interfaces is one of the biggest issues with IoT devices. Administrators may find it challenging to properly control the operating system, apps, and firmware of IoT devices throughout their lifespan as a result. With IoT devices, for instance, administrators might not be able to purchase, check the integrity of, install, configure, save, retrieve, execute, stop, delete, swap out, update, and patch software. Also, when a bad thing happens, like a power outage or a lack of network connectivity, the software on IoT devices might be automatically modified. Moreover, it can be difficult for users to connect with and manage some IoT devices because they lack application or human user interfaces for device use and management. While such interfaces do exist, they cannot give the complete capability offered by conventional IT equipment. This may lead to problems, including the inability to inform users that their personally identifiable information is being processed by an IoT device, making it challenging to get meaningful consent for this processing. The lack of widely agreed standards for IoT application interfaces, which can obstruct interoperability across IoT devices, is another issue.

IoT Device Management Challenges. The management of IoT devices poses several challenges due to their large numbers and diverse software and hardware. Firstly, managing IoT devices at scale is challenging since the majority lack standardized procedures for centralized management. Administrators may not be able to fully manage the firmware, operating system, and apps of an IoT device during its lifecycle due to the lack of management options. Secondly, the extensive range of software utilized by IoT devices, such as firmware, common and real-time operating systems, and applications, makes software management extremely difficult during their entire lifecycle. This has an impact on things like patch management and configuration. Thirdly, the possibility that IoT device hardware will not be repairable is a substantial additional management burden. The inability of the hardware to be repaired, modified, or internally inspected makes it challenging to troubleshoot or replace faulty components. Finally, the inventory, registration, and provisioning of IoT devices introduced into an organization might not be done through the standard IT processes. Administrators struggle to keep track of all the IoT devices on the network due to a lack of inventory capabilities, which makes it impossible to adequately secure them.

Varied Expectation of Device Lifespan. A manufacturer might plan for a specific IoT device

to be utilized for a short period of time before being discarded. An organization that buys such an item might desire to utilize it for a longer period of time, but the manufacturer might decide to stop providing support for it (such as by no longer issuing fixes for known vulnerabilities) or supply chain constraints might prevent them from doing so (e.g., supplier no longer releases patches for a particular IoT device component). Although the issue of varying lifespan expectancies is neither new nor unique to the IoT, it may be particularly significant for some IoT devices due to the possible dangers to safety, dependability, and other factors when operating devices past their intended lifespans.

Multi-user and Heterogeneous Ownership Challenges. The existence of multi-user devices, where numerous users can access and control the device without clearly recognizing their authorization, further complicates the security of IoT devices. Family members, housemates, or coworkers may share these gadgets, and each user may have varying degrees of access to and control over them. This makes it difficult to establish unambiguous ownership and accountability for the security of the device because different users may have varying degrees of responsibility. Another issue is heterogeneous ownership which describes the wide range of stakeholders, including manufacturers, independent sellers, service providers, and end users, who may own and manage IoT devices. This makes it more difficult to manage the security of IoT devices, especially when data must be shared with cloud-based services or when upkeep is necessary. Certain maintenance procedures are solely authorized by manufacturers, and attempting to carry them out may void the device's warranty.

Limitations of Cybersecurity and Privacy Capabilities. The wide variety of cybersecurity and privacy features generally included in traditional IT devices are not or cannot be supported by many IoT devices. For instance, a "black box" IoT device could not be able to provide companies with access to its logs or might not record cybersecurity and privacy events. Pre-market capabilities for IoT devices might not be strong enough or work poorly; for instance, utilizing strong encryption and mutual authentication to protect communications might result in unacceptably long delays. Many IoT devices cannot be equipped with post-market features. Also, it is possible that current pre- and post-market capabilities will not be able to scale to meet IoT requirements. For instance, a network-based cybersecurity appliance for traditional IT devices might be unable to handle the volume of

Table 3.1: Summary of different IoT security standards

Security Standard	Purpose	Targeted to	Scope	# of pages
NIST IR 8228 [1]	Security and privacy risk management	Users	All kinds of IoT Devices	44
NIST IR 8259 [26]	Building secure IoT device	Manufacturers	All kinds of IoT Devices	36
ENISA [21]	Recommendation for baseline security	Users and manufacturers	All kinds of IoT Devices	103
ETSI EN 303 645 [27]	Consumer IoT security	Manufacturers	All kinds of IoT Devices	34
OWASP [22]	Secure building and usage of IoT	Manufacturer, developers and consumers	All kinds of IoT Devices	12
UK Govt [28]	Improve the security of consumer IoT	Manufacturers, developers and service providers	Smart homes and smart wearables	24

network traffic and generate significant data from many IoT devices.

3.2 Review of Major IoT Security Standards

To identify unique challenges in deriving actionable security rules, we review several major IoT security standards (as summarized in Table 3.1).

NIST IR 8228 [1]. It is an internal report published by the National Institute of Standards and Technology (NIST), a federal agency of the US government. The goal of this report is to assist users in better understanding and managing the cybersecurity and privacy risks associated with individual IoT devices across their life cycles. Particularly, this 44-page report outlines three high-level risk mitigation goals for the security of IoT devices, and each risk mitigation goal is further divided into several risk mitigation areas. Moreover, NIST IR 8228 has listed 25 expectations along with 49 challenges to achieve those expectations and their mapping with the NIST SP 800-53r5 [40] for mitigating security and privacy risks in IoT systems.

NIST IR 8259 [26]. It is also an internal report from NIST, which is intended for IoT device manufacturers to assist them improve the security of their IoT products. This 36-page report describes six cybersecurity activities which are broken down into 65 questions that a device manufacturer should

consider to secure their IoT devices. During the pre-market phase, the manufacturer's decisions and actions are primarily influenced by the first four of those six activities, whereas the remaining two activities are primarily for the post-market phase of an IoT device. Each activity's questions are open-ended and allow for speculation, which may cause the manufacturers to make a perplexing choice that is unsuitable for use as actionable rules.

ENISA Baseline Security Recommendation for IoT [21]. The aim of this report, published by the European Union Agency for Cybersecurity (ENISA), is to create baseline cybersecurity guidelines for both consumers and IoT manufacturers, with a particular focus on critical infrastructures. This report covers many domains of IoT (e.g., smart homes, smart cities, smart grids, etc.), and it is intended for IoT software developers, manufacturers, information security experts, security solution architects, etc. There are 83 security measures outlined, divided into 11 security domains that cover every IoT ecosystem horizontally, in this 103-page report. However, all of these security measures cannot be used as actionable security rules as they are insufficiently specific.

ETSI EN 303 645 - V2.1.1 [27]. The European Telecommunications Standards Institute European Standard 303 645 (ETSI EN 303 645) establishes a security baseline while covering all consumer IoT devices. Although the target audience of this article is primarily manufacturers of various IoT devices, it also aims to assist IoT users. This 34-page document has 67 provisions with examples divided into 13 high-level recommendations and refers to multiple external documents for further technical details. With a focus on technical controls, the ETSI document has specific guidelines, but technical details are insufficient to be used for actionable security rules [24].

OWASP IoT Security Guidance [22]. The Open Worldwide Application Security Project (OWASP) Internet of Things Project has released the OWASP IoT top ten lists of IoT vulnerabilities in an effort to help manufacturers, developers, and consumers better understand IoT security risks and take appropriate mitigation measures. The specialty of this project is its simplicity, where they avoid separating guidelines for different stakeholders. That is why it is the shortest security guideline, with only 12 pages, among the security standards that we reviewed. This report lists the top 10 recommendations to secure IoT devices without providing detailed or specific steps on how to follow those recommendations in real-world product development.

Code of Practice by the UK Government [28]. This security guideline is developed by the UK Department for Digital, Culture, Media, and Sport in conjunction with the National Cyber Security Centre and follows engagement with industry, consumer associations, and academia. Its goal is to provide guidelines to all organizations involved in developing, manufacturing, and retailing consumer IoT products on achieving a secure-by-design approach. It lists 13 high-level security outcomes that are to be reached by following the recommendations in this 24-page report. In spite of those outcomes, this guideline gives stakeholders the liberty to apply each guideline on their own terms instead of providing concrete ways to do so [24].

3.3 Preliminary Studies

In this section, we will review the Natural Language Processing (NLP) techniques that we have used in our work thus far. We first experiment with various popular topic modeling strategies. The statistical technique known as "topic modeling" can be used to determine the subject of diverse texts. NLP uses this unsupervised machine learning approach to recognize hidden semantic patterns in text corpora [41]. There are various methods for extracting a topic from a text. Latent Dirichlet Allocation (LDA) [42] and Non-negative Matrix Factorization (NMF) [43] are two of them that are frequently used to extract topics from various texts, and we have employed both of them in this research. In addition, we have used a variety of text embedding methods, such as Embeddings from Language Models (ELMo) [44] and Bidirectional Encoder Representations from Transformers (BERT) [45], to summarize security standards.

3.3.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a topic modeling technique used to identify hidden topics in a corpus of documents. The method makes the assumption that each document consists of a variety of themes, each of which is distributed over a set of words. Each word in a document is connected to a certain topic by the LDA algorithm, which represents each document as a mixture of topics. The LDA model then calculates the probability distributions of the themes and words inside each subject in the corpus. LDA is employed in our preliminary study to extract the most important

recommendations from the security standards.

3.3.2 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is a matrix factorization approach used to find the underlying structure in a dataset. The method makes the assumption that the data can be expressed as a linearly non-negative combination of a number of basis vectors. NMF can recognize the underlying topics in a corpus of documents, making it particularly helpful for text data. In our preliminary investigation, NMF is used in a manner similar to LDA to draw out the most key recommendations from the security standards.

3.3.3 ELMo

ELMo is a deep contextualized word embedding technique. In order to understand the context in which words appear in a corpus of text, it employs a deep bidirectional language model. The vectors or embeddings in ELMo are produced by a bidirectional long short-term memory (LSTM) model that has been trained on a sizable text corpus using a coupled language model (LM) objective. By feeding the whole input sentence into the two-layer biLMs with character convolutions, ELMo representations are able to accurately reflect the context within the phrase. Many studies show that ELMo is simple to integrate into a variety of language understanding tasks, including textual entailment, question answering, summarising, and sentiment analysis, and may greatly advance the state-of-the-art of diverse tasks by lowering 20% relative error [44]. In our early analysis, we explored the effectiveness of using ELMo in a summarization task and compares its performance with that of BERT.

3.3.4 BERT

BERT is another cutting-edge method for natural language processing developed by Google. ELMo features an LSTM-based architecture, but BERT offers a deeply bidirectional transformer-based pre-trained model. Two unsupervised tasks, Next Sentence Prediction and Masked Language Model (MLM), are used to pre-train BERT (Next sentence prediction). BERT was pre-trained using a sizable unlabeled text corpus, which allowed the model to comprehend language more deeply. For

the pre-training corpus, 800 million words from BooksCorpus [46] and 2,500 million words from English Wikipedia were employed. With just one additional output layer, BERT may be fine-tuned to produce state-of-the-art outcomes in a variety of NLP tasks. BERT has already achieved the best results in eleven NLP tasks. As BERT has been pre-trained on a large amount of text, making it proficient in understanding the nuances of language, making it a suitable choice for summarizing security controls.

3.3.5 Key Findings

Based on initial experiments on different relevant NLP techniques, we find that topic modeling techniques, such as LDA and NMF, can identify the topics of security controls, but they cannot extract the security values necessary for constructing security rules. Therefore, we shifted our focus to fine-tuning the pre-trained language model to extract security values from security controls, which is essential for our proposed methodology. Moreover, we have conducted experiments to compare the effectiveness of BERT and ELMo in summarizing security controls (as reported in Chapter 7). Our experimental results show that BERT outperformed ELMo in extracting the most crucial information from security controls. However, ELMo needs more fine-tuning to be more effective in summarizing security controls.

Chapter 4

Methodology

This chapter first provides an overview of our proposed methodology and then elaborates on each step.

4.1 Overview

The overview of the proposed methodology is shown in Figure 4.1. The inputs to our system are originated from security standards (e.g., their description) and logs and configurations from a target system (e.g., IoT, clouds, networks). Our approach is divided into two primary phases: (i) *building a knowledge base*, and (ii) *defining actionable security rules*. More specifically, during the first phase (elaborated in Section 4.2), we map various security standard recommendations with their controls in NIST SP 800-53r5 [40] and annotate those mappings (Step 1.1). Then, we collect various IoT device-specific information to construct *structural knowledge* base (e.g., their sensors and actuators) and *functional knowledge* base (e.g., their network interfaces) (Step 1.2). During the second phase (elaborated in Section 4.3), we summarize the security controls, extract values from different summarized controls, and derive security rules for that control, which will be inspected by an expert (Step 2.1). Afterwards, we instantiate the derived security rules with device-specific information stored in the structural and functional knowledge bases and interpret them in a formal language (Step 2.2). In the figure, for both phases, we indicate if a step is fully automated (A), semi-automated (SA), or manual (M); their rationale is detailed in Section 4.5. Finally, we demonstrate

the applicability of our approach in security applications such as security auditing (in Section 5). The details of each phase are described as follows.

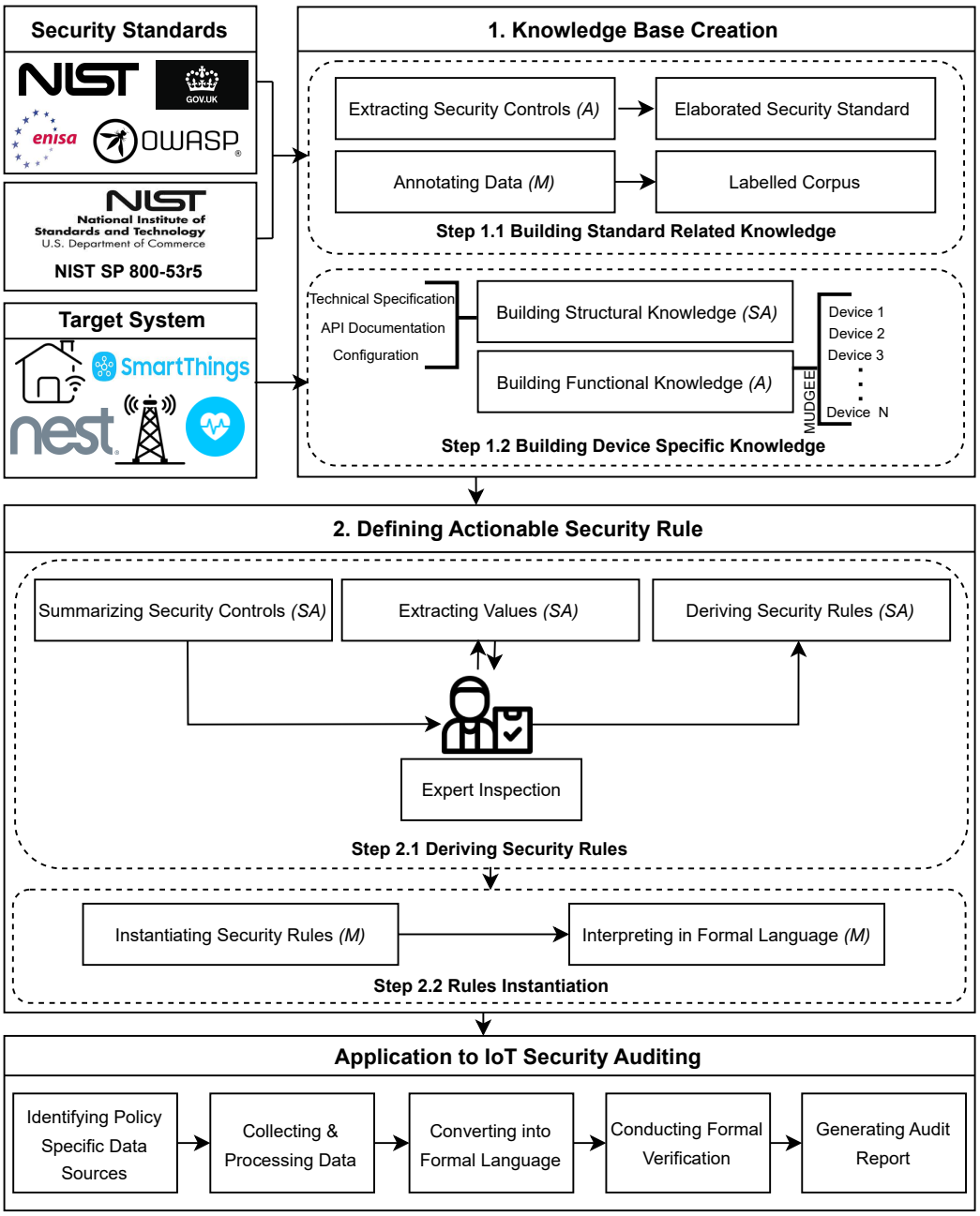

Figure 4.1: An overview of our methodology (where (A): fully automated step, (SA): semi-automated step, and (M): manual step)

4.2 Knowledge Base Creation

The *knowledge base* is created for both security standards and devices as follows.

4.2.1 Building Standard Related Knowledge

The goal of creating a knowledge base of security standards is to centralize all IoT security standards and their corresponding security controls (which provide preventive measures to mitigate a particular security issue in a system) from NIST SP 800-53r5 [40] (which defines security controls for IT in general) to be used for actionable rule derivation. There are application-specific recommendations, such as those found in NIST IR 8228, which provide security advice for IoT devices, and general security implementation guidelines in NIST 800-53r5, which are application agnostic. In our knowledge base, we simply merge them to provide more insight on how to implement an application-specific security recommendation using generic security implementation guidelines. Note that security control contains multiple sub-controls, each with a name and discussion, which either add functionality or specificity to a base control or increase the strength of a base control by further clarifying the technicalities. Figure 4.2 demonstrates the development and arrangement of our elaborated security standard from the referred security controls. To this end, we first extract security expectations and their mappings to security controls specified in each expectation, and then we extract corresponding security controls from NIST SP 800-53r5 to complete the mappings and build the elaborated security standards organized by variables, expectations, controls, sub-controls, and discussions. Afterwards, to further understand a control, we extract values and attributes from each security control. To that end, we first manually annotate the security control values based on the answers to the following three questions: (i) *Do the values accomplish a particular task?* (ii) *Are the procedures to complete this task known?* (iii) *Is it possible to implement a control technically?* After value annotation, we consider security sub-controls as our attributes and accordingly annotate them. Second, we train a Named Entity Recognition (NER) [47] model with annotated security controls and extract both values and attributes by utilizing the learned model.

Example 1. The `Expectation 21` from NIST IR 8228 refers to the security controls `SC-8`, `SC-23`, and `AC-18` in NIST SP 800-53r5. We first extract these three security controls and their

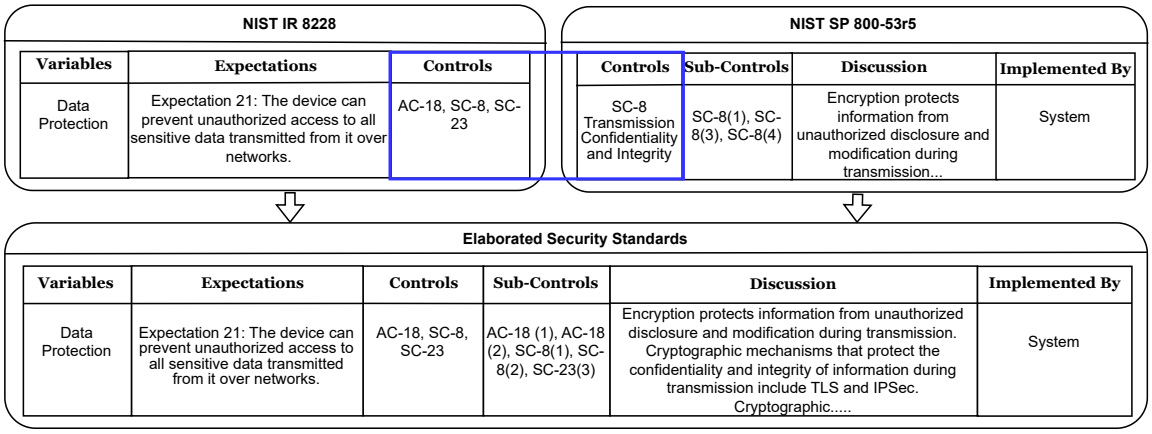

Figure 4.2: Development and Arrangement of Elaborated Security Standards

sub-controls with their discussions from NIST SP 800-53r5. Then, we create our elaborated security standards, which are arranged by variables, controls, sub-controls, and their discussions. In Figure 4.4, the first box contains the variable highlighted in red along with the `Expectation 21`. In the second box, we annotate security sub-control (e.g., cryptographic protection) as an attribute. Additionally, the third box contains the summarized security control, where the values in red are the results of our annotation after meeting all the above-mentioned criteria.

4.2.2 Building Device Specific Knowledge

To instantiate derived security rules that are specific to IoT devices, it is essential to have the knowledge of both their *structural* (e.g., their sensors and actuators) and *functional* (e.g., their network behaviour) characteristics. The *structural knowledge* of an IoT device includes different capabilities of its sensors and actuators, which are derived from the manufacture design specifications of different IoT devices. For this purpose, we leverage the approach proposed by Dolan et al. [8] as follows. First, we gather all the technical information that provided on a device's website. Second, we extract those information from the device's API documentation that describes API calls to change system states. Third, we gather essential information from IoT device configuration files, which are publicly accessible and include all essential device characteristics [8]. The *functional knowledge* of an IoT device includes its network behaviors that can be captured through manufacturer usage descriptor (MUD) (i.e., a framework by IETF for formally describing the network behaviour [48])

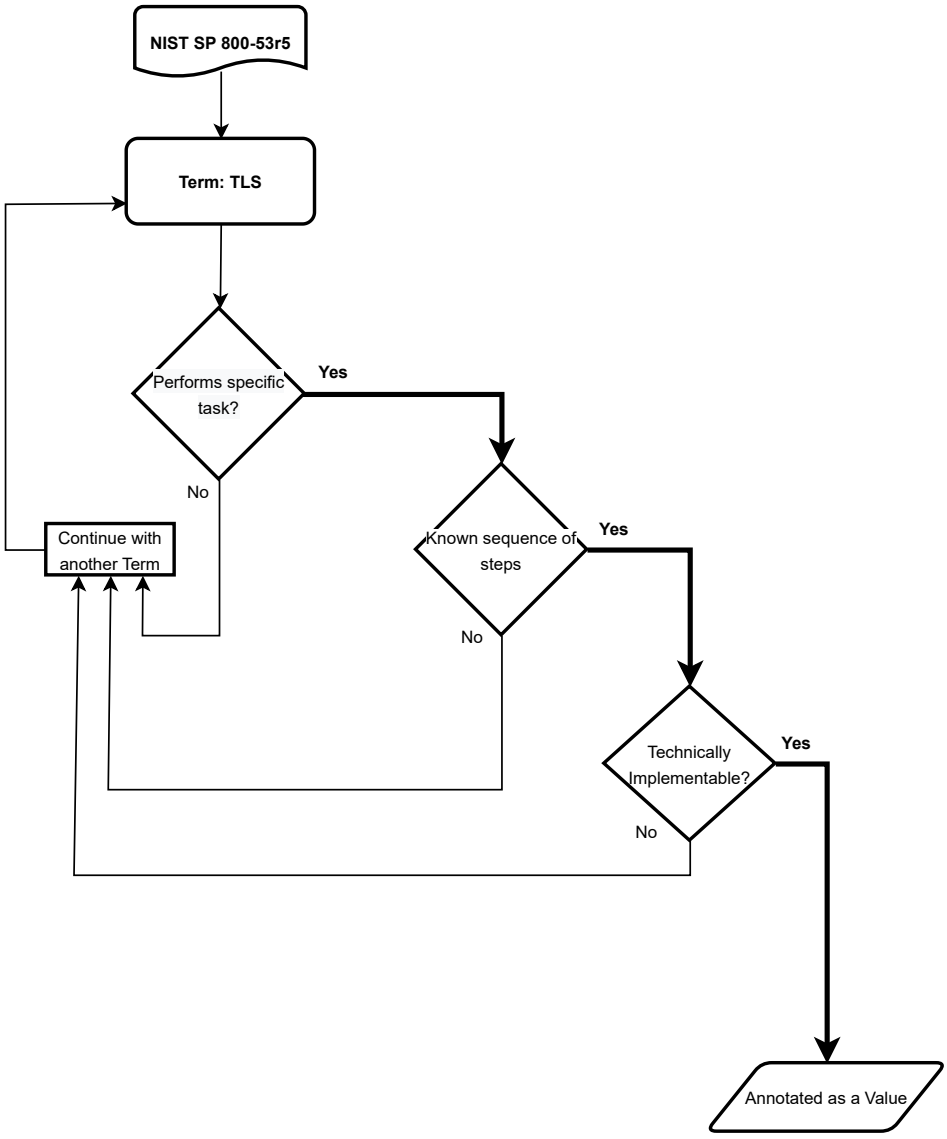

Figure 4.3: Labeling Values

profiles using MUDGEE [49]. MUD provides the list of all the protocols and ports that are used by IoT devices to communicate over the network.

Example 2. Given the Nest Protect Smoke and CO Alarm Sensor obtained from Nest Protect technical specs [23] and API documentation [50], obtained structural knowledge is: "*Software version:* `4.0`*; Device Unique Identifier:* `peyiJNo0IldT2YlIVtYaGQ`*; is_online:* `true`*; Read Permission:* `Enabled/Disabled`*; last_connection:* `2016-10-31T23:59:59.000Z`*; is_online:* `true`*; battery_health:* `ok`*; co_alarm_state:* `ok`*; smoke_alarm_state:* `ok`*; is_manual_test_active:*

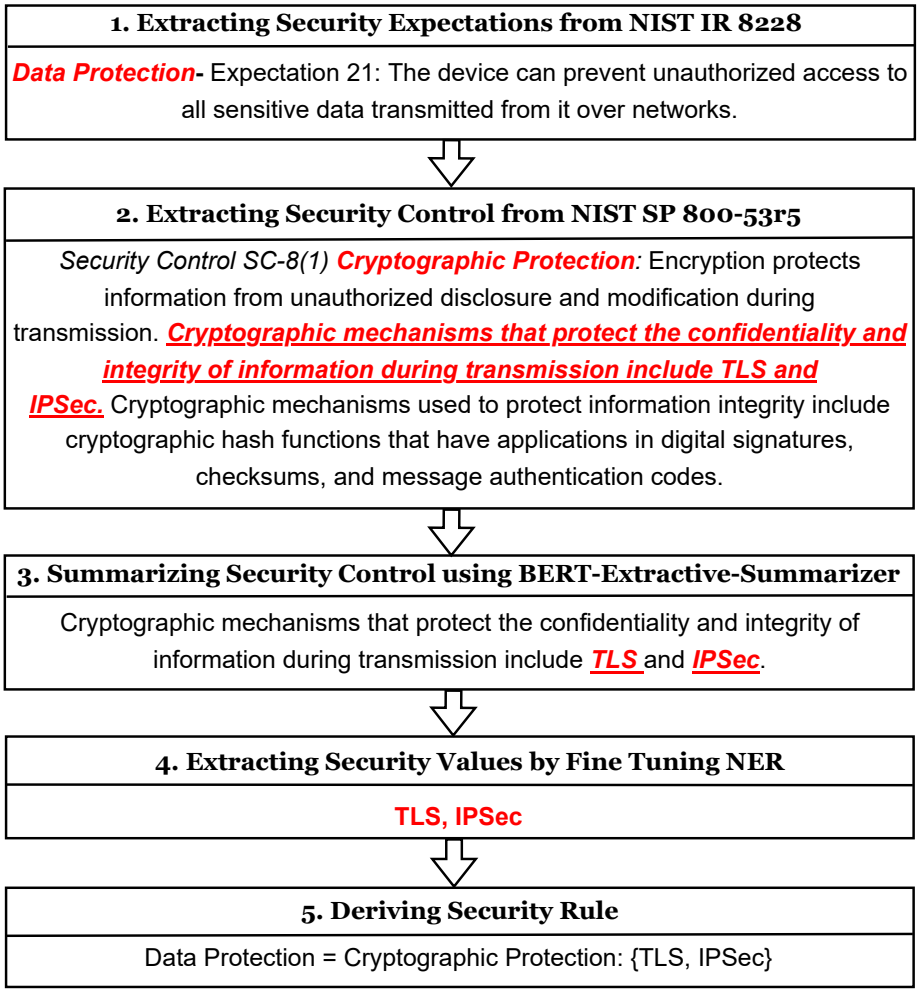

Figure 4.4: An illustration of automatically derived security rules. The `Expectation 21` from NIST IR 8228 is shown in the first box, then the security sub-control is displayed in the second box, and the summarised form of it is presented in red. The red-underlined terms in the third box's summary of the security sub-control represent values that were retrieved using the NER model displayed in the fourth box. The final box displays our automatically derived security rule.

`true`". The corresponding functional knowledge obtained by using MUDGEE is: "*IP Protocols:* `TCP, UDP, HOPOPT, IPv6-ICMP`; *and Ports:* `443, 11095, 53, 67`".

4.3 Defining Actionable Security Rules

We describe how we define actionable rules in the following.

4.3.1 Summarizing Security Controls

As there are many security controls and sub-controls with detailed descriptions of their security recommendations, we summarize them from the standard knowledge base (built in Section 4.2) utilizing BERT-Extractive-Summarizer [51], which is a BERT-based summarizing package. BERT is the state-of-the-art word embedding technique that is bi-directionally trained and can have a deeper sense of language context. BERT's extractive summarizing approach evaluates each sentence's comprehension and significance to the text and then delivers the most crucial segments. Thus, in this work, we opt for the extractive summarization technique instead of abstractive summarization, which changes the semantics of the recommendations due to newly generated words and phrases. BERT is used by a python-based RESTful service for text embedding, and for summary selection, KMeans clustering is used to identify sentences closest to the centroid [51]. Text summarization is still an ongoing topic in NLP research to achieve a competitive accuracy to that of a human [52]. Due to this factor, our generated summaries need to be reviewed by experts to ensure their correctness.

4.3.2 Extracting Values

Before deriving security rules, we extract attributes and values from the security controls. We use Named Entity Recognition (NER), an NLP approach, to extract values from the security sub-control's discussion. As we only need to extract two types of entities from the security controls, we annotate our values and attributes as described in Section 4.2.1 in order to use them as training data to fine-tune a Hugging Face model (e.g., BERT-base-NER [47]). After fine-tuning the model, we utilize it to extract values from the summarized security controls and sub-control's discussion, and for attributes, we extract the security sub-controls. In Example 3, we explain the value and attribute extraction process.

4.3.3 Deriving Security Rules

After extracting security values and attributes from security controls, we generate our security rules. Our security rules are initially generated automatically by utilizing the variables along with

attributes and values, and then we express them in formal language. More specifically, we pull the variable names which are stored in the elaborated security standards, then we acquire the attributes and values from the previous step and put all these data into the format of our rule. The format of our derived security rule is as follows: $variable_1 = \{attribute_1 : \{values_1, values_2, ..., values_n\}, attribute_2 : \{values_1, values_2, ..., values_n\}, ...\}$. Below using an example, we show how we define our security rules with the obtained values and attributes.

Example 3. The summarization process for the `Expectation 21` and security control `SC-8(1)` is shown in Figure 4.4. The first two boxes contain `Expectation 21` and its related security sub-control, respectively, and the red highlighted texts indicate the summarized security sub-control. We extract the low-level security values , which are `TLS` and `IPSec` and attribute (e.g., `Cryptographic protection`) highlighted in red. The last box shows our low-level security rule: "*Data Protection* = {*Cryptographic protection:* {`TLS, IPSec`}}".

4.4 Instantiating to Actionable Rules

This section instantiates our derived security rules for IoT device-specific information and formalizes them into first-order logic for security applications. Specifically, *instantiation* is the process of making derived security rules specific to IoT devices so that a rule can be efficiently verified from the available IoT device data (e.g., logs, console output, etc.). However, it is insufficient to rely only on the automatically derived security rules because the security controls' values and attributes do not encompass all possible values in the context of IoT devices. To maintain accuracy, our system requires expert intervention after automatically extracting values from security controls. A specialist will eliminate undesirable or irrelevant values and determine whether any missing values should be added to our knowledge base, as illustrated at the beginning of Example 4. We then instantiate security rules to customize them for a particular IoT device, since security rules produced from IoT security standards are generally applicable to all sorts of IoT devices. We leverage our knowledge base from Section 4.2 to instantiate our derived security rules. After instantiating the security rules, we translate them into first-order logic because formal verification methods are more useful and effective than manual inspection for automated reasoning [30, 32]. Table 4.1 shows an excerpt of

our derived actionable rules.

Example 4. Device's sensitive data during transmission over Network: {`WiFi, BLE, LTE, NFC, PLC, RFID, Z-Wave, Zigbee`} should be cryptographically protected using Cryptographic mechanism: {`TLS, IPSec, AMQP, CoAP, DDS, MQTT`}. Suppose an example of an instantiated security rule for Nest Protect Smoke and CO Alarm Sensor is: "*Nest Protect Smoke and CO Alarm Sensor device's (device_id:* `peyiJNo0IldT2YlIVtYaGQ`*)* `smoke_alarm_state` *during transmission over Network:* `WiFi` *should be cryptographically protected using Cryptographic mechanism:* `TLS`*; should use protocol:* `TCP` *and port numbers:* {`443, 11095, 53, 67`}". Leveraging our proposed method, we formalize this rule as follows.

Rule 1:

$$\forall cp \in \{TLS, IPSec, AMQP, CoAP, DDS, MQTT\},$$
$$\forall s \in SensitiveData, \forall n \in \{WiFi, BLE, LTE, NFC,$$
$$PLC, RFID, Z-Wave, Zigbee\}, \forall d\_id \in DeviceID$$
$$CryptographicProtection(d\_id, s, cp) \wedge$$
$$transmission(s, n) \implies DataProtected(d\_id, s)$$

4.5 Rationale behind Our Semi-Automated Approach

Table 4.2 shows the objective of different steps of our approach, as well as our explanation as to why each of the steps is either manual, semi-automated, or fully automated. Specifically, the first column lists all the steps of our approach, second column indicates how those steps are performed (i.e., automatic, semi-automatic, or manual), third column describes each step's objective, and fourth column states the rationale behind using the stated approach of those steps.

Table 4.1: An excerpt of derived and instantiated security rules using our approach

Control	Summary	Derived Rule	Instantiated Rule
SC-8(1)	Cryptographic mechanisms that protect the confidentiality of information during transmission include `TLS` and `IPSec`.	Data Protection1 = Cryptographic protection: {`TLS, IPSec`}	Device's sensitive data during transmission over Network: {`WiFi, BLE, LTE, NFC, PLC, RFID, Z-Wave, Zigbee`} should be cryptographically protected using Cryptographic mechanism: {`TLS, IPSec, AMQP, CoAP, DDS, MQTT`}
SC-8(3)	Message externals include message headers and routing information should be cryptographically protected.	Data Protection2 = Cryptographic protection for message externals: {`headers information, routing information`}	Device's network packet's Message headers and routing information: {`Version, Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, Source Address, Destination Address`} should be protected using Cryptographic mechanism: {`TLS, IPSec`}.
SC-8(4)	Communication patterns (e.g., frequency, periods, predictability amount) should be concealed or randomized by encrypting the links and transmitting in continuous, fixed, or random patterns.	Data Protection3 = Randomized communication pattern: {`frequency, periods, predictability, amount`}	Device's Communication patterns: {`frequency, periods, predictability, amount`} should be randomized or concealed by Cryptographic mechanism: {`TLS, IPSec`}.
SC-23(1)	Invalidate session identifiers upon user logout or other session termination.	Data Protection4 = Invalidating session identifiers at logout: `Enabled`	Device's Session identifiers: {`"CD723LGeXlf-01:34"`} should be invalidated upon user_state: {`logout or session termination`}.
AC-18(3)	Wireless networking should be disabled when not used.	Data Protection5 = Disable wireless networking: `Enabled`	Device's Network: {`WiFi, BLE, LTE, NFC, PLC, RFID, ZWave, Zigbee`} should be disabled when not used.

Table 4.2: Different steps of our approach, their objectives, and rationales

Step	**Approach**	**Objective**	**Rationale**
Step 1.1.1 Extracting Security Controls	Automatic	To centralize all IoT security standards and their corresponding security controls in a single document so that it can be efficiently used by our toolchain later.	Since it extracts security controls based on the mappings and it is error-free, extracting security controls from NIST SP 800-53r5 does not necessitate any expert review.
Step 1.1.2 Annotating Data	Manual	To fine-tune the NER model so that it can extract security values from security controls.	As there are no trained NER models available to extract security values, we had to annotate the security values manually to fine-tune the NER model.
Step 1.2.1 Building Structural Knowledge	Semi-automatic	To instantiate derived security rules that are specific to IoT devices by using the structural knowledge of IoT devices.	We leverage the approach of Dolan et al. in [8] to automatically extract device specifications, configs, and API documentation, while all additional device-specific data is manually verified to assure its completeness.
Step 1.2.2 Building Functional Knowledge	Automatic	To instantiate derived security rules using the functional knowledge such as the network behavior of IoT devices.	We utilize MUDGEE [49], which automatically delivers IoT network port and protocol number without manual inspection.

Continued on next page

Table 4.2 – *Continued from previous page*

Step	Approach	Objective	Rationale
Step 2.1.1 Summarizing Security Controls	Semi-automatic	To extract the most crucial information from lengthy security controls, which are otherwise tedious and time-consuming tasks.	After automatically generating the summary using the BERT-Extractive-Summarizer, we need expert assessment to ensure the semantics and validity of the summaries.
Step 2.1.2 Extracting Values	Semi-automatic	To automatically extract security values from security controls which can be used in the security rules.	We have a fine-tuned NER model to extract security values from security controls, but expert inspection is vital to ensure that the model does not exclude any required values or extract extraneous information.
Step 2.1.3 Deriving Security Rules	Semi-automatic	To help security experts to utilize the actionable security rules in different security applications.	We derive actionable security rules after extracting values and using our knowledge base; however, expert evaluation is crucial to preserve the accuracy of the generated rules because security controls do not include all of the potential security values or attributes in the context of IoT.

Continued on next page

Table 4.2 – *Continued from previous page*

Step	**Approach**	**Objective**	**Rationale**
Step 2.2.1 Instantiating Security Rules	Manual	To make the derived security rules specific to IoT devices so that a rule can be efficiently verified from the available IoT device data.	Security rules are manually instantiated with IoT device-specific information stored in our knowledge since there is no mapping between security rules and IoT device-specific data, ensuring that only a particular device-specific information is included in the instantiated rule.
Step 2.2.2 Interpreting in Formal Language	Manual	To enable formal verification tools to carry out security verification.	Instantiated security rules are converted manually to formal language as there are no readily available tools to convert the natural language to mathematical expressions.

Chapter 5

Applications

This chapter shows how our actionable rules can be applied to different security mechanisms (e.g., security auditing, Intrusion Detection Systems (IDS), and secure application development).

5.1 Application to IoT Security Auditing

Below we demonstrate the different steps of IoT security auditing utilizing our actionable security rules.

5.1.1 Identifying, Collecting, and Processing Rule-Specific Audit Data

To validate security compliance for each security rule, it is essential to determine the relevant IoT data sources, collect them, and prepare them for the specific audit tools (e.g., formal methods). Logs, configuration files, and databases are the primary sources of audit data in IoT devices, and IoT hub or IoT cloud server stores these data. Different data types and sources, such as device-related data, connectivity-related data, user-related data, and application-related data, are identified based on the security rules [53]. After identifying relevant data sources, we gather data and process them in a structured manner so that they can be converted into formal language. It is crucial to transform the data into a consistent format because different data sources store the data in different formats. Finally, audit data and security rules are converted to formal language for verification. In this work, we particularly use constraint satisfaction problem (CSP), which is also used in other

auditing solutions (e.g., [30–32]). To that end, each data group is represented as tuples, and the code is append with the relationships for security rules (discussed in Section 4.4). Listing 1 shows the tuples in our Sugar [54] code.

5.1.2 Conducting Formal Verification

For verification, we utilize formal verification techniques, e.g., Boolean satisfiability problem (SAT) solver. Specifically, we leverage an SAT-based tool, namely, Sugar [54], to perform the verification process and interpret the verification results. Afterward, Sugar verifies all the constraints, and then we can interpret if any security rule is breached. Lastly, an audit report is generated after getting results from a formal verification tool. The Sugar tool evaluates "true" or "false" based on the result of a security rule breach. A security expert can investigate further to determine the root cause of a breach only after discovering it in an auditing process, eliminating the need for them to manually go through all the irrelevant information of IoT devices for security breaches.

```
1  // Declaration
2  (domain DeviceID 0 5000) (domain CryptoMech 1 6)
3  (domain NetType 11 20)(domain SensitiveData 21 40)
4  (int D DeviceID) (int CR CryptoMech)
5  (int N NetType) (int S SensitiveData) |\DNumber|
6
7  // Relations Declarations and Audit Data as their Support
8  (relation CryptProtection 3 (supports ((2471 13 4) (2798 29 2) (861 9 4) ))
9  (relation Transmission 2 (supports ((12 9) (29 10) (9 1) )) |\DNumber|
10
11 // Security property: DataProtectionTransmission
12 (predicate (DataProtectionTransmission D S CR N) (and (CryptProtection D S
       CR)
13 (Transmission S N) (not (DataProtection D S)) ))
14 (DataProtectionTransmission D S CR N)
```

Listing 5.1: Sugar source code for verifying Rule 1

Example 5. The CSP code to audit the data protection using the rule presented in Listing 1.1. Each domain and variable is first declared (Lines 2-5). Then, the set of involved relations, namely, *CryptProtection* and *Transmission*, are defined and populated with their supporting tuples (Lines 7-8), where the support is generated from simulated data by utilizing the Amazon IoT simulator [55]. Then, the data protection at transmission is declared as a predicate, denoted by *DataProtection-Transmission*, over these relations (Lines 10-11). Finally, the predicate should be instantiated (Line 19) to be able to be verified. The `UNSAT` result on Sugar means that all constraints are not satisfied, and hence, there is no violation of the rule. Note that the predicate will be unfolded internally by Sugar for all possible values of the variables, which allows to verify each instance of the problem among possible values of `device ID, cryptographic mechanism,` and `network types`. We evaluate this auditing step in Section 7.

5.2 Other Applications

This section describes other applications to our solution.

5.2.1 Snort IDS

Snort [56], a potent open-source intrusion detection system (IDS) and intrusion prevention system (IPS), finds potentially malicious activities by employing a rule-based language that integrates anomaly, protocol, and signature inspection techniques. Our low-level security rules obtained from security standards can be easily translated into snort rules. To convert our low-level security rules into Snort rules, first, we need to know the format of Snort rules and the required data for the rules. Snort IDS/IPS rules consist of two parts, *rule header* and *rule option*. The *rule header* contains the following fields: `action, protocol, source address, source port, direction, a destination address, and destination port`. The *rule option* of Snort is divided into a keyword and an argument, defined inside parentheses and separated by a semicolon. In this work, we obtain protocol and port numbers from our low-level security rules. In the same manner, our security rules can be utilized by other IDS systems, such as Suricata [57], Zeek [58], OSSEC [59], etc.

Example 6. Below is a low-level rule instantiated for Nest Protect Smoke and CO Alarm device, which ensures encrypted data transmission, and then we convert it into a Snort rule. Our derived security rule is: "*Nest Protect Smoke and CO Alarm Sensor device's (device_id:* `peyiJNo0IldT2YlIVtYaGQ`*)* `smoke_alarm_state` *during transmission over Network:* `WiFi` *should be cryptographically protected using Cryptographic mechanism:* `TLS`*; should use protocol:* `TCP` *and port numbers:* {`443, 11095, 53, 67`}". The corresponding Snort rule is: "`alert tcp any any <> $HOME_NET ![443, 11095, 53, 67] (msg: \Unencrypted Traffic"; sid:1000005)`". If this snort rule matches the network traffic data - which actually means the fields of TCP packet (source address, source port, destination address, and destination port) match with the rule *(*`any, any, IP address of $HOME_NET`*, port numbers other than* `443, 11095, 53, or 67`*)*, respectively, then an alert is generated that outputs the message "*Unencrypted Traffic*" with the signature `ID 1000005`.

5.2.2 Secure Application Development

As most IoT application developers are not security experts, they might need concrete guidelines and recommendations to develop secure applications and interfaces following existing security standards and best practices. Our security rules provide IoT manufacturers and developers with actionable guidelines which can be followed to implement them in actual IoT systems, as demonstrated through the following example.

Example 7. We utilize the used port numbers (`443, 11095, 53, and 67`) from IoT device-specific data (in Example 2) to communicate with servers, while the high-level security standard is ambiguous about which port to use. A code snippet is presented in Listing 1.2 and shows the port numbers (Line 4) used by SmartThings SmartApp [60] to listen to the server (Line 6).

```
const SmartApp = require('@smartthings/smartapp');
const express = require('express');
const server = express();
const PORT = [443, 11095, 53, 67]; |\DNumber|

/* Start listening at your defined PORT */
server.listen(PORT, () => console.log(`Server is up and running
on port ${PORT}`));
```

Listing 5.2: Port numbers derived from our security rules used by SmartThings SmartApp

Similar to these applications, our actionable security rules might further be applied to other security mechanisms, such as access control, monitoring, risk assessment, etc. to cover various security aspects in IoT.

Chapter 6

Implementation

We describe the implementation of the automated steps of our approach as follows. To build knowledge of security standards, we develop a Python script that extracts the security expectations from NIST IR 8228 and the referenced NIST SP 800-53r5 security controls from control catalog (provided by NIST), and store them in a CSV file with attributes such as variables, expectations, controls, sub-controls and their discussions. To build device-specific knowledge, network data such as port numbers and protocols are extracted by leveraging MUDGEE [49], which creates MUD [48] profiles of IoT devices by monitoring network traces and technical specifications of different IoT devices are extracted by leveraging the approach from [8]. For the summarization of security controls, we use BERT-Extractive-Summarizer [51]. Then, annotation of security controls is performed using NER Annotator [61]. To extract values from security controls, we fine-tune a Hugging Face transformer-based named entity recognition model called Bert-base-uncased [47]. Lastly, for verification, we use the Boolean satisfaction (SAT) solver tool, namely, Sugar V2.2.1 [54].

In the following, we provide the rest of the four (04) derived security rules and their sugar code used for the verification of the data collected from the NEST Protect Smoke & CO device.

Rule 2:

$$\forall cr \in TLS, IPSec, \forall hr \in Header\ length, Total\ length, Identification,$$
$$Flags, Fragmented\ offset, Time\ to\ live, Protocol, Header\ checksum,$$
$$Source\ IP\ address, Destination\ IP\ address, Options,\ \forall d_i d \in Device\ ID,$$
$$\forall s \in Sensitive\ Data$$
$$CryptographicProtection(d_i d, hr, cr) \implies Data\ Protected(d_i d, s)$$

```
// Declaration
(domain DeviceID 0 5000)
(domain Header 0 12)
(domain CryptMech 1 2)
(domain SensitiveData 21 40)

( int D DeviceID)
( int H Header)
( int C CryptMech)
( int S SensitiveData)

;// Relations Declarations and Audit Data as their Support
( relation CryptProtection 3 ( supports ((3760 0 8) (559 5 10) (4577 11 36)
    ));

// Security property: Cryptographic Protection for Message Externals
( predicate (CryptForMsgExternals D H C S)
(and (CryptProtection D H C)
(not (DataProtection D S)) ))

(CryptForMsgExternals D H C S)
```

Listing 6.1: Sugar source code for verifying Rule 2

Rule 3:

$$\forall cp \in TLS, IPSec, \forall com \in \{frequency, periods, predictability, amount\},$$

$$\forall d_id \in DeviceID, \forall s \in SensitiveData$$

$$CommunicationPatternEncrypted(cp, com, d_id) \implies DataProtected(d_id, s)$$

```
Declaration
(domain DeviceID 0 5000)
(domain Common 0 12)
(domain CryptMech 1 2)
(domain SensitiveData 21 40)

( int D DeviceID)
( int Cm Common)
( int C CryptMech)
( int S SensitiveData)

;// Relations Declarations and Audit Data as their Support
( relation ComPatEnc 3 ( supports ((2493 2 16) (57 7 40) (4865 7 8) ));


// Security property: Conceal or Randomize Communications
( predicate (ConRanCom D Cm C S)
(and (ComPatEnc  D Cm C)
(not (DataProtection D S)) ))

(ConRanCom D Cm C S)
```

Listing 6.2: Sugar source code for verifying Rule 3

Rule 4:

$$\forall sid \in Session\ identifier,\ \forall id \in Device\ ID,$$

$$\forall u \in User,\ \forall l \in Logout,\ \forall s \in Sensitive\ Data:$$

$$InvalidateSessionIdentifiersUponLogout(d_id,\ sid,\ u,\ l)$$

$$\implies\ Data\ Protected(d_id,\ s)$$

```
1  //Declaration
2  (domain DeviceID 0 5000)
3  (domain Session 0 30000)
4  (domain User 0 100000)
5  (domain SensitiveData 21 40)
6  (domain LogVal 0 1)
7
8  ( int D DeviceID)
9  ( int Ses Session)
10 ( int U User)
11 ( int S SensitiveData)
12 ( int l LogVal)
13
14 ;// Relations Declarations and Audit Data as their Support
15 ( relation invalidateSessionLogout 4 ( supports ((4603 13 1 1) (1609 1 2 0)
       (4849 4 1 0) ))
16
17 ;// Security property: INVALIDATE SESSION IDENTIFIERS AT LOGOUT
18 ( predicate (invalidateSessionAtLogout D Ses U S l)
19 (and (invalidateSessionLogout   D Ses U l)
20 (not (DataProtection D S)) ))
21
22 (invalidateSessionAtLogout D Ses U S l)
```

Listing 6.3: Sugar source code for verifying Rule 4

Rule 5:

$$\forall n \in TLS, IPSec, AMQP, CoAP, DDS, MQTT,$$

$$\forall d_i d \in DeviceID, \forall s \in SensitiveData$$

$$DisableWirelessNetworking(n, d_i d) \implies DataProtected(d_i d, s)$$

```
// Declaration
(domain DeviceID 0 5000)
(domain NetType 11 14)
(domain SensitiveData 21 25)

( int D DeviceID)
( int N NetType)
( int S SensitiveData)

;// Relations Declarations and Audit Data as their Support
( relation DisableNet 2 ( supports ((22 14) (15 14) (11 10) ));

// Security property: DisableNetworking
( predicate (DisableNetworking D S N)
(and (DisableNet  D S CR)
(not (DataProtection D S)) ))

(DisableNetworking D S N)
```

Listing 6.4: Sugar source code for verifying Rule 5

Chapter 7

Experiments

This chapter describes the details of our experiments and results.

7.1 Experimental Settings

We run our experiment on a workstation with an Intel(R) Core(TM) i7-10700 2.90GHz CPU and 16 GB of physical memory. To generate our dataset, we utilize NIST IR 8228 [1] and NIST SP 800-53r5 [40] and Amazon IoT device simulator platform [55] with 5,000 IoT devices and their logs, configuration files, and network data. Figure 7.1 illustrates the count of technical and non-technical security controls for each expectation of NIST IR 8228. We convert them into the input format, Constraint Satisfaction Problem (CSP), of Sugar [54]. We iterate each experiment for 200 times and average measurements.

Table 7.1: Performance evaluation of value extraction

	Precision	**Recall**	**F1-Score**
Values	0.82	0.98	0.89
Attributes	0.97	0.94	0.95
Average	0.87	0.97	0.91

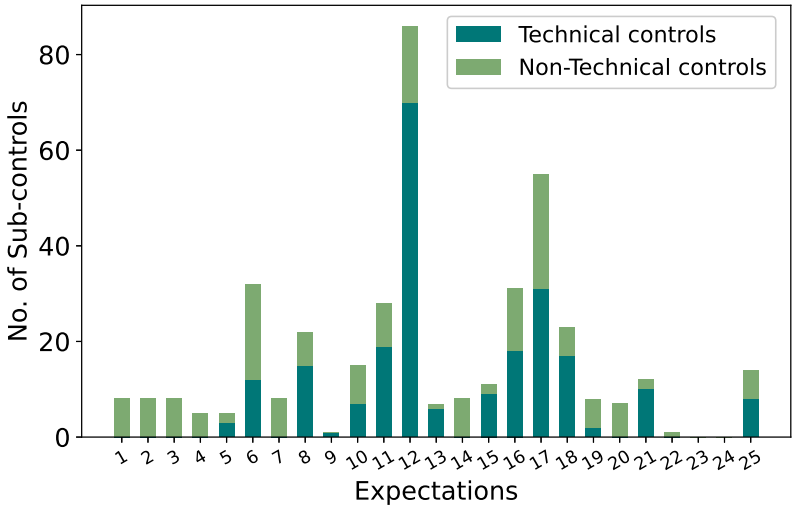

Figure 7.1: Count of security controls for each expectation from NIST IR 8228 [1]

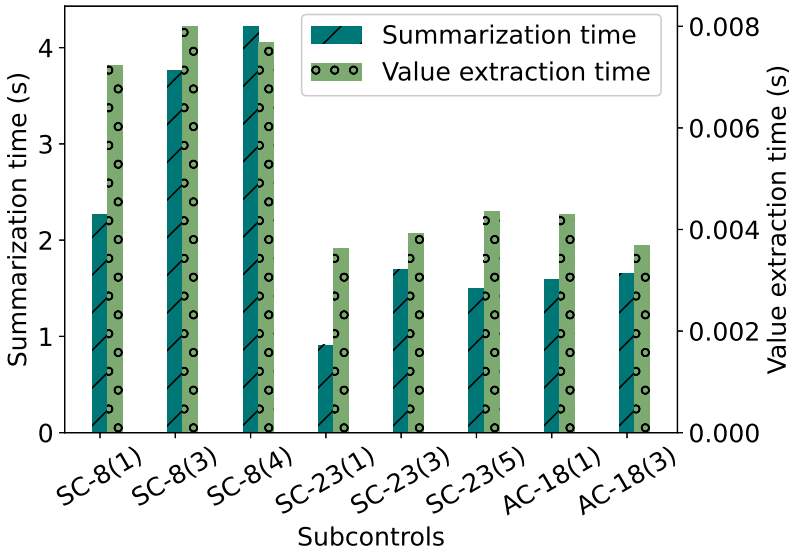

Figure 7.2: Time required for summarizing and value extraction by our approach

7.2 Evaluation of Summarization and Value Extraction

In the first set of experiments, we measure the time required for summarizing each security control's discussion and value extraction step as well as the accuracy of our value extraction using precision, recall, and F1-score. Figure 7.2 shows that the time required for summarizing varies from less than one second to just over four seconds, because some security sub-controls are rather lengthy over the others, summarising them requires more time. However, since the summarization procedure

is performed only once, overheads are tolerable for auditing such big settings. This figure also demonstrates that extracting values from the discussion of summarized security sub-controls takes only a fraction of a second, which is very time efficient compared to the summarization process. As shown in Table 7.1, the precision scores for values and attributes are 82% and 97%, respectively. For recall, scores of values and attributes are 98% and 94%, respectively. Values and attributes have an F1-score of 89% and 95%, respectively.

7.3 Evaluation of Derived Security Rules

Our second set of experiments is to evaluate the effectiveness of our derived security rules by examining their execution time, memory usage, and CPU usage, along with the measurement of the reduction in manual effort.

Our approach aims at reducing the manual effort required by an expert for deriving actionable security rules. Figures 7.3 and 7.4 demonstrate the amount of reduction in manual effort for summarizing and deriving actionable security rules, where we compare a fully manual approach with ours for this measurement using four similarity metrics (e.g., Cosine similarity [62], Jaro-Winkler similarity [63], Sorensen similarity [64], and Jaccard similarity metrics [65]). Based on the similarity between summarized sub-controls and derived security rules, we measure the reduction in manual effort by security experts. In other words, a security expert needs to exert less work when the summaries and derived security rules are more similar or closely resemble manually summarized and derived security rules. Figure 7.3 shows how our summarization tool reduced the amount of work required to summarise eight security sub-controls, with Cosine and Jaro-Winkler similarity scores averaging the highest percentages of 57% and 65%, respectively, among these four. The Sorensen similarity score is then anywhere between 50% and 37%, with Jaccard's score being the lowest. Next, Figure 7.4 shows the effort reduced in deriving a security rule, with Cosine and Jaro-Winkler similarity scores again averaging the highest percentages of 50% and 52%, respectively, among these four. The Sorensen similarity score is then anywhere between 45% and 30%, with Jaccard's score being the lowest again. Overall it reduces around 50% of manual effort and for

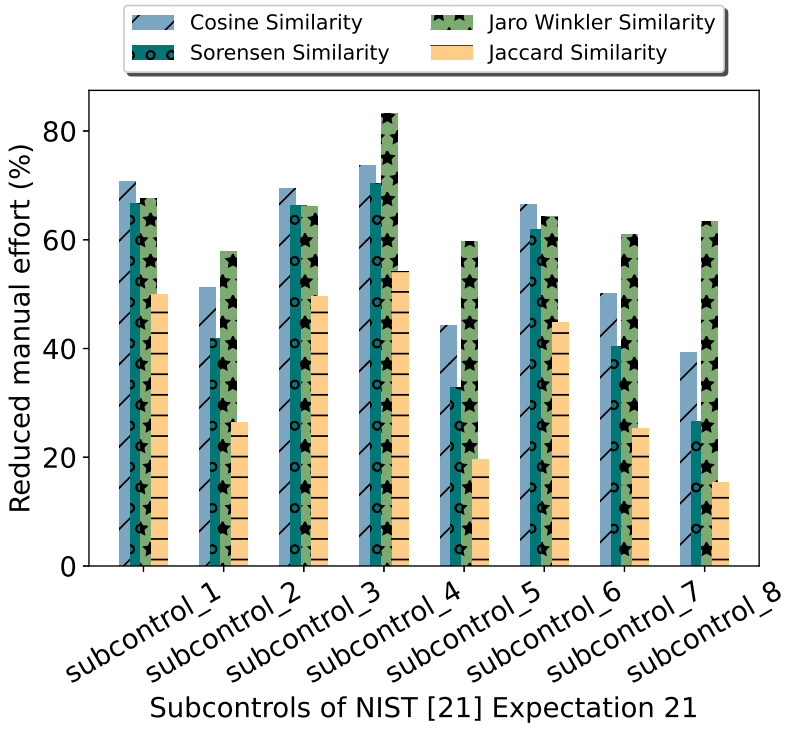

Figure 7.3: Similarity score of summarized sub-control

security experts, this represents a significant decrease in manual work and time-consuming activities. The main purpose of this set of experiments is to show the resemblance of our derived rules with manually summarized sub-controls. For this purpose, we use popular similarity metrics and compare their results. We assume that those scores (i.e., calculating the resemblance between two outputs) might give a hint on the reduced manual effort that these derived rules can bring. However, we acknowledge that a user survey will be needed to more accurately evaluate the usability of our approach (as further discussed in Chapter 8).

We then evaluate the efficiency of our derived security rules in terms of time, CPU, and memory utilization. In Figure 7.5, we observe that overall it takes less than ten seconds for 5,000 IoT devices to validate each of the five rules derived from Expectation 21. As the number of devices grows, the required time to validate each rule also increases, but after 1,500 devices, a significant reduction in increase is observed, which again increases after 4,000 devices. Given the number of devices, ten seconds is a very realistic amount of time to perform auditing. Figure 7.6 shows the CPU usage by varying the number of devices. With a range of between 20% and 25%, CPU utilization increases

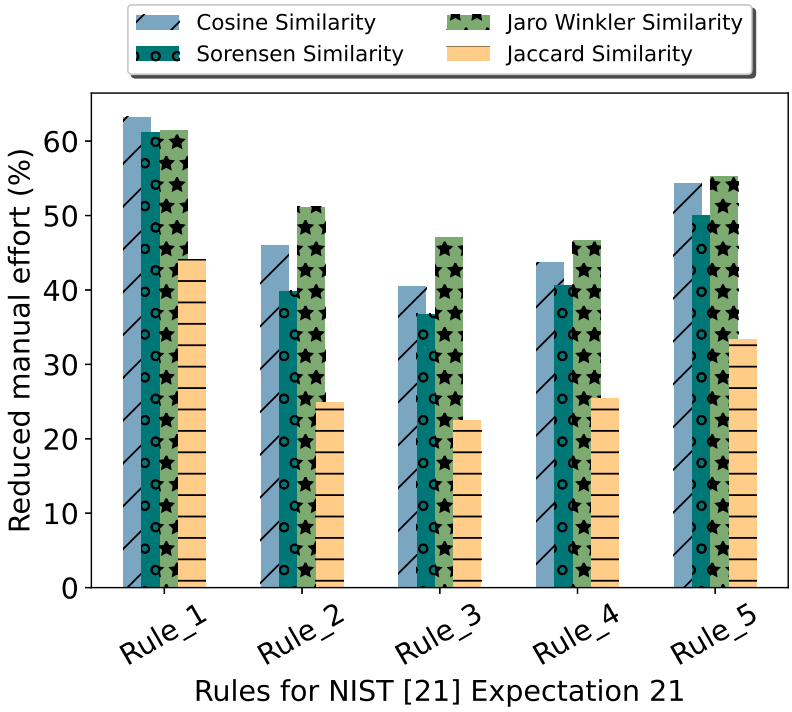

Figure 7.4: Similarity score of derived rules

almost linearly for all five security rules. Since there are 5,000 IoT devices, and each one generates a unique set of data, the CPU usage for auditing is reasonable. Note that we only utilize a single PC for our experiments; the cost would be significantly lower if we could run Sugar for verification on multiple VMs. Our final experiment (Figure 7.7) measures the memory usage of our auditing solution. All of the five rules show a similar trend as CPU consumption. At its peak, it requires around 43 MB of memory, and overall, it requires less than 41 MB. It is noteworthy that Rule 1 uses more resources because there are more tuples and, therefore, more data to validate.

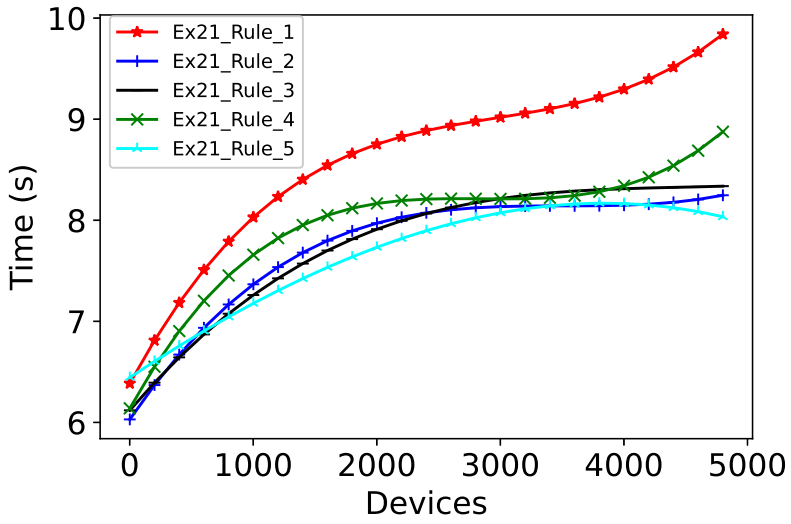

Figure 7.5: Time requirement

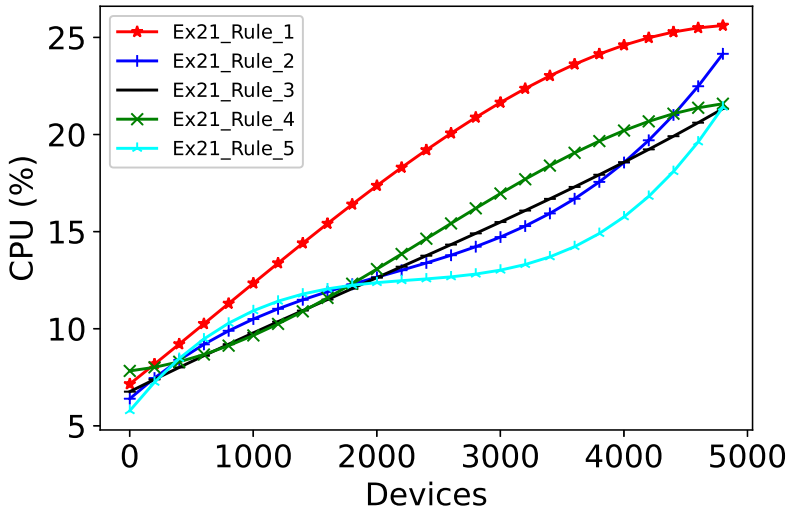

Figure 7.6: CPU utilization

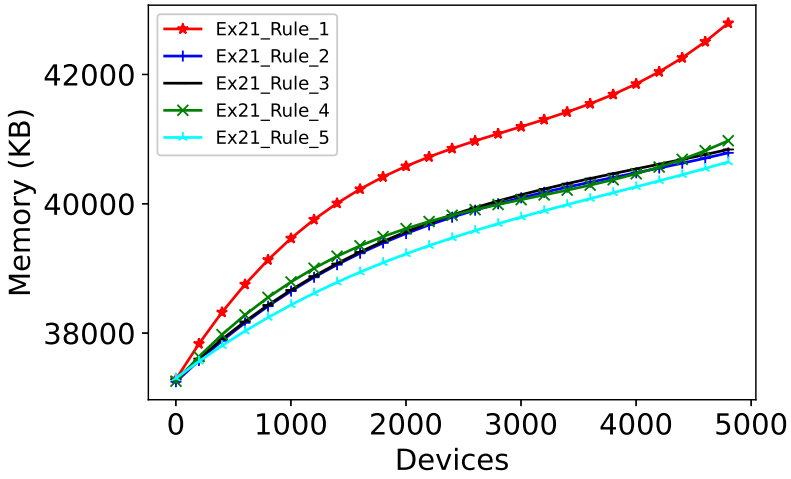

Figure 7.7: Memory utilization

Chapter 8

Discussion

This chapter discusses different aspects of our work.

8.1 Guidelines for the Required Manual Effort

Our approach requires the involvement of security specialists in order to function to its maximum potential. An individual with in-depth knowledge and experience in protecting information systems is referred to as a security specialist or expert. Firstly, a security specialist will review the automatically generated summaries of security controls to ensure they are complete and not missing any crucial information. Secondly, security experts will verify the accuracy of the retrieved values from the summarized security controls. Following these actions, low-level security rules will be created using the retrieved values and any additional potential values relevant to the IoT. Lastly, a security expert must carefully consider each possible value of a security rule that will be applied during security auditing. The formalization of the low-level security rule into first-order logic will result in CSP code for the Sugar tool. Our derived actionable security rules can be converted into any formal language based on the requirement of the security tools and can be used for various security purposes.

8.2 Covering Other Security Standards

In this thesis, we consider the IoT security standard from NIST IR 8228 and utilize its mapping to NIST SP 800-53r5 to derive actionable security rules. However, there are many other security standards from different federal and non-federal organizations available for IoT systems and cloud platforms, which can be easily incorporated with our methodology by getting their mapping to NIST SP 800-53r5. To that end, European Union Agency for Cybersecurity (ENISA) Baseline Security Recommendations for IoT in the Context of Critical Information Infrastructures [21] provides a mapping with NIST SP 800-53r5 in their security standard. Additionally, OWASP is working on a project to provide a mapping of the OWASP IoT Top 10 2018 to various industry policies and publications [66]. Once available, those mappings can be utilized to cover other security standards using our approach.

8.3 Validating the Usability of Our Solution

To further validate the usability of our approach, we plan to carry out a user survey in the future. This study might provide feedback on the effectiveness (e.g., possible increasing efforts due to any incorrectness or mistake in our derived rules) and usability of our tool. The results and feedback of this study can be considered in the following version of our proposed approach. Specifically, to analyze the usability of our derived security rules, we will develop multiple scenarios where participants can experience our tool in contrast to a fully manual approach as well as a semi-guided approach to derive rules followed by a questionnaire with a variety of closed-ended (e.g., multiple choice and Likert scale) and open-ended (e.g., strength, weakness, and suggestions) questions. Our target group for this survey will be security researchers and industry practitioners (leveraging our existing collaborations) as potential users of such tools.

8.4 Feedback to Standardization Authorities

As our solution aims at mapping high-level security standard specifications to low-level system implementations, it might be able to identify existing issues (e.g., missing concrete or related information to realize a security recommendation) in a standard specification. Additionally, interpreting the final and intermediate outcomes of our solution might provide insights into further clarifying the recommendations in current security standards.

We intend to provide standardization bodies feedback that will help them build future standards in a clearer and more useful manner. In doing so, we hope to reduce the underutilization of security standards while also ensuring that emergent technologies and assaults are addressed effectively. Overall, we believe that our method can increase the flexibility and efficacy of security standards in solving security threats.

Chapter 9

Conclusion

The chapter proposed a novel method for deriving actionable security rules for different security applications from high-level security standards. This was accomplished by gathering device-specific data, instantiation of derived security rules, and verifying the results using formal techniques. The experiment's findings showed how well the derived rules worked for security auditing and suggested other security-related domains where they might be used. To achieve this, we collected device-specific data to instantiate the derived security rules. Then formal verification techniques were used to verify the rules' effectiveness in identifying potential security vulnerabilities. The findings demonstrated the viability of this approach for use in security audits by demonstrating the effectiveness of the derived rules in identifying security issues. We also pointed out that our approach is applicable to security applications other than security audits, such as intrusion detection systems and secure application developments.

Overall, the proposed approach presents a promising solution to the challenges associated with IoT security. By deriving actionable security rules from high-level security standards and using formal tools to verify their effectiveness, we have demonstrated the potential of this approach in security auditing as well as application in intrusion detection systems and secure development. By offering an automated and scalable solution for IoT device security, this strategy may eventually have larger applications in the realm of cybersecurity.

There exist a few limitations to this thesis work.

- Device-specific data collection is currently done manually, which restricts its capacity to

scale. We intend to automate this process in our future work to lessen the effort on security experts. The methodology's applicability will be increased beyond its current scope with the addition of additional security standards in future work.

- In order to reduce the amount of effort needed by security experts, we aim to automate the majority of the steps involved in the generation of actionable security rules.

- In order to further improve our approach, we intend to conduct a user study to evaluate its usability and efficacy. The study will involve real-world security practitioners who have experience with security standards and best practices. These practitioners will be asked to use the approach and provide feedback on their experience, potentially identifying areas for improvement. We can better grasp the practical consequences and uses of our approach by including real-world security practitioners in the evaluation process. The feedback from the study can be used to refine and enhance the approach, making it more user-friendly and effective in addressing real-world security concerns.

We believe that our approach has the potential to enhance the implementation and adoption of security standards in emerging technologies, and hope to see it contribute to the wider security community in the future.

Bibliography

[1] Katie Boeckl, Katie Boeckl, Michael Fagan, William Fisher, Naomi Lefkovitz, Katerina N Megas, Ellen Nadeau, Danna Gabel O'Rourke, Ben Piccarreta, and Karen Scarfone. *Considerations for managing Internet of Things (IoT) cybersecurity and privacy risks*. US Department of Commerce, National Institute of Standards and Technology, 2019.

[2] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. SoK: Security evaluation of home-based IoT deployments. In *IEEE SP*. IEEE, 2019.

[3] Eyal Ronen and Adi Shamir. Extended functionality attacks on iot devices: The case of smart lights. In *IEEE EuroS&P*. IEEE, 2016.

[4] Sukhvir Notra, Muhammad Siddiqi, Hassan Habibi Gharakheili, Vijay Sivaraman, and Roksana Boreli. An experimental study of security and privacy risks with emerging household appliances. In *IEEE CNS*, 2014.

[5] Vijay Sivaraman, Dominic Chan, Dylan Earl, and Roksana Boreli. Smart-phones attacking smart-homes. In *ACM WiSec*, 2016.

[6] Pierre-Antoine Vervier and Yun Shen. Before toasters rise up: A view into the emerging iot threat landscape. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 556–576. Springer, 2018.

[7] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. Smart locks: Lessons for securing commodity internet of things devices. In *ACM ASIACCS*, pages 461–472, 2016.

[8] Andy Dolan, Indrakshi Ray, and Suryadipta Majumdar. Proactively extracting iot device capabilities: An application to smart homes. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 42–63. Springer, 2020.

[9] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the Mirai botnet. In *USENIX Security*, 2017.

[10] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light commands: Laser-Based audio injection attacks on Voice-Controllable systems. In *USENIX Security*, 2020.

[11] Wenbo Ding, Hongxin Hu, and Long Cheng. IOTSAFE: Enforcing safety and security policy with real IoT physical interaction discovery. In *NDSS*, 2021.

[12] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. HoMonit: Monitoring smart home apps from encrypted traffic. In *ACM CCS*, 2018.

[13] Z Berkay Celik, Gang Tan, and Patrick D McDaniel. IoTGuard: Dynamic enforcement of security and safety policy in commodity IoT. In *NDSS*, 2019.

[14] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Lannan Luo. PFIREWALL: Semantics-aware customizable data flow control for smart home privacy protection. *arXiv preprint arXiv:2101.10522*, 2021.

[15] Yinxin Wan, Kuai Xu, Guoliang Xue, and Feng Wang. IoTArgos: A multi-layer security monitoring system for internet-of-things in smart homes. In *IEEE INFOCOM*. IEEE, 2020.

[16] Eirini Anthi, Lowri Williams, Małgorzata Słowińska, George Theodorakopoulos, and Pete Burnap. A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, 6(5):9042–9053, 2019.

[17] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Zhuoqing Morley Mao, Atul Prakash, and SJ Unviersity. ContexIoT: Towards providing contextual integrity to appified IoT platforms. In *NDSS*, 2017.

[18] Z Berkay Celik, Patrick McDaniel, and Gang Tan. Soteria: Automated IoT safety and security analysis. In *USENIX ATC*, pages 147–158, 2018.

[19] Pantaleone Nespoli, Daniel Díaz-López, and Félix Gómez Mármol. Cyberprotection in IoT environments: A dynamic rule-based solution to defend smart devices. *Journal of Information Security and Applications*, 60:102878, 2021.

[20] Menachem Domb, Elisheva Bonchek-Dokow, and Guy Leshem. Lightweight adaptive random-forest for IoT rule generation and execution. *Journal of Information Security and Applications*, 34:218–224, 2017.

[21] EU ENISA. Baseline security recommendations for IoT in the context of critical information infrastructures. *European Union Agency for Cybersecurity Heraklion, Greece*, 2017.

[22] OWASP top 10 Internet of Things 2018. 2018. URL https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf.

[23] Nest protect and CO alarm, 2021. URL https://store.google.com/product/nest_protect_2nd_gen_specs?hl=en-US.

[24] Christopher Bellman and Paul C van Oorschot. Systematic analysis and comparison of security advice as datasets. *Computers & Security*, 124:102989, 2023.

[25] John Verry. Should I use NIST 8228 or NIST 8259 for IoT design or IoT testing?, Jun 2020. URL https://www.pivotpointsecurity.com/should-i-use-nist-8228-or-nist-8259-for-iot-design-or-iot-testing/.

[26] Michael Fagan, Katerina Megas, Karen Scarfone, and Matthew Smith. Recommendations for IoT device manufacturers: Foundational activities and core device cybersecurity capability baseline (2nd draft). Technical report, National Institute of Standards and Technology, 2020.

[27] ETSI. En 303 645 - v2.1.1 - cyber; cyber security for consumer internet of things: Baseline requirements. 2020. URL https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf.

[28] The UK government. code of practice for consumer IoT security. 2019. URL https://www.gov.uk/government/publications/code-of-practice-for-consumer-iot-security.

[29] Carol J Fung and Bill McCormick. An effective policy sharing mechanism for smart home networks. In *IEEE CNSM*. IEEE, 2020.

[30] Suryadipta Majumdar, Taous Madi, Yushun Wang, Yosr Jarraya, Makan Pourzandi, Lingyu Wang, and Mourad Debbabi. Security compliance auditing of identity and access management in the cloud: Application to OpenStack. In *IEEE CloudCom*. IEEE, 2015.

[31] Suryadipta Majumdar, Taous Madi, Yushun Wang, Yosr Jarraya, Makan Pourzandi, Lingyu Wang, and Mourad Debbabi. User-level runtime security auditing for the cloud. *IEEE Transactions on Information Forensics and Security*, 13(5):1185–1199, 2017.

[32] Taous Madi, Suryadipta Majumdar, Yushun Wang, Yosr Jarraya, Makan Pourzandi, and Lingyu Wang. Auditing security compliance of the virtualized infrastructure in the cloud: Application to OpenStack. In *ACM CODASPY*, 2016.

[33] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, Xianzheng Guo, and Patrick Tague. Smartauth: User-centered authorization for the internet of things. In *USENIX Security*, 2017.

[34] Ping Lou, Guantong Lu, Xuemei Jiang, Zheng Xiao, Jiwei Hu, and Junwei Yan. Cyber intrusion detection through association rule mining on multi-source logs. *Applied Intelligence*, 51 (6):4043–4057, 2021.

[35] Seiichi Ozawa, Tao Ban, Naoki Hashimoto, Junji Nakazato, and Jumpei Shimamura. A study of IoT malware activities using association rule learning for darknet sensor data. *International Journal of Information Security*, 19(1):83–92, 2020.

[36] Martin Husák, Tomáš Bajtoš, Jaroslav Kašpar, Elias Bou-Harb, and Pavel Čeleda. Predictive cyber situational awareness and personalized blacklisting: a sequential rule mining approach. *ACM Transactions on Management Information Systems (TMIS)*, 11(4):1–16, 2020.

[37] Fatemeh Safara, Alireza Souri, and Masoud Serrizadeh. Improved intrusion detection method for communication networks using association rule mining and artificial neural networks. *IET Communications*, 14(7):1192–1197, 2020.

[38] Zhongyuan Xu and Scott D Stoller. Mining attribute-based access control policies. *IEEE Transactions on Dependable and Secure Computing*, 12(5):533–545, 2014.

[39] Matthew W Sanders and Chuan Yue. Mining least privilege attribute based access control policies. In *ACSAC*, 2019.

[40] NIST. Security and privacy controls for information systems and organizations., 2020. URL https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf.

[41] Topic model, Feb 2023. URL https://en.wikipedia.org/wiki/Topic_model.

[42] Latent dirichlet allocation, Dec 2022. URL https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation.

[43] Non-negative matrix factorization, Feb 2023. URL https://en.wikipedia.org/wiki/Non-negative_matrix_factorization.

[44] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL http://arxiv.org/abs/1802.05365.

[45] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[46] The HF Datasets community. Bookcorpus · datasets at hugging face. URL https://huggingface.co/datasets/bookcorpus.

[47] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[48] Eliot Lear, Ralph Droms, and Dan Romascanu. Manufacturer usage description specification. Technical report, 2019. URL https://www.rfc-editor.org/rfc/rfc8520.html.

[49] Ayyoob Hamza, Hassan Habibi Gharakheili, and Vijay Sivaraman. Combining MUD policies with SDN for IoT intrusion detection. In *IoT SP*, 2018.

[50] Nest API reference, 2021. URL https://developers.nest.com/documentation/api-reference.

[51] Derek Miller. Leveraging BERT for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*, 2019.

[52] Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, et al. Review of automatic text summarization techniques & methods. *Journal of King Saud University-Computer and Information Sciences*, 2020.

[53] Shancang Li, Kim-Kwang Raymond Choo, Qindong Sun, William J Buchanan, and Jiuxin Cao. IoT forensics: Amazon echo as a use case. *IEEE Internet of Things Journal*, 6(4): 6487–6497, 2019.

[54] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara. Compiling finite linear CSP into SAT. *Constraints*, 14(2):254–272, 2009.

[55] Amazon IoT device simulator. URL https://aws.amazon.com/solutions/implementations/iot-device-simulator/.

[56] Snort. URL https://www.snort.org/.

[57] Suricata, Aug 2022. URL https://suricata.io/.

[58] The zeek network security monitor. URL https://zeek.org/.

[59] World's most widely used host intrusion detection system - hids. URL https://www.ossec.net/.

[60] SmartThingsCommunity. SmartThings SmartApp Node.js SDK. URL https://github.com/SmartThingsCommunity/smartapp-sdk-nodejs/blob/2fb4f4612e946a11b223531ca60557869d4abe49/README.md.

[61] Annotation tool for ner. *NER annotator*. URL https://tecoholic.github.io/ner-annotator/.

[62] Richmond Alake. Understanding cosine similarity and its application, Nov 2021. URL https://towardsdatascience.com/understanding-cosine-similarity-and-its-application-fd42f585296a.

[63] Jaro–winkler distance, 2022. URL https://en.wikipedia.org/wiki/JaroWinkler_distance.

[64] Sorensen–dice coefficient, Jul 2022. URL https://en.wikipedia.org/wiki/Sorensen-Dice_coefficient.

[65] Fatih Karabiber. Jaccard similarity. URL https://www.learndatasci.com/glossary/jaccard-similarity/.

[66] OWASP. OWASP IoT top 10 2018 mapping project., 2018. URL https://github.com/scriptingxss/OWASP-IoT-Top-10-2018-Mapping.