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

Md Wasiuddin Pathan Shuvo[1][0000−0002−4837−0655], Md Nazmul Hoq[1][0000−0002−1846−0537], Suryadipta Majumdar[1][0000−0002−6501−4214], and Paria Shirani[2][0000−0001−5592−1518]

[1] Concordia University, Montreal, Canada
wpshuvo57@gmail.com, mdnazmul.hoq@concordia.ca,
suryadipta.majumdar@concordia.ca
[2] University of Ottawa, Ottawa, Canada
pshirani@uottawa.ca

Abstract. 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 paper, 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 (NIST IR 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.

1 Introduction

Recent cyber-attacks are typically caused by various safety and security threats that result from implementation flaws and insecure default configurations [2, 15, 21, 38, 42, 45, 53]. 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.) [5]. Due to similar issues in implementing the best practices, several other recent attacks also led to severe security and safety consequences, such as unauthorized access to smart homes [21], injecting fake voice commands to smart home devices and hubs [49], and health hazards to infants in a smart home [14]. As a result, the accountability and transparency of those devices and their operations often become questionable [2]. This might be due to the fact that most security solutions (e.g., [12–14]) are not using standards as a basis for their security evaluation.

Several works (e.g., [4, 9–11, 14, 16, 25, 34, 54, 57]) 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 [5]).

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., NIST IR 8228 [8], ENISA [17], OWASP [40]) 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). This problem is further illustrated using a motivating example in Section 2.4.

In this paper, 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 the Internet of Things (IoT). More specifically, first, we conduct a study on major security standards such as [8,13,17,18,23,40]. 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 paper are as follows.

- 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, intrusion detection systems (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 paper is organized as follows. Section 2 describes the preliminaries and challenges. Section 3 details our methodology. Section 4 provides the application to different security mechanisms, and Section 5 presents the experimental results. Section 6 discusses different aspects of our approach. Section 7 reviews related works and compares them with our approach. Section 8 concludes the paper by providing future research directions.

2 Preliminaries

To keep our discussion more concrete, the rest of the paper will be on the scope of IoT standards and smart home networks. In the following, we provide backgrounds on common terminologies in security standards, review major IoT standards, and illustrate our motivating example.

2.1 Background on 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. 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.

2.2 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 1).

NIST IR 8228 [8]. 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

Table 1. Summary of different IoT security standards

Security Standard	Purpose	Targeted to	Scope	# pages
NIST IR 8228 [8]	Security and privacy risk management	Users	All kinds of IoT Devices	44
NIST IR 8259 [18]	Building secure IoT device	Manufacturers	All kinds of IoT Devices	36
ENISA [17]	Recommendation for baseline security	Users and manufacturers	All kinds of IoT Devices	103
ETSI EN 303 645 [23]	Consumer IoT security	Manufacturers	All kinds of IoT Devices	34
OWASP [40]	Secure building and usage of IoT	Manufacturer, developers and consumers	All kinds of IoT Devices	12
UK Govt [13]	Improve the security of consumer IoT	Manufacturers, developers and service providers	Smart homes and smart wearables	24

has listed 25 expectations along with 49 challenges to achieve those expectations and their mapping with the NIST SP 800-53r5 [37] for mitigating security and privacy risks in IoT systems.

NIST IR 8259 [18]. It is also an internal report from NIST, which is intended for IoT device manufacturers to assist them to 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 [17]. 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 can not be used as actionable security rules as they are insufficiently specific.

ETSI EN 303 645 - V2.1.1 [23]. 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 [7].

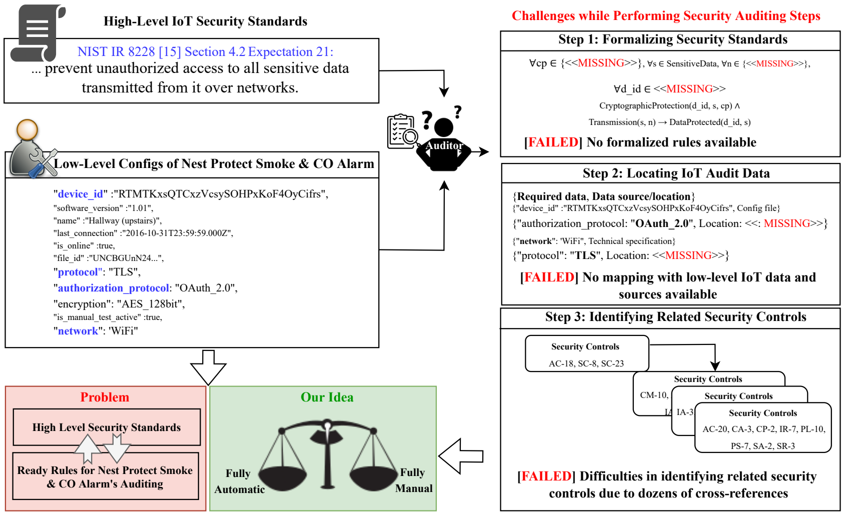
OWASP IoT Security Guidance [40]. The Open Worldwide Application Security Project (OWASP) Internet of Things Project has released the OWASP IoT top ten lists of IoT vulnerabilities in an effort to help manufacturers, developers, and consumers better understand IoT security risks and take appropriate 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 [13]. 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 [7].

2.3 Challenges in Deriving Actionable Rules from Standards

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

- Firstly, most existing security standards are provided at a 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 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 [7].


High-Level IoT Security Standards
NIST IR 8228 [15] Section 4.2 Expectation 21:
... prevent unauthorized access to all sensitive data transmitted from it over networks.
Low-Level Configs of Nest Protect Smoke & CO Alarm
Auditor
Challenges while Performing Security Auditing Steps
Step 1: Formalizing Security Standards
∀cp ∈ {<<MISSING>>}, ∀s ∈ SensitiveData, ∀n ∈ {<<MISSING>>},
∀d_id ∈ <<MISSING>>
CryptographicProtection(d_id, s, cp) ∧
Transmission(s, n) → DataProtected(d_id, s)
[FAILED] No formalized rules available
Step 2: Locating IoT Audit Data
{Required data, Data source/location}
{"device_id" :"RTMTKxsQTCxzVcsySOHPxKoF4OyCifrs", Config file}
{"authorization_protocol: "OAuth_2.0", Location: <<: MISSING>>}
{"network": 'WiFi", Technical specification}
{"protocol": "TLS", Location: <<MISSING>>}
[FAILED] No mapping with low-level IoT data and sources available
Step 3: Identifying Related Security Controls
Security Controls
AC-18, SC-8, SC-23
Security Controls
CM-10,
IA
Security Controls
IA-3
Security Controls
AC-20, CA-3, CP-2, IR-7, PL-10,
PS-7, SA-2, SR-3
[FAILED] Difficulties in identifying related security controls due to dozens of cross-references
Problem
High Level Security Standards
Ready Rules for Nest Protect Smoke & CO Alarm's Auditing
Our Idea
Fully Automatic
Fully Manual

```
"device_id" :"RTMTKxsQTCxzVcsySOHPxKoF4OyCifrs",
"software_version" :"1.01",
"name" :"Hallway (upstairs)",
"last_connection" :"2016-10-31T23:59:59.000Z",
"is_online" :true,
"file_id" :"UNCBGUnN24...",
"protocol": "TLS",
"authorization_protocol: "OAuth_2.0",
"encryption": "AES_128bit",
"is_manual_test_active" :true,
"network": 'WiFi"
```

Fig. 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

- Lastly, security standards contain too many different types of interrelated guidance on a single subject [52], 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 Section 3.

2.4 Motivating Example

A motivating example is shown in Figure 1 where an IoT security standard (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 [36]). 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 [8]. 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 those two extreme approaches and find a practical solution to derive actionable rules for IoT devices by proposing a semi-automated approach. In the following, we elaborate on our proposed approach.

3 Methodology

This section first provides an overview and then details our methodology.

3.1 Overview

The overview of the proposed methodology is shown in Figure 2.

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 3.2), we map various security standard recommendations with their controls in NIST SP 800-53r5 [37] 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 3.3), 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 3.5. Finally, we demonstrate the applicability of our approach in security applications such as security auditing (in Section 4). The details of each phase are described as follows.

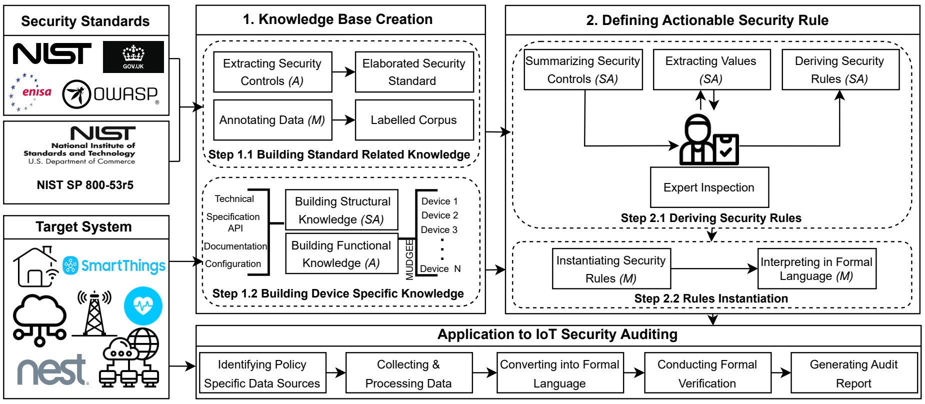
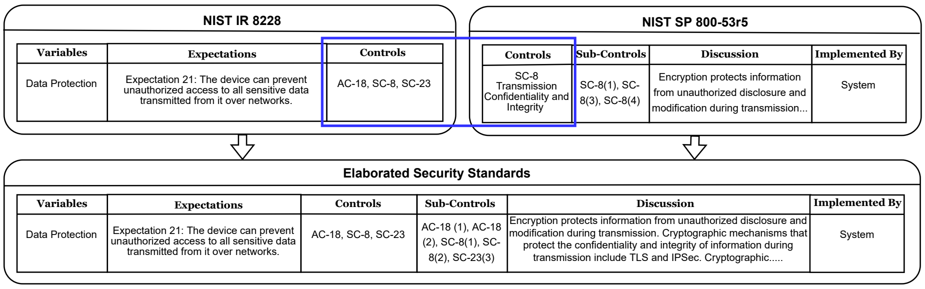

Security Standards
NIST
GOV.UK
enisa
OWASP
NIST National Institute of Standards and Technology U.S. Department of Commerce
NIST SP 800-53r5
Target System
SmartThings
nest
1. Knowledge Base Creation
Extracting Security Controls (A)
Elaborated Security Standard
Annotating Data (M)
Labelled Corpus
Step 1.1 Building Standard Related Knowledge
Technical Specification
API
Documentation
Configuration
Building Structural Knowledge (SA)
Building Functional Knowledge (A)
MUDGEE
Device 1
Device 2
Device 3
Device N
Step 1.2 Building Device Specific Knowledge
2. Defining Actionable Security Rule
Summarizing Security Controls (SA)
Extracting Values (SA)
Deriving Security Rules (SA)
Expert Inspection
Step 2.1 Deriving Security Rules
Instantiating Security Rules (M)
Interpreting in Formal Language (M)
Step 2.2 Rules Instantiation
Application to IoT Security Auditing
Identifying Policy Specific Data Sources
Collecting & Processing Data
Converting into Formal Language
Conducting Formal Verification
Generating Audit Report

Fig. 2. An overview of our methodology (where (A): fully automated step, (SA): semi-automated step, and (M): manual step)

3.2 Knowledge Base Creation

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

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 [37] (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 3 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) [12] model with annotated security controls and extract both values and attributes by utilizing the learned model.



NIST IR 8228

Variables	Expectations	Controls
Data Protection	Expectation 21: The device can prevent unauthorized access to all sensitive data transmitted from it over networks.	AC-18, SC-8, SC-23

NIST SP 800-53r5

Controls	Sub-Controls	Discussion	Implemented By
SC-8 Transmission Confidentiality and Integrity	SC-8(1), SC-8(3), SC-8(4)	Encryption protects information from unauthorized disclosure and modification during transmission...	System

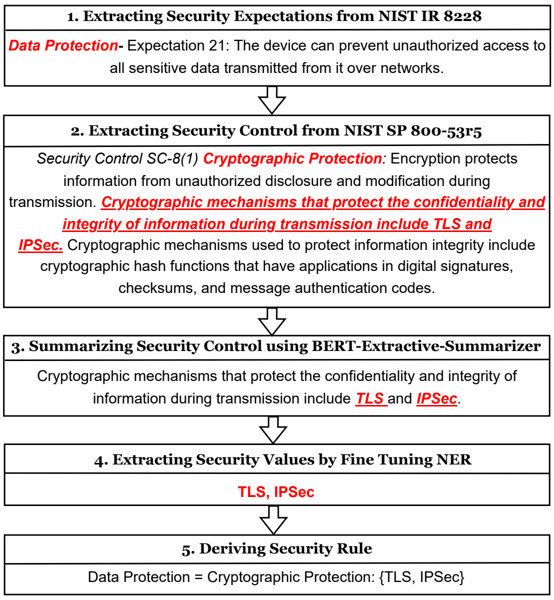
Elaborated Security Standards

Variables	Expectations	Controls	Sub-Controls	Discussion	Implemented By
Data Protection	Expectation 21: The device can prevent unauthorized access to all sensitive data transmitted from it over networks.	AC-18, SC-8, SC-23	AC-18 (1), AC-18 (2), SC-8(1), SC-8(2), SC-23(3)	Encryption protects information from unauthorized disclosure and modification during transmission. Cryptographic mechanisms that protect the confidentiality and integrity of information during transmission include TLS and IPSec. Cryptographic.....	System

Fig. 3. Development and Arrangement of Elaborated Security Standards

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 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, 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.

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. [15] as follows. First, we gather all the technical information that is provided on a device's website. Second, we extract that 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 [15]. 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



1. Extracting Security Expectations from NIST IR 8228

Data Protection- Expectation 21: The device can prevent unauthorized access to all sensitive data transmitted from it over networks.

2. Extracting Security Control from NIST SP 800-53r5

Security Control SC-8(1) Cryptographic Protection: Encryption protects information from unauthorized disclosure and modification during transmission. *Cryptographic mechanisms that protect the confidentiality and integrity of information during transmission include TLS and IPSec.* Cryptographic mechanisms used to protect information integrity include cryptographic hash functions that have applications in digital signatures, checksums, and message authentication codes.

3. Summarizing Security Control using BERT-Extractive-Summarizer

Cryptographic mechanisms that protect the confidentiality and integrity of information during transmission include *TLS* and *IPSec*.

4. Extracting Security Values by Fine Tuning NER

TLS, IPSec

5. Deriving Security Rule

Data Protection = Cryptographic Protection: {TLS, IPSec}

Fig. 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.

behaviour [27]) profiles using MUDGEE [20]. 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 [36] and API documentation [35], 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: true*". The corresponding functional knowledge obtained by using MUDGEE is: "*IP Protocols: TCP, UDP, HOPOPT, IPv6-ICMP; and Ports: 443, 11095, 53, 67*".

3.3 Defining Actionable Security Rules

We describe how we define actionable rules in the following.

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 3.2) utilizing BERT-Extractive-Summarizer [33], 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 [33]. Text summarization is still an ongoing topic in NLP research to achieve a competitive accuracy to that of a human [55]. Due to this factor, our generated summaries need to be reviewed by experts to ensure their correctness.

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 3.2 in order to use them as training data to fine-tune a Hugging Face model (e.g., BERT-base-NER [12]). 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.

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. 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`}}*".

3.4 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 3.2 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, 31]. Table 2 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: `peyiJNoOIldT2YlIVtYaGQ`) `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:

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

3.5 Rationale behind Our Semi-Automated Approach

Table 3 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.

4 Applications

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

Table 2. An excerpt of derived and instantiated security rules using our approach

Sub-Controls	Summaries	Derived Rules	Instantiated Rules
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.

4.1 Application to IoT Security Auditing

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 [28]. 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 3.4). Listing 1 shows the tuples in our Sugar [50] code.

Table 3. Different steps of our approach, their objectives, and rationales

Steps	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 in [15] 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 [20], which automatically delivers the IoT network port and protocol number without manual inspection.
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.
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.

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 [50], 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.

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 [3]. Then, the data protection at transmission is declared as a predicate, denoted by *DataProtectionTransmission*, 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 verifying each instance of the problem among possible values of `device ID, cryptographic mechanism,` and `network types`. We evaluate this auditing step in Section 5.

Listing 1.1. Sugar source code for verifying Rule 1

```
 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)

 6 // Relations Declarations and Audit Data as their Support
 7 (relation CryptProtection 3 (supports ((2471 13 4) (2798
       29 2) (861 9 4) ))
 8 (relation Transmission 2 (supports ((12 9) (29 10) (9 1) )
       )

 9 // Security property: DataProtectionTransmission
10 (predicate (DataProtectionTransmission D S CR N) (and (
       CryptProtection D S CR) (Transmission S N) (not (
       DataProtection D S)) ))
11 (DataProtectionTransmission D S CR N)
```

4.2 Other Applications

Snort IDS. Snort [47], 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, 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[3], Zeek[4], OSSEC[5], 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: peyiJNoOIldT2YlIVtYaGQ) 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`.

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 [46] to listen to the server (Line 6).

[3] https://suricata.io/

[4] https://zeek.org/

[5] https://www.ossec.net/

Listing 1.2. Port numbers derived from our security rules used by SmartThings SmartApp

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

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

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 of IoT.

5 Implementation and Experiments

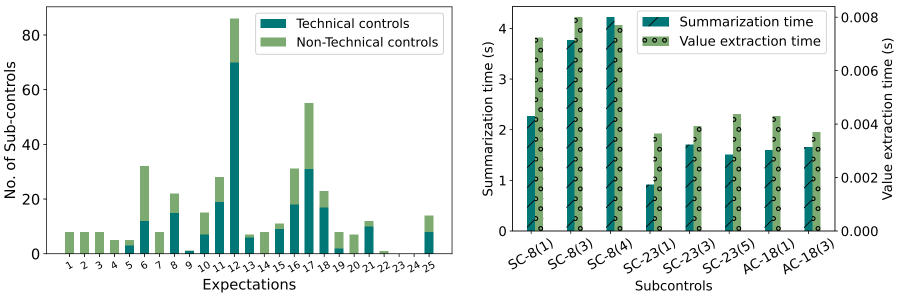
This section describes the details of our implementation and experiments.

5.1 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 [20], which creates MUD [27] profiles of IoT devices by monitoring network traces and technical specifications of different IoT devices are extracted by leveraging the approach from [15]. For the summarization of security controls, we use BERT-Extractive-Summarizer [33]. Then, annotation of security controls is performed using NER Annotator [6]. To extract values from security controls, we fine-tune a Hugging Face transformer-based named entity recognition model called Bert-base-uncased [12]. Lastly, for verification, we use the Boolean satisfaction (SAT) solver tool, namely, Sugar V2.2.1 [50].

5.2 Experiments

Experimental Setting. 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 [8] and NIST SP 800-53r5 [37] and Amazon IoT device simulator platform [3] with 5,000 IoT devices and their logs, configuration files, and network data. Figure 5a illustrates the count of technical and non-technical security controls for each expectation of NIST IR 8228. We



(a) Count of security controls for each expectation from NIST IR 8228 [8]

(b) Time required for summarizing and value extraction by our approach

Fig. 5. Count of security controls and efficiency of summarization and extraction

Table 4. Performance evaluation of value extraction

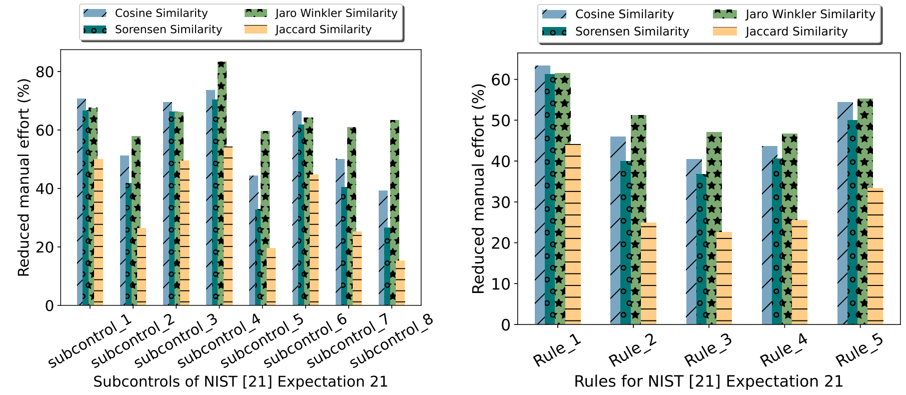
	Precision	**Recall**	**F1-Score**
Values	0.82	0.98	0.89
Attributes	0.97	0.94	0.95
Average	0.87	0.97	0.91

convert them into the input format, Constraint Satisfaction Problem (CSP), of Sugar [50]. We average the results after 200 iterations of each experiment.

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 5b 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 others, and 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 4, 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.

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.

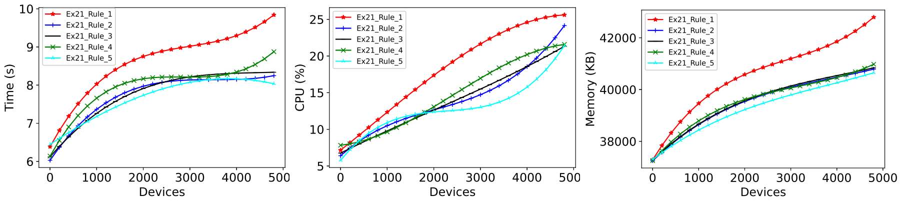
Our approach aims at reducing the manual effort required by an expert for deriving actionable security rules. Figure 6 demonstrates the amount of reduction in manual effort for summarizing and deriving actionable security rules, where we compare a fully manual approach with ours for this measurement using four similarity metrics (e.g., Cosine similarity [1], Jaro-Winkler similarity [24], Sorensen similarity [48], and Jaccard similarity metrics [26]). 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 6a 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 6b 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 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 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 Section 6).



(a) Similarity score of summarized sub-control (b) Similarity score of derived rules

Fig. 6. Manual effort reduction for summarizing and deriving rules

We then evaluate the efficiency of our derived security rules in terms of time, CPU, and memory utilization. In Figure 7a, 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 7b shows the CPU usage by varying the number of devices. With a range of between 20% and 25%, CPU utilization increases 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 (Figures 7c) 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.



(a) Time requirement (b) CPU utilization (c) Memory utilization

Fig. 7. Efficiency results of our auditing step for 5,000 smart home devices.

6 Discussion

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.

Covering Other Security Standards. In this paper, 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 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 [17] 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 [39]. Once available, those mappings can be utilized to cover other security standards using our approach.

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.

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 such feedback to the standardization authorities that might be useful to design future standards in a clearer and more useful manner.

7 Related Works

This section reviews existing IoT security works and compares them with ours. We first review rule-based IoT security solutions. Fung et al. [19] introduce a user-defined rule-sharing model for the IoT environment and track the reputation of rules based on the feedback of different rules adopted by users without the need for central facilities. PFIREWALL [11] generates data minimization rules to control data flow by filtering communication between devices and platforms of IoT systems to reduce data leakage. IoTSAFE [14] performs static analysis and dynamic testing to identify run-time physical interactions in the IoT environment to enforce security rules. Soteria [9] verifies the safety and security rules of IoT platforms by performing static code analysis in IoT applications. On the other hand, IoTGuard [10] is a dynamic safety and security rule enforcement system by code instrumentation. Nespoli et al. [34] detect vulnerabilities in IoT and dynamically adapt to surrounding devices and services based on rules. Dome et al. [16] utilize the Random-Forest model to convert behavioral patterns into rules, build a threat prediction model, and monitor rule violations. Majumdar et al. [31] [32] conduct security auditing leveraging formal techniques in cloud platforms. Madi et al. [30] also carried out security auditing on a cloud platform by proposing an auditing framework for OpenStack. Most of the above-mentioned works develop their own rules, whereas our goal is to utilize existing security standards.

Another line of research focuses on access control, monitoring, and intrusion detection of IoT systems. ContexIoT [25] is a context-based permission system to provide contextual integrity for different IoT platforms. SmartAuth [51] uses static analysis to collect security-related data from IoT apps to design authorization procedures to address over-privileged issues in IoT systems. HoMonit [57] utilizes side-channel techniques to monitor the encrypted traffic of IoT systems. IoTArgos [54] is a multi-layer security monitoring system that uses machine learning techniques to find intrusions and anomalies in IoT platforms. Anthi et al. [4] develop a multi-layer intrusion detection system. In contrast, our work provides actionable security rules which can be utilized in various security applications.

We also review the association rule mining works. P. Lou et al. [29] obtain 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. Ozawa et al. [41] use association rule mining to discover regularities in darknet data. Husak et al. [22] use sequential rule mining to analyze intrusion detection alerts and to predict security events for creating a predictive blacklist. Safara et al. [43] use an association rule mining algorithm to extract appropriate features from raw data and then use the features for detecting anomalies in communication networks. Xu et al. [56] propose an Attribute-based access control (ABAC) policy mining algorithm. Sanders et al. [44] use the rule mining approach to analyze systems' audit logs for automatically generating ABAC policies. Unlike them, this work derives actionable security rules from IoT standards.

Comparative Study. Table 5 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 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 be used directly for various security applications.

Table 5. 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**	**Using Security Standards**
ContextIoT [25]	Custom Algorithm	IoT	Access Control	NA	–	–	NA	○	●	–
Soteria [9]	Static Analysis	IoT	Intrusion Detection	–	–	–	NA	○	–	–
IoTGuard [10]	Dynamic Analysis	IoT	Intrusion Detection	–	–	–	NA	●	●	–
Majumdar et al. [31]	Formal Method	Cloud	Auditing	–	●	–	●	○	–	○
Madi et al. [30]	Formal Method	Cloud	Auditing	–	●	–	●	NA	–	○
Majumdar et al. [32]	Formal Method	Cloud	Auditing	–	●	–	●	NA	●	○
Homonit [57]	Custom Algorithm	IoT	Monitoring System	NA	–	–	–	–	–	–
IoTSafe [14]	Static and Dynamic	IoT	Intrusion Detection	NA	–	–	–	●	–	–
PFIREWALL [11]	Custom Algorithm	IoT	Access Control	NA	–	○	○	●	●	–
This Work	Formal Method	IoT	Auditing, Secure development, etc.	●	●	○	●	○	○	●

8 Conclusion

This paper proposed an approach to derive actionable security rules from high-level security standards. Additionally, we collected device-specific data to instantiate derived security rules and conducted verification leveraging formal tools for IoT devices. Our experiment results showed the effectiveness of our derived rules in security auditing. Moreover, our derived actionable security rules can be utilized in other security applications. In the following steps, we envision being able to automate device-specific data collection, which is collected manually now. Our future work will incorporate more security standards into our methodology and automate the majority of the stages involved in the derivation of actionable security rules, requiring the least amount of work from security specialists. Additionally, in our future work, we intend to conduct a user study to evaluate the usability of our solution from the feedback of real-world security practitioners.

Acknowledgments

The authors thank the anonymous reviewers and our shepherd, Christopher Wood, for their valuable comments. This material is based upon work supported by the Natural Sciences and Engineering Research Council of Canada and Department of National Defence Canada under the Discovery Grants RGPIN-2021-04106 and DGDND-2021-04106.

References

1. Alake, R.: Understanding cosine similarity and its application (Nov 2021), https://towardsdatascience.com/understanding-cosine-similarity-and-its-application-fd42f585296a
2. Alrawi, O., Lever, C., Antonakakis, M., Monrose, F.: SoK: Security evaluation of home-based IoT deployments. In: IEEE SP. IEEE (2019)
3. Amazon IoT device simulator, https://aws.amazon.com/solutions/implementations/iot-device-simulator/
4. Anthi, E., Williams, L., Słowińska, M., Theodorakopoulos, G., Burnap, P.: A supervised intrusion detection system for smart home iot devices. IEEE Internet of Things Journal **6**(5), 9042–9053 (2019)
5. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al.: Understanding the Mirai botnet. In: USENIX Security (2017)
6. Arunmozhi: Annotation tool for ner. NER annotator (2022), https://tecoholic.github.io/ner-annotator/
7. Bellman, C., van Oorschot, P.C.: Systematic analysis and comparison of security advice as datasets. Computers & Security **124**, 102989 (2023)
8. Boeckl, K., Boeckl, K., Fagan, M., Fisher, W., Lefkovitz, N., Megas, K.N., Nadeau, E., O'Rourke, D.G., Piccarreta, B., Scarfone, K.: Considerations for managing Internet of Things (IoT) cybersecurity and privacy risks. US Department of Commerce, National Institute of Standards and Technology (2019)
9. Celik, Z.B., McDaniel, P., Tan, G.: Soteria: Automated IoT safety and security analysis. In: USENIX ATC. pp. 147–158 (2018)
10. Celik, Z.B., Tan, G., McDaniel, P.D.: IoTGuard: Dynamic enforcement of security and safety policy in commodity IoT. In: NDSS (2019)
11. Chi, H., Zeng, Q., Du, X., Luo, L.: PFIREWALL: Semantics-aware customizable data flow control for smart home privacy protection. arXiv preprint arXiv:2101.10522 (2021)
12. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
13. Department for Digital, Culture, M..S.: The UK government. code of practice for consumer IoT security (2019), https://www.gov.uk/government/publications/code-of-practice-for-consumer-iot-security
14. Ding, W., Hu, H., Cheng, L.: IOTSAFE: Enforcing safety and security policy with real IoT physical interaction discovery. In: NDSS (2021)
15. Dolan, A., Ray, I., Majumdar, S.: Proactively extracting iot device capabilities: An application to smart homes. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 42–63. Springer (2020)
16. Domb, M., Bonchek-Dokow, E., Leshem, G.: Lightweight adaptive random-forest for IoT rule generation and execution. Journal of Information Security and Applications **34**, 218–224 (2017)
17. ENISA, E.: Baseline security recommendations for IoT in the context of critical information infrastructures. European Union Agency for Cybersecurity Heraklion, Greece (2017)
18. Fagan, M., Megas, K., Scarfone, K., Smith, M.: Recommendations for IoT device manufacturers: Foundational activities and core device cybersecurity capability baseline (2nd draft). Tech. rep., National Institute of Standards and Technology (2020)

19. Fung, C.J., McCormick, B.: An effective policy sharing mechanism for smart home networks. In: IEEE CNSM. IEEE (2020)
20. Hamza, A., Gharakheili, H.H., Sivaraman, V.: Combining MUD policies with SDN for IoT intrusion detection. In: IoT S&P (2018)
21. Ho, G., Leung, D., Mishra, P., Hosseini, A., Song, D., Wagner, D.: Smart locks: Lessons for securing commodity internet of things devices. In: ACM ASIACCS. pp. 461–472 (2016)
22. Husák, M., Bajtoš, T., Kašpar, J., Bou-Harb, E., Čeleda, P.: Predictive cyber situational awareness and personalized blacklisting: a sequential rule mining approach. ACM Transactions on Management Information Systems (TMIS) **11**(4), 1–16 (2020)
23. Institute, E.T.S.: En 303 645 - v2.1.1 - cyber; cyber security for consumer internet of things: Baseline requirements (2020), https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf
24. Jaro–winkler distance (2022), https://en.wikipedia.org/wiki/JaroWinkler_distance
25. Jia, Y.J., Chen, Q.A., Wang, S., Rahmati, A., Fernandes, E., Mao, Z.M., Prakash, A., Unviersity, S.: ContexIoT: Towards providing contextual integrity to appified IoT platforms. In: NDSS (2017)
26. Karabiber, F.: Jaccard similarity, https://www.learndatasci.com/glossary/jaccard-similarity/
27. Lear, E., Droms, R., Romascanu, D.: Manufacturer usage description specification. Tech. rep., Internet Engineering Task Force (2019)
28. Li, S., Choo, K.K.R., Sun, Q., Buchanan, W.J., Cao, J.: IoT forensics: Amazon echo as a use case. IEEE Internet of Things Journal **6**(4), 6487–6497 (2019)
29. Lou, P., Lu, G., Jiang, X., Xiao, Z., Hu, J., Yan, J.: Cyber intrusion detection through association rule mining on multi-source logs. Applied Intelligence **51**(6), 4043–4057 (2021)
30. Madi, T., Majumdar, S., Wang, Y., Jarraya, Y., Pourzandi, M., Wang, L.: Auditing security compliance of the virtualized infrastructure in the cloud: Application to OpenStack. In: ACM CODASPY (2016)
31. Majumdar, S., Madi, T., Wang, Y., Jarraya, Y., Pourzandi, M., Wang, L., Debbabi, M.: Security compliance auditing of identity and access management in the cloud: Application to OpenStack. In: IEEE CloudCom. IEEE (2015)
32. Majumdar, S., Madi, T., Wang, Y., Jarraya, Y., Pourzandi, M., Wang, L., Debbabi, M.: User-level runtime security auditing for the cloud. IEEE Transactions on Information Forensics and Security **13**(5), 1185–1199 (2017)
33. Miller, D.: Leveraging BERT for extractive text summarization on lectures. arXiv preprint arXiv:1906.04165 (2019)
34. Nespoli, P., Díaz-López, D., Mármol, F.G.: Cyberprotection in IoT environments: A dynamic rule-based solution to defend smart devices. Journal of Information Security and Applications **60**, 102878 (2021)
35. Nest API reference, https://developers.nest.com/documentation/api-reference
36. Nest protect and CO alarm, https://store.google.com/product/nest_protect_2nd_gen_specs?hl=en-US
37. NIST: Security and privacy controls for information systems and organizations., https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

38. Notra, S., Siddiqi, M., Gharakheili, H.H., Sivaraman, V., Boreli, R.: An experimental study of security and privacy risks with emerging household appliances. In: IEEE CNS. IEEE (2014)
39. OWASP: OWASP IoT top 10 2018 mapping project., https://github.com/scriptingxss/OWASP-IoT-Top-10-2018-Mapping
40. OWASP: OWASP top 10 Internet of Things 2018 (2018), https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf
41. Ozawa, S., Ban, T., Hashimoto, N., Nakazato, J., Shimamura, J.: A study of IoT malware activities using association rule learning for darknet sensor data. International Journal of Information Security **19**(1), 83–92 (2020)
42. Ronen, E., Shamir, A.: Extended functionality attacks on iot devices: The case of smart lights. In: IEEE EuroS&P. IEEE (2016)
43. Safara, F., Souri, A., Serrizadeh, M.: Improved intrusion detection method for communication networks using association rule mining and artificial neural networks. IET Communications **14**(7), 1192–1197 (2020)
44. Sanders, M.W., Yue, C.: Mining least privilege attribute based access control policies. In: ACSAC (2019)
45. Sivaraman, V., Chan, D., Earl, D., Boreli, R.: Smart-phones attacking smart-homes. In: ACM WiSec (2016)
46. SmartThingsCommunity: SmartThings SmartApp Node.js SDK, https://github.com/SmartThingsCommunity/smartapp-sdk-nodejs/blob/2fb4f4612e946a11b223531ca60557869d4abe49/README.md
47. Snort, https://www.snort.org/
48. Sorensen–dice coefficient (Jul 2022), https://en.wikipedia.org/wiki/Sorensen-Dice_coefficient
49. Sugawara, T., Cyr, B., Rampazzi, S., Genkin, D., Fu, K.: Light commands: Laser-Based audio injection attacks on Voice-Controllable systems. In: USENIX Security (2020)
50. Tamura, N., Taga, A., Kitagawa, S., Banbara, M.: Compiling finite linear CSP into SAT. Constraints **14**(2), 254–272 (2009)
51. Tian, Y., Zhang, N., Lin, Y.H., Wang, X., Ur, B., Guo, X., Tague, P.: Smartauth: User-centered authorization for the internet of things. In: USENIX Security (2017)
52. Verry, J.: Should I use NIST 8228 or NIST 8259 for IoT design or IoT testing? (Jun 2020), https://www.pivotpointsecurity.com/should-i-use-nist-8228-or-nist-8259-for-iot-design-or-iot-testing/
53. Vervier, P.A., Shen, Y.: Before toasters rise up: A view into the emerging iot threat landscape. In: International Symposium on Research in Attacks, Intrusions, and Defenses. pp. 556–576. Springer (2018)
54. Wan, Y., Xu, K., Xue, G., Wang, F.: IoTArgos: A multi-layer security monitoring system for internet-of-things in smart homes. In: IEEE INFOCOM. IEEE (2020)
55. Widyassari, A.P., Rustad, S., Shidik, G.F., Noersasongko, E., Syukur, A., Affandy, A., et al.: Review of automatic text summarization techniques & methods. Journal of King Saud University-Computer and Information Sciences (2020)
56. Xu, Z., Stoller, S.D.: Mining attribute-based access control policies. IEEE Transactions on Dependable and Secure Computing **12**(5), 533–545 (2014)
57. Zhang, W., Meng, Y., Liu, Y., Zhang, X., Zhang, Y., Zhu, H.: HoMonit: Monitoring smart home apps from encrypted traffic. In: ACM CCS (2018)