Risk Prediction of IoT Devices Based on Vulnerability Analysis

PASCAL OSER, Ulm University and European Organization for Nuclear Research (CERN)
RENS W. VAN DER HEIJDEN, Independent Researcher
STEFAN LÜDERS, European Organization for Nuclear Research (CERN)
FRANK KARGL, Ulm University

Internet of Things (IoT) devices are becoming more widespread not only in areas such as smart homes and smart cities but also in research and office environments. The sheer number, heterogeneity, and limited patch availability provide significant challenges for the security of both office networks and the Internet in general. The systematic estimation of device risks, which is essential for mitigation decisions, is currently a skill-intensive task that requires expertise in network vulnerability scanning, as well as manual effort in firmware binary analysis.

This article introduces SAFER,[1] the Security Assessment Framework for Embedded-device Risks, which enables a semi-automated risk assessment of IoT devices in any network. SAFER combines information from network device identification and automated firmware analysis to estimate the current risk associated with the device. Based on past vulnerability data and vendor patch intervals for device models, SAFER extrapolates those observations into the future using different automatically parameterized prediction models. Based on that, SAFER also estimates an indicator for future security risks. This enables users to be aware of devices exposing high risks in the future.

One major strength of SAFER over other approaches is its scalability, achieved through significant automation. To demonstrate this strength, we apply SAFER in the network of a large multinational organization, to systematically assess the security level of hundreds of IoT devices on large-scale networks.

Results indicate that SAFER successfully identified 531 out of 572 devices leading to a device identification rate of 92.83 %, analyzed 825 firmware images, and predicted the current and future security risk for 240 devices.

CCS Concepts: • **Security and privacy** → **Distributed systems security**; • **Computer systems organization** → **Embedded systems**; • **Networks** → **Cyber-physical networks**; **Network management;**

Additional Key Words and Phrases: IoT, security risk assessment, device identification, firmware analysis, vulnerability analysis, risk prediction, future risk, safer network, CERN

[1]We plan to publish SAFER in 2022 on https://safer.network.

This work was sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research (grant no. 05E15CHA).

Authors' addresses: P. Oser, Ulm University, Ulm, Germany, European Organization for Nuclear Research (CERN), Geneva, Switzerland; email: pascal@safer.network; R. W. van der Heijden, Independent researcher, Nuremberg, Germany; email: rensvdheijden@gmail.com; S. Lüders, European Organization for Nuclear Research (CERN), Geneva, Switzerland; email: stefan.lueders@cern.ch; F. Kargl, Ulm University, Ulm, Germany; email: frank.kargl@uni-ulm.de.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2022 Copyright held by the owner/author(s).

2471-2566/2022/05-ART14
https://doi.org/10.1145/3510360

ACM Reference format:
Pascal Oser, Rens W. van der Heijden, Stefan Lüders, and Frank Kargl. 2022. Risk Prediction of IoT Devices Based on Vulnerability Analysis. *ACM Trans. Priv. Secur.* 25, 2, Article 14 (May 2022), 36 pages.
https://doi.org/10.1145/3510360

1 INTRODUCTION

Today, the **Internet of Things (IoT)** is not just a vision or research topic [41], it is already becoming reality. Examples of IoT systems include smart homes with smart door locks, garden mowers, or smart fridges; smart cities where sensor networks measure air quality or display free parking spots; and companies and organizations where smart buildings manage energy and heating or CCTV cameras monitor premises. According to Siemens,[2] the number of IoT devices worldwide is estimated to reach 75 billion by 2025.

This work is motivated by our own environment at the **European Organization for Nuclear Research (CERN)**, a large multinational research organization where thousands of visiting researchers bring IoT devices into the network for research purposes. Devices includes networked oscilloscopes, storage, printers, cameras, all kinds of networked sensors that researchers can easily integrate into the network with the help of a bring-your-own-device policy.

Many of the IoT devices found at CERN are not regularly maintained, as no single person or team can keep an overview over all the types of devices and the firmware deployed on those devices. Over time, CERN's network therefore accumulates a large number of devices with vulnerabilities that may be entry points for attacks [40, 46, 57]. Even though CERN's network may be an extreme case, many companies, universities, and even private networks face a similar challenge, and with more and more IoT devices being deployed, the problem only becomes bigger. A first step toward mitigation is to assess the risk such devices pose to the network. Already in 2017, researchers published first works on new methodologies for IoT risk assessments [43] and simple scripts to automate security testing of IoT devices [36], but those approaches are very limited in their accuracy and scalability and cannot address the problem on the scale of an organization like CERN. There scalability and automation become essential features. Thus, developing a fully or at least highly automated risk assessment framework that can deal with a broad variety of devices and is easily extensible for new devices is needed to cope with the plethora of models and firmware versions, to provide a time-saving, reproducible, and consistent assessment. This does not come without further non-trivial automation challenges to assess the security risk of devices, which we address in this work. We believe that such a framework should even go one step further. Beyond simply assessing the risk that a device poses to the network today, it should also indicate if a device is likely to create security problems in the future because the manufacturer or model has a history of critical vulnerabilities and patches were not provided in a timely manner.

In our work, we established such a framework, which we call *SAFER*, the Security Assessment Framework for Embedded-device Risks. SAFER assesses the security risks of IoT devices and predicts how the security of such devices may evolve in the future with a very high degree of automation. We focus on IoT devices because they show a larger heterogeneity than typical desktops and servers, and exemplify the problems outlined in the following in a very strong form. When designing SAFER, we needed to establish new approaches to identify device models and their firmware, new risk metrics to assess current and predict future risks, and new ways to automate these processes. We developed and tested SAFER in the unique environment at CERN, which provided a unique opportunity for evaluating our approaches.

[2]https://new.siemens.com/global/en/company/stories/research-technologies/digitaltwin/iot-story.html.

To perform a risk analysis, SAFER first tries to automatically infer the IoT device's model and firmware version to then retrieve the device's vulnerabilities. SAFER then retrieves and analyzes corresponding firmware images, and retrieves known vulnerabilities from online databases like the **National Vulnerability Database (NVD)**.[3] Based on identified vulnerabilities, SAFER then determines the **current device security risk indicator (CDSRI)**.

But SAFER goes one step further: it also analyzes the history of earlier vulnerabilities of device models and manufacturers, and how quickly those vulnerabilities have been patched. This is based, for example, not only on NVD data but also on automated analysis of firmware release notes. From this data, we then derive a **future device security risk indicator (FDSRI)** that aims to predict how likely new vulnerabilities may emerge in the future (the so-called **vulnerability trend (VT)**) and how long it may take the manufacturer to fix them (the so-called **patch trend (PT)**). This allows SAFER to predict how likely devices that may currently be considered "secure" may turn into a security risk in the future or whether currently risky devices, once regularly patched, may be operated securely.

This article presents three main contributions:

(1) In Section 4, we describe a device identification approach that relies on different identification mechanisms and a fusion mechanism that allows us to identify not only device models but also firmware running on those devices. This includes four sub-contributions:
   - An enhanced variant of a previously published identification mechanism called *CCD* that relies on TCP timestamps for identifying device models (Section 4.3.1);
   - A novel identification mechanism called *WPD* that retrieves information from device web pages to identify models and firmware versions (Section 4.3.2);
   - A fusion algorithm relying on subjective logic to merge results from different identification mechanisms (Section 4.2); and
   - An extensive evaluation of the overall identification mechanism in a large-scale, real-world test bed (Section 4.4).

(2) In Section 5, we introduce and evaluate different new risk metrics that are calculated based on identification results and information retrieved automatically from public sources like firmware repositories of vendors, CVE data, and firmware patches from vendors. We provide the following sub-contributions:
   - The CDSRI, which provides information about the current security risk of a specific IoT device as of the time of evaluation (Section 5.3);
   - The FDSRI, which aims to predict how likely this device may create security risks in the future. For this, it relies on information from past vulnerabilities and how quickly they were patched by its vendor (Section 5.4); and
   - An evaluation of 38 different device models to illustrate the framework's capability to accurately predict the respective risks.

(3) Those concepts have been implemented with a high degree of automation and integrated into the SAFER framework as discussed in Sections 3.1 and 6.

The rest of the article is structured as follows. We introduce related work in Section 2. An overview of SAFER including details of its main components is given in Section 3. Section 4 then discusses results for contribution 1. In particular, we present how subjective logic [27] is used to fuse identification results of two exemplary IoT device identification mechanisms (Sections 4.3.1 and 4.3.2). Section 5 then introduces results toward contribution 2 and discusses how risks for the

[3] http://nvd.nist.gov/.

identified devices can be assessed in a highly automated way. First, Section 5.1 presents a component that automatically retrieves and analyzes firmware images for contained vulnerabilities. Based on found vulnerabilities, Section 5.3 then discusses how to calculate the CDSRI. Section 5.4 introduces two different trend values required to establish the FDSRI. Section 6 provides an overall discussion of the framework and some of its details regarding the automation of tasks and puts it in relation to related work. This also completes contribution 3. We conclude the article in Section 7 with a summary and an outlook on future work.

2 RELATED WORK

In this section, we introduce the related work of SAFER. Since SAFER consists of multiple components focusing on different tasks, we group the related work by contributions.

2.1 Device Identification

This section discusses related work that identifies devices using different approaches and relates to our first contribution. Feng et al. [19] propose ARE, an acquisitional rule-based engine to automatically generate rules for discovering and annotating IoT devices without any training data. ARE extracts device information from the website of the IoT device to map it to a product description. Authors use ARE to detect the category, vendor, and model.

Bezawada et al. [5] propose a method to identify device models based on IoT behaviors their solution IoTSense [6] utilizes. The authors train a machine learning model with passively collected network data of different home IoT devices and claim a identification rate of 93% to 100%. Huang et al. [24] introduce IoTInspector, passively scanning IoT device traffic via ARP spoofing being a non-applicable approach in large networks. IoTScanner developed by Siby et al. [54] focuses more on the privacy aspect of IoT devices. However, the framework identifies IoT device models of wireless networks by observing the link layer. IoTScanner analyzes the network traffic and uses the frame header information for device identification. The authors evaluate their framework with six IoT devices and prove the general feasibility of their approach.

2.2 Device Protection

This section focuses on protecting IoT devices from attacks. Zheng et al. [65] propose IoTAegis, which focuses on preventing the compromise of IoT devices by following a *device hygiene* approach. The authors show IoTAegis uploading the most recent firmware to a Hewlett-Packard printer or changing a weak user password to a random generated password. George and Thampi [22] propose a graph-based security framework to secure industrial IoT networks from vulnerability exploitation. Feng et al. [20] analyze source code of real-world IoT exploits and vulnerability reports found in attack toolkits, exploit databases, blogs and forums on the Internet, honeypots, and underground markets. The authors apply natural language processing on those reports to automatically create Snort rules to block these attacks.

2.3 Firmware Analysis and Fuzzing

This section discusses related work that identifies publicly known vulnerabilities or creates new device vulnerabilities and relates to our third contribution. Costin et al. [14] emulate embedded firmware images to perform dynamic analysis on. This lead authors to discover multiple vulnerabilities. Chen et al. [11] introduce IoTFuzzer to find memory corruptions in IoT devices without access to firmware images. The authors are able to cause software crashes but lack the discovery of more attack vectors, like authentication bypass. For some devices, the authors needed to verify the discovered vulnerability and opened devices to manually verify software crashes via debugging ports. Overall, IoTFuzzer is an approach to identify implementation weaknesses and to higher IoT software quality. Srivastava et al. [55] present FirmFuzz, an automated firmware analysis framework using dynamic analysis. It analyzes Linux-based firmware images and identifies bugs using e.g. a generational fuzzing approach. The authors used FirmFuzz to analyze 32 images of 27 unique devices.

2.4 Security Scoring

This section focuses on our second contribution: the challenges of aggregating vulnerabilities to device risk scores. Multiple researchers work on specialized versions of the **Common Vulnerability Scoring System (CVSS)** [23, 48, 58, 59]. With SAFER, we assess a high amount of different device types and hence need a scoring system able to generate universal and comparable metrics.

2.5 Security Frameworks

This section discusses related work on full framework solutions in comparison to preceding works focusing on parts thereof. Alrawi et al. [3] introduce SoK and provide two types of contributions: first the literature review and second the home IoT assessment. They assess home IoT devices and did their assessment on 45 devices. Their approach is working with passive gathering of IoT network traffic. Mohsin et al. [39] propose the IoTRiskAnalyzer framework to compute the likelihood and attacker cost for exploiting individual vulnerabilities on IoT devices. The authors claim to formally and quantitatively analyze these risks using probabilistic model checking. Their framework needs as input a set of software, hardware, data, and communication vulnerability scores, along with a set of candidate IoT configurations. Radomirovic [49] addresses security and privacy issues of different IoT communication protocols by taking the Dolev-Yao adversary model into account. This means the adversary is capable of blocking, eavesdropping, or injecting arbitrary messages on communication channels. The author stresses that one should consider such an adversary in the network while operating devices. Hence, he motivates scanning for rogue devices and malicious software within the network. Shodan[4] has a feature to list exploit code for services of found IoT devices. This feature is highly error prone, since Shodan cannot always determine the exact version number of services available to the network. Additionally, version numbers can be spoofed by adversaries who already took over the system. Moreover, Shodan shows only a small subset of available exploits, since it cannot detect third-party software running on the device not exposing any service via the network. Additionally, Shodan's exploit search is prone to be verbose and listing wrong results, since Shodan lists any exploit—regardless of, for example, the operating system—where the exposed service banner matches. Ali and Awad [2] describe a cyber- and physical security vulnerability assessment for IoT-based smart homes. The authors apply the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Allegro method to assess the IoT security risks. Since their work is not fully automated, they propose to develop a framework evaluating security risks for smart home IoT devices in their future work. We highlight that the authors also find a solution like SAFER important to assess IoT device risks in the future. Sachidananda et al. [50] designed a framework to perform security analysis of IoT devices. The authors performed their security analysis on 8 IoT devices. They used Nmap and PacketFence's DHCP fingerprint database to scan and identify devices. Device identification consists of the device type and the operating system. The authors show in their work an example of a device identification, which is a Google Nest IP camera detected with an operating system of the vendor Lacie building hard drives as, for example, **network attached storage (NAS)** devices. The vulnerabilities found are based on the operating system of a device detected by their fingerprinting mechanism. The

[4] http://shodan.io.

authors state in their future work to expand upon their research and conduct more complex security analysis in the future.

2.6 Gateway Approaches

We define gateway approaches as security frameworks that require a computer that is constantly on the network to passively collect and monitor all traffic information from the devices being inspected. This might be a particular performance challenge in very large and high-speed networks.

Ko and Mickens [31] introduce DeadBolt to protect IoT devices against adversaries using their DeadBolt Access Point, which restricts the communications—in- and outgoing—for IoT devices completely, if devices with TPM modules cannot prove running trusted and up-to-date software. For "lightweight" devices, like the ones we have in our organization, the authors propose to use DeadBolt to scan for malicious traffic or implement a TLS tunnel due to no possibility to directly interact with the device. The authors evaluated DeadBolt on 12 devices. Miettinen et al. [38] implement dynamic protection measures on a network gateway managed by IoT Sentinel to let compromised IoT devices coexist in the same network with not-compromised devices. The authors create fingerprints based on analyzing device connection behaviors passively to identify the device type, create a risk assessment, and restrict the network communication accordingly. The authors also perform a vulnerability assessment, which consists of querying a CVE database for the device type. IoT Sentinel targets to protect devices by blocking traffic as Feng et al. [20] do, but both solutions do not secure the device itself. We state that those approaches would not be needed if users would be aware of the security risk of their devices and would care to lower the device risks. This would involve using alternative firmware versions or devices—if available. This is the goal we want to achieve with SAFER.

2.7 IoT Nutrition Labels

This section discusses the visualization of risks, which are presented in this case by so-called nutrition labels for IoT devices. Nutrition labels, initially used in the food industry, show consumers how healthy a product is based on ingredients. Researchers [29, 30, 52] applied the idea of labeling IoT devices to show their security and privacy levels. This is a good approach to advise device owners by color indicators, as with combinations of, for example, a *red* privacy and *green* security label.

2.8 Vulnerability Prediction

We want to highlight approaches focusing on vulnerability prediction.

Chakraborty et al. [10] propose ReVeal, which performs vulnerability prediction on the Linux Debian Kernel and Chromium. The authors use both code bases, because those are well-maintained public projects with large evolutionary history including plenty of publicly available vulnerability reports. Jimenez et al. [25] analyzed the Linux Kernel with more than 570,000 commits from 2005 to 2016. The authors observed that prediction approaches based on header files and on function calls perform best for future vulnerabilities. In comparison to the approaches of Chakraborty et al. [10] and Jimenez et al. [25], SAFER does not consider the vulnerabilities for a single software with a large code base for vulnerability prediction only. Instead, SAFER retrieves vulnerabilities for different software (with a then small code base) of each device model's firmware images. SAFER consecutively performs vulnerability prediction based on all software vulnerabilities a device model is/was potentially prone to. This lets SAFER perform vulnerability predictions on the device model level. Dam et al. [15] propose a long short-term memory (LSTM) model to learn both semantic and

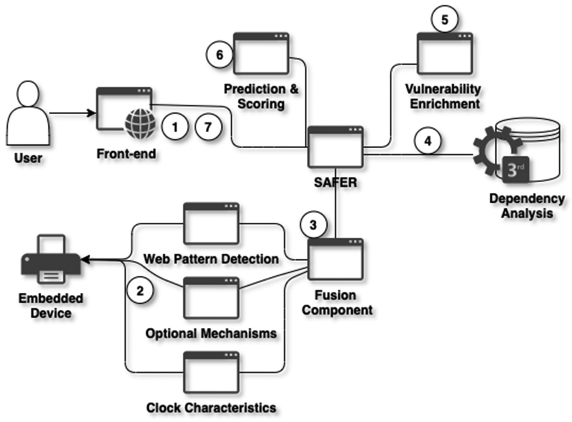

Fig. 1. SAFER consists of different components working together to identify devices, fetch their vulnerabilities and score the results.

syntactic features in source code. With this knowledge, the model predicts vulnerabilities for 18 Android applications written in Java. Garg et al. [21] propose a machine translation-based approach that learns from vulnerable source code that was replaced by fixed source code. The framework was compared with FrameVPM using state-of-the-art vulnerability prediction metrics for the products Linux Kernel, OpenSSL, and Wireshark. In contrast to Garg et al. [21], SAFER does not consider active patching of third-party source code. SAFER analyzes firmware images and identifies trends, for example, if vendors become more security aware over time by patching more quickly or if device models expose fewer severe vulnerabilities.

In the following, we introduce our framework and consecutively explain its components in full detail.

3 THE SAFER FRAMEWORK

3.1 Framework Overview

This article introduces SAFER to assess the security risks of IoT devices and to predict how the security of such devices may evolve in the future. SAFER is a holistic solution to identify IoT devices, gather vulnerability information for identified devices, score this information, and display the results to users.

Figure 1 shows an overview of SAFER and its components. Initiating a device assessment starts by using the front-end [44] (step 1). The front-end either takes the host name of the IoT device or a network prefix that will be scanned automatically. In step 2, SAFER uses multiple identification mechanisms to robustly infer the device type. In step 3, SAFER combines the results of the different identification mechanisms using a probabilistic logic framework called *subjective logic*. Identification also includes inferring the current firmware version of the device. The three most likely identification results are retained in the next steps. Step 4 will then retrieve the corresponding firmware image from the vendor's support website and determine contained third-party libraries and other information about the firmware using static analysis. Step 5 gathers publicly known vulnerabilities for the identified device model and all included third-party libraries using public vulnerability repositories. In step 6, all information is aggregated and SAFER computes a current risk metric and possible projections of that risk in the future. This enables users to understand the results of the current and future risk SAFER estimates for identified devices.

3.2 Threat Model

The main goal of SAFER is to identify device risks and advise users about them *before* attackers find and exploit vulnerabilities in their devices. SAFER is not designed to prevent or identify attacks, but to suggest preventive measures to device owners on how to reduce attack vectors and hence risks. Defending against more sophisticated attacks (i.e., Advanced Persistent Threats (APTs)) is not the primary aim of SAFER, although a reduction in attack surface is also beneficial in this context if users react on SAFER's advice to, for example, update or isolate devices.

Compromised devices could still be identified if the attack does not take specific measures to mask SAFER's identification. If adversaries want to spoof SAFER's identification mechanisms, they would need to know the potential multitude mechanisms used for identification and how they work, which is not trivial. However, to spoof device characteristics and mimic a device that is rated by SAFER as "secure" would again require sophisticated internal knowledge of how the potential multitude of SAFER's identification mechanisms classify their data. To conclude, SAFER cannot identify if a device is compromised, and SAFER is also not resilient to spoofing attacks. SAFER can only increase the difficulty of spoofing attacks by using a multitude of previously undisclosed identification mechanisms and advising device owners to apply low-risk firmware versions to their devices.

4 DEVICE MODEL IDENTIFICATION THROUGH DATA FUSION

For an effective risk prediction in later steps, SAFER requires the information about the category, manufacturer, model, and firmware versions. We experienced at CERN that a mandatory device registration and device labeling procedure for device owners has a high rate of falsely labeled devices. CERN, for example, mitigates this for workstations and servers by installing inventory applications on those devices. Installing applications is not applicable for IoT devices, and a universal IoT identification approach does not exist. Thus, we expect an even higher rate of falsely labeled IoT devices at CERN if device owners manually label their IoT devices.

Systematically extracting device information is a non-trivial task for IoT and especially challenging for those IoT devices that are part of an industrial control system (ICS) or some other cyber-physical system (CPS). For such device classes, industry-standard scanning tools such as Nessus[5] or Nmap[6] are considered to be too invasive [12] for IoT devices in comparison to their identification outcomes. SAFER employs dedicated, less invasive scanning techniques that are much closer to typical user interactions to extract information about the device. These techniques are also designed for autonomous processing, which enables SAFER to scale to large networks. Regarding our first contribution to identify devices more accurately, SAFER combines the output of multiple identification mechanisms through data fusion using the mathematical framework of *subjective logic*.

4.1 Approach

The device identification in SAFER employs two exemplary identification mechanisms (step 2 in Figure 1): **clock-characteristics detection (CCD)** and **web pattern detection (WPD)**. As their names suggest, these detection mechanisms address different aspects of the identification process for the **device under test (DUT)**. These and other non-invasive mechanisms in the literature are typically probabilistic in nature. To combine the results of our two mechanisms with subjective logic, the probabilistic results are translated into *opinions* that mathematically represent the different devices as well as the confidence in the classification. By applying suitable data fusion

[5]http://www.tenable.com.
[6]http://www.nmap.org.

operators within the subjective logic framework, SAFER can merge the results of an arbitrary set of mechanisms and can thus be extended easily with new mechanisms (this corresponds to step 3 in Figure 1). As a result, SAFER utilizes subjective logic to decide on the most likely device model being the DUT from a potentially multitude of identification mechanisms. This enables SAFER to choose the correct device model, if device identification mechanisms differ.

4.2 Subjective Logic

Subjective logic [27] is a mathematical framework used for trust management and data fusion. The key difference between subjective logic and more traditional approaches is the explicit distinction between *first-order uncertainty* (inherent uncertainty of an event) and *second-order uncertainty* (uncertainty due to errors). In the context of SAFER, the first-order uncertainty corresponds to situations where a detection mechanism cannot distinguish two devices. Second-order uncertainty, however, corresponds to a variety of factors, such as measurement error. This refers, for example, to CCD's precision value of the random forest classifier. Recent years have seen applications of subjective logic across a variety of fields [16, 35, 42] since the initial stages of development in the early 2000s. In this section, we focus on the application of subjective logic: for a brief introduction, refer to Appendix A.

In the context of SAFER, the random variable X represents a device model out of all potential device models (i.e., the domain $\mathbb{X}$ of a subjective opinion $\omega_{\mathbb{X}}^{A}$ held by actor A). The belief function $\mathbf{b}$ assigns the weight of evidence to each candidate $x \in \mathbb{X}$. In SAFER, the actor A is an evidence source (e.g., the mechanisms WPD or CCD discussed in the following); each mechanism acts as an independent evidence source. These evidence sources are then combined using the **cumulative belief fusion (CBF)** operator. To apply subjective logic for our purpose in SAFER, we have to choose a fusion operator, quantify the evidence (discussed in the following, separately for each mechanism), and appropriately interpret the resulting opinion.

In SAFER, we choose the CBF operator as the fusion operator, which closely matches our use case (see Appendix A). In addition, because SAFER allows each device identification mechanism to generate opinions over different domains, a mapping step is required to perform valid fusion. SAFER first maps all opinions to a larger domain, the smallest superset $X = \bigcup_{X_i}$, where X_i is the domain of a mechanism i. To map the belief carried in each opinion and preserve the quantification of evidence, we map the belief for an opinion $\omega_{X_i}^{i}$ to ω_{X}^{i} maps belief as follows:

$$\mathbf{b}_X^i(x) = \mathbf{b}_{X_i}^i(x) \qquad \text{if } x \in X_i, \tag{1}$$

$$\mathbf{b}_X^i(x) = 0 \qquad \text{otherwise}. \tag{2}$$

By definition, the uncertainty remains the same.

4.3 Individual Detection Mechanisms

This section introduces two examples of device identification mechanisms we designed for the SAFER prototype. Other detection mechanisms from the literature, such as the machine learning model of IoTSense [6], could be easily integrated thanks to the flexible subjective logic framework. The first mechanism is CCD, an extension of our previous work [45] that uses TCP interactions to probe the internal clock of the device. The second mechanism is WPD and based on the idea of our earlier approach [1]. We strongly enhanced our earlier approach to create WPD, which we present in this work. WPD uses a traditional scanning approach, exploiting the fact that IoT devices frequently expose a web interface.

4.3.1 CCD: Device Identification Using Clock Characteristics. The broad variety of IoT devices makes it challenging to design identification mechanisms that are universally applicable. A mechanism that uses an universal information source is the CCD approach. CCD is one of the mechanisms used in the SAFER prototype that initiates from our previous work [45]. CCD probes an internal clock of a device over the network using timestamps in TCP. The advantage of using TCP timestamps [7] is that they are easily accessible and scanning is minimally disruptive, because nearly all IoT devices implement TCP and use it for their primary functionality. The underlying assumption of CCD is that all devices of the same model share the same clock-related characteristics, which make them identifiable. In this work, we extend the existing evaluation of CCD from our previous work [45] to a larger dataset with more device models to demonstrate the scalability of the framework. We then use the results of this evaluation to explain how subjective logic opinions should be formed for use within SAFER.

Overview of CCD. CCD uses a non-invasive scanning method to identify the device model of a particular device, which is based on clock characteristics that are exposed through the TCP timestamps the device provides over time. Compared to other scanning methods with similar intrusiveness (e.g., a very low intensity Nmap scan), CCD can provide more device information [45] for trained models by exploiting information in TCP timestamps. To achieve this, a scan is performed over a much longer period of time than other methods. A single scan results in a vector of 576 TCP timestamps, collected over a period of 48 hours. These vectors are used as input for a random forest [9] classification algorithm, which is trained to detect known device models. In essence, the classifier compares the clock characteristic features of the DUT to a dataset of known device models and provides the probability of a match for each of these.

Applying CCD. CCD can be applied using a previously collected dataset of device model scans, which can be expanded with data from new device models. As the random forest classifier is used here in a supervised learning strategy, to expand the dataset, a series of initial scans of a new device model needs to be performed. However, this task can be performed as part of the standard operation. The result of the initial scans can be shared and enables the detection of the same device model (even if it is a different instance of the same device model in other networks), because the features are related to the device model rather than the environment. This process works for IoT devices that support TCP timestamps and is not limited to specific vendors or protocols. Hardware specifications of IoT devices are identical and use similar firmware. This lets clock characteristics become similar for device models. Non-IoT devices are built with varying and more complex hardware/operating systems, where TCP timestamps enable, for example, clock-based fingerprinting approaches [32].

CCD evaluation. In this article, we extend the existing evaluation of CCD in our previous work [45] to a larger dataset with more device models to demonstrate scalability. We also use this dataset to determine the appropriate quantification of evidence that enables the integration of CCD in the subjective logic based fusion used by SAFER. The fusion process itself is then separately evaluated in Section 4.4.

Dataset. We collected 5,993,052 unique scans, each containing 576 TCP timestamp samples taken over a time frame of 48 hours, as the dataset for this evaluation. The scans were performed in the CERN network, which has a wide variety of IoT and non-IoT devices brought and used by visiting researchers from all over the world. The scans were performed from several hosts throughout different subnets in the network infrastructure. The ground truth for these scans is provided by a manual identification process in SAFER's *setup phase*, which labels each scan with the associated device model. This manual identification is the ground truth for our dataset, against which we

Table 1. Model and Device Identifications Using CCD

Category	Models	Physical Hosts	Scans
CCTV	12	65	6,675
NAS	16	37	6,816
Printer	13	406	5,853,523
Projector	4	14	1,621
Routers	1	3	56
PLCs	1	11	1056
Streaming	2	26	47,561
Telepresence	4	38	58,141
Oscilloscopes	1	2	144
IP-phones	1	2	196
IP2Serial	1	9	1,684
SOC	1	4	416

evaluate SAFER. The manual process is not required for the practical application of SAFER, as a ground truth is only required for the experimental validation of SAFER. The dataset contains 752 unique IoT devices associated with 57 different device models. The dataset was collected from a network with a total of 13,181 various networked devices, which included servers, workstations, virtual machines, and embedded devices. The classification into IoT/non-IoT was done using a manual classification procedure.

Methodology. To evaluate CCD, we split the dataset into a training set representing 75% of all scans, a test set using 20%, and a validation set using 5%. In this experiment, the random forest classifier at the heart of CCD is trained with 10 iterations, where the final model is the average of these individual iterations. We evaluated in a preliminary unpublished study that the classifier performs best by using 60 decision trees. Each iteration uses a different split of training and test set to avoid overfitting to a specific training set. The evaluation of CCD, which should provide an unbiased assessment of the model's performance on data not used during any training iteration, is performed using the validation set.

Results. We used the clock characteristics identification mechanism in CERN's large-scale network to identify the IoT devices in the dataset. For the 57 device models of various categories that CCD is able to identify, we gained the following results for the training set: precision 90.81%, recall 90.19%, and accuracy 98.67%. We then used our training set to identify scans of the validation set, which our classifier was never trained with. This resulted in the following results for the validation set: precision 90.43%, recall 90.42%, and accuracy 98.64%. There are notable differences between different categories of device models, as can be seen in Table 1.

Discussion. In Table 1, we show the identified IoT devices, which are grouped for the sake of readability. We observed multiple NAS device models of QNAP and Synology that are only identifiable with an accuracy of 30% to 40%, which reduces the overall results significantly. One reason is that both manufacturers use the same hardware for different device models. CCD identifies these device similarities but is not able to distinguish these devices with high accuracy. To combine the evidence with other identification mechanisms, we convert the output into a subjective logic opinion. CCD's testing was performed by multiple hosts within different subnets of CERN's network infrastructure by initiating TCP connections in an active manner. This is less likely to be affected by intermediate network equipment, as the TCP timestamps of the DUT's clock are persisted unmodified in the packet itself and thus not significantly affected by inter-arrival times. Additionally, CCD has a sampling frequency of 5 minutes between the 576 data points, which makes it less susceptible to network interference. We stress that our scans are non-invasive, as they mimic normal user behavior.

Opinion configuration. To generate an opinion, the supporting evidence that CCD provides per device model must be quantified. The random forest classifier provides a probability of a correct match, $p(x)$. A naive belief definition would assign the probability output directly to the belief function. However, this fails to consider the uncertainty in the predictions.[7] We define the belief in a particular classification result x as

$$b(x) = \frac{p(x) \cdot D + \frac{k}{2}}{D + k \cdot n}. \tag{3}$$

This equation considers two types of uncertainty: the uncertainty due to classification errors k and the uncertainty due to the limited model size, which is defined by the number of decision trees used for training D. The intuition is that k redistributes belief mass to uncertainty, based on the positive predictive value (i.e., precision) of CCD, essentially reducing overestimated belief due to overfitting.[8] To ensure this always generates valid opinions, the division by n, the total number of identifiable device models, is needed. However, as the amount of decision trees used for training D increases, our confidence in the correct classification increases; therefore, we use this to weight the probability. The values for these parameters are based on an unpublished preliminary study. In that study, we determined the optimal performance to be at $D = 60$ with the same set of n identifiable devices, resulting in a precision value for the classifier of 0.7, such that $k = 0.7$.[9] $D = 60$ is a parameter of the random forest classifier, used both in this work and in our previous work [45]. The remaining belief mass not assigned to any x is then uncertainty:

$$u = 1 - \left(\sum_{i=1}^{n} b_i\right). \tag{4}$$

To conclude, Equation (3) describes the probability of having identified the correct device model in relation to the probability of classifying all device models. Since one could possibly overfit the classifier using many decision trees D, we introduced k as the precision value that represents how useful CCD's validation set results are in general.

4.3.2 WPD: Device Identification Using Web Pattern Detection. As a second source of information to identify the device model, and additionally the *firmware version*, we developed a mechanism focusing on IoT web pages containing the relevant information. WPD uses patterns in the user interfaces of IoT devices, which are typically web pages, to automatically identify the device model and firmware. We convert the evidence provided by this mechanism into a subjective logic opinion, which then enables us to fuse the different evidence sources in our fusion approach. In contrast to the well-known Nmap tool, WPD does not use banners of non-HTTP services for identification. This is because we found that using Nmap to identify device models based on service banners only was not able to achieve a sufficient identification ratio. The low identification ratio is caused by the fact that, within the set of devices we are interested in, the service banners contain very little information to distinguish device models. Moreover, identifying the firmware version of a device using banners also poses a challenge, since version information is often removed from banners,

[7] In subjective logic terms, this conversion would essentially state that the probability of observing a specific device model given the scan results is distributed exactly by p, with no uncertainty.
[8] This works because the precision was determined using cross validation.
[9] We chose not to adapt these values based on the new dataset to avoid a bias in the results of the evaluation of SAFER as a whole. There, our aim is to show the generalizability of the results and avoid biases for the specifics of the CERN network.

and the version of a service does not uniquely identify a firmware image. Therefore, we designed WPD as an alternative approach to address this challenge.

Overview. For IoT devices, the primary way to administrate and configure the device is often through a web interface, typically exposed through HTTP. A common way to manually gather information about the device is to analyze these administration interfaces; WPD essentially automates this process. WPD retrieves web pages and searches for patterns that reveal information about the device, such as strings in HTML pages and version numbers in embedded JavaScript snippets. WPD also uses hash values of images on these web pages, as well as **Application Programming Interfaces (APIs)** to fetch device-related information. We detected that CERN's IoT devices also serve web pages on ports other than 80 and 443, which we integrated as well. The main novelty of WPD is not the automated operation but rather the evidence quantification that allows us to fuse the evidence with other evidence. Using our large, heterogeneous network, we are also able to demonstrate that this mechanism scales after an initial pattern definition effort.

To limit intrusiveness of WPD, our patterns are designed to require few HTTP queries: a generic query that is always used and a device-specific query. The first query corresponds to a typical user interaction, such as opening a device's main page in a typical web browser. The resulting files are then analyzed using all configured patterns to gather basic information about the device; if the information retrieved is limited (e.g., if only the manufacturer can be identified), then a second query is performed. The second query accesses HTTP resources that can be used to identify a specific device model, such as a manufacturer-specific debug page that provides additional device information.

Web patterns to identify devices are set manually once per major user interface release of a firmware during SAFER's setup phase. We observed that manufacturers aim to minimize development efforts and equip several device models with identical firmware and hence similar user interface. This enables us to identify various devices using the same web patterns even for multiple firmware versions. Web patterns are created once by a single person to let SAFER (1) identify yet unknown devices and (2) major user interface updates. Pursuing a community-based approach, those patterns are propagated to a central SAFER instance enabling the entire community to identify such devices as well.

Applying WPD. WPD can be applied to a network using a pre-configured set of web patterns, which are part of the setup phase of SAFER. This pre-configured set of web patterns is based on several years of observations within the large and heterogeneous CERN network infrastructure. Many of the patterns are designed with specific device types or device models in mind; we expect that WPD is regularly improved to include additional web patterns for new devices or firmware versions to improve accuracy. In the following, we introduce the types of patterns that are currently supported by SAFER.

String patterns: WPD collects string patterns from page sources of the IoT device, which are often device manufacturer, device model, and sometimes even firmware version specific. For example, if the device is a printer, the identification mechanism detects common printer string patterns, like "Cartridge" or "Tray" occurring in well-defined positions in the page source. Naively searching for web patterns in the page source without considering the context can cause false positives, because, for example, a random number may be erroneously be identified as a firmware version. We describe a web page's context using XPATH, which narrows the potential page content where WPD can apply its web patterns to.

Embedded libraries: IoT manufacturers embed a variety of custom-built or public third-party libraries, such as JavaScript or CSS libraries, into their device's web pages. WPD recognizes the occurrence of model-specific libraries, for example, and uses this as additional evidence for device identification.

Hashes of images: Some manufacturers embed descriptive text into images. For example, Sony's network camera includes the text "Ipela network camera SNC-RZ50P" inside an image on the device web page. Hence, WPD can detect the device model via comparing hashes of such images.

APIs: Some manufacturers, such as QNAP,[10] provide applications to administer their devices, which can scan the network for their devices. We analyzed the network traffic of such an application for QNAP devices, for example, where we detected always unprotected and generic REST-API URL. WPD mimics such a scan using the same REST-API endpoints as the administrative applications to read out the model and firmware version directly.

WPD evaluation. To evaluate the performance of WPD, we scanned the CERN network, which at the time contained 13,181 networked devices, out of which 687 are classified as IoT devices. This ground truth is derived from a manual classification process to verify if WPD identified all IoT devices, as well as WPD's associated IoT device models and firmware versions. However, this manual process is not required for the practical application of SAFER. Within this dataset, WPD identified 526 out of 687 IoT devices, which is an identification rate of 76.56%. We manually verified the correctness of this classification and found that there were no false positives. The detailed results of the detected devices can be found in Table 2, which lists the identified devices per category, models, and firmware version found. WPD identified that the 526 detected IoT devices can be categorized into 50 manufacturers, 110 models, and 124 firmware versions of home, business, and laboratory IoT devices. For the 161 IoT devices that were not correctly identified, a partial classification was possible for a further 110 devices (e.g., only category, manufacturer, or model). The remaining 51 IoT devices were fully protected by an authentication prompt.

These results assume no knowledge of credentials for the landing pages of IoT devices. However, if the user specifies the credentials, WPD is able to analyze password-protected devices.

Opinion configuration. The WPD opinion is set by matching previously specified web patterns to a device web page. This identifies—in the best case—the device manufacturer, model, and firmware version. Increasing the amount of web pattern matches increases the identification certainty of the potential device model. If WPD identified a firmware version, detected image hashes, or identified configuration pages, we increased the certainty even more. This gathered certainty affirms the correctness of the identified device information of WPD. To configure a subjective opinion, it is relevant to have device model knowledge and, moreover, identification mechanism knowledge. This is important, because we consider the most describing web patterns of a device model only. To explain the background knowledge, which results in the belief Equation (6), we split the calculation into different parts.

The first part of Equation (6) is defined within the brackets stating $amount\_text\_patterns$ and $amount\_hashes$. Based on our observations, CERN's assessed devices serve either a web page containing device describing text or pages mainly containing an image that embeds device information. We identified image-embedded device information for seven device models of seven different categories. Since WPD only matches few text patterns in such an "image-only" case, we set the weight of $amount\_hashes$ to 0.3. This resulted in one hash having similar weight than two matching text pattern, because in an "image-only" case, we identified that the image contains the vendor and model of the device. To compare this, identifying a vendor and model with text patterns would result in two used text patterns. We argue that hashes are rarely used and are—due to their weight—limited in occurrence.

[10]https://www.qnap.com/en/utilities/essentials.

Table 2. Diversity of Detected Device Types Using WPD

Categories	Physical Devices	Manufacturers	Models	Firmware Versions
CCTV	39	6	19	10
KVM-Switch	5	1	1	0
Infoscreen	1	1	1	0
IP-phone	6	5	3	3
IP-to-Serial	18	3	3	2
IP-to-USB	2	1	2	3
IPMI	67	3	6	9
NAS	32	4	5	4
Oscilloscope	16	4	13	18
PLC	71	3	13	15
Printer	97	9	30	47
Projector	23	1	3	0
Router	31	3	3	0
SoC	9	1	1	0
Switch	2	1	2	3
Telepresence	101	3	4	10
Thermometer	6	1	1	0

The second part of Equation (6) relates to $amount\_config\_patterns$. First, we add $\frac{1}{6}$ per matched $amount\_config\_patterns$ analog to matched $amount\_text\_pattern$, the reason being that we only consider up to six web patterns (text and config) to calculate the belief value, which we discuss in the later part of this section. We argue that if WPD matched a web pattern on a configuration page containing detailed technical or administrative information of a device, we consider it more likely that this information is correct. To express this increased belief in the Equation (6), we added the factor 0.1 to $amount\_config\_patterns$.

The last part of Equation (6) refers to $fw\_detected$. This value is set to 0.4 as shown in Equation (5), if WPD found the firmware version that occurs in well-defined places within a device's web page. WPD outputs only the most characteristic patterns it matched for a device model, its vendor, and the firmware. In our experiment, most commonly two text patterns matched the device vendor and model, and one config pattern matched the firmware version. This lets the first part of Equation (6) be $\frac{2}{6}$, the second part to be $\frac{1}{6} + 0.1$, and the last part to be 0.4. This results in a high belief of 1.0, meaning that the device was identified by WPD.

$$fw\_detected = \begin{cases} 0.0, & \text{not detected} \\ 0.4, & \text{detected} \end{cases} \tag{5}$$

$$\begin{aligned} b(x) = &\left(amount\_text\_patterns \cdot \frac{1}{6} + amount\_hashes \cdot 0.3\right) \\ + &\left(amount\_config\_patterns \cdot \frac{1}{6} + amount\_config\_patterns \cdot 0.1\right) + fw\_detected \end{aligned} \tag{6}$$

$$b(x) = \min\{b(x), 1.0\}, \tag{7}$$

where *amount_hashes* is the number of recognized images (using known hashes), *amount_text_patterns* is the number of matches from text-based web patterns, *amount_config_patterns* is the number of matches from configuration page web patterns, and *fw_detected* represents if WPD detected the firmware version.

The maximum amount of text patterns ($amount\_text\_patterns$ and $amount\_config\_patterns$) for WPD is set to 6. We limit matching text patterns and hashes because WPD is designed to use only the most describing ones per device model. This prevents WPD from including, for example, unrelated patterns of other categories or vendors that would falsify the belief of a device identification.

4.4 Evaluation of Device Identification through Fusion

After all device identification mechanisms have generated a subjective logic opinion for the DUT, SAFER takes all these results and fuses them using the subjective logic operator CBF. In this section, we evaluate the fusion performance of SAFER using the two detection mechanisms discussed in the previous section: CCD and WPD. As discussed in Section 4.2, the fused subjective logic opinion assigns a certainty to each device model, resulting in a ranking of the most likely device model candidates based on the observations. In SAFER, we decided to propagate the three most likely device model candidates to SAFER's later steps. However, in this third experiment, the aim is to analyze the quality of the fusion process of SAFER for the purpose of device model identification specifically, and thus the experiment requires a single output. We use the most likely device model as this single output.

During our experiment, we conducted 4,988 device identifications on known device models with our identification mechanisms. This contains 2,494 identifications of CCD and WPD resulting in 2,494 fusion operations. We repeated device identifications at least three times at different points in time to verify reproducibility of the fusion results. Grouping those to unique entities for identified IoT devices results in 572 devices. A fusion results—in our current setup—in a list of 57 or 58 possible device models likely to be the DUT. The reason is that CCD outputs probabilities of all 57 trained IoT models and WPD outputs the model with the highest probability. This results in a result set of either 57, if WPD identified the device model that is in the same set of CCD, or 58, if the result sets are disjoint.

For 360 out of the 2,494 fusion operations, neither CCD nor WPD identified a device model. Since the correct device model is not in the fusion domain, we omit these because the fusion component cannot clearly decide on the correct device model. Out of the remaining 2,134 operations where device models were identified, the fusion component chooses the correct device model for 1,975 operations, yielding an identification rate of 92.55%. Within the correctly identified devices, we find that the confidence with which the correct device model is chosen is high. This is reflected by the median certainty of the opinion (i.e., $b(x_{correct})$) for correct device models is 67.83%. Moreover, the difference between the device model with the highest certainty x_1 and the device model with the second-highest certainty x_2 (i.e., $b(x_1)-b(x_2)$) is an average of 0.5454. Moreover, one can find in 96.34% of the 2,134 fusion operations the correct device model within the three most likely ranked devices. We find that WPD identified the correct device—of the 2,494 fusion operations—for 77.09% and CCD for 21.90%. For all missed device models of WPD, it still identifies the correct category and manufacturer for 29.80%. The reason CCD identified fewer devices correctly is that the fusion dataset contains a large fraction of device models on which CCD was not yet trained on. CCD was still able to identify the correct device category if it missed a device model with 9.47% and the manufacturer with 11.85%.

In our experiment, the overall results of the device fusion component provide a confident decision in the identification of the correct device. In particular, the median certainty of the 1,995 correct classifications is 67.83% for the correct device model and the median uncertainty is < 0.1. The difference in certainty between the most likely classification and that of the next most likely classification is, on average, 54.54%. Thus, in addition to being highly certain, the identification is

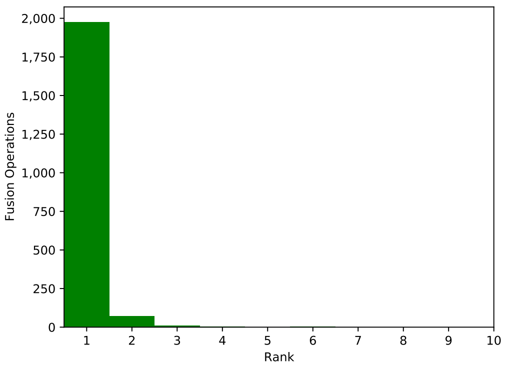

Fig. 2. Rank of Correct Identified Devices in Fusion Operations.

confident of a single classification. By using the certainty of the devices as a ranking mechanism and determining the rank of the correct classification from the ground truth, we obtain Figure 2.[11] This figure clearly shows that even if the classification is not correct, the correct classification is among the highest certainty options.

Using the data fusion approach subjective logic to fuse multiple device identification mechanisms, we achieve a correct identification ratio of 92.55% for 2,134 fusion operations.[12]

For our physical devices, we identify 531 out of 572 devices leading to a device identification rate of 92.83%. We recall that WPD achieved on this fusion dataset an identification rate of 77.09% and CCD 21.90%. Thus, the fusion components' identification rate of 92.55% is only achievable using the device fusion approach of subjective logic and not by using the identification mechanisms independently. Compared to related work, using identification mechanisms independently, we apply a data fusion method to enable the use of multiple device identification mechanisms under uncertainty. This fusion component of SAFER makes it unique to previous device identification works.

If both identification mechanisms have the same result set, the fusion component adds more certainty to the device model in common by using the CBF operator. If an identification mechanism is not able to detect an IoT device, it can—due to the benefits of subjective logic—include degrees of uncertainty and vagueness indicating "I don't know." SAFER displays the device identification certainty value to users, making them aware of such uncertain results.

One point for potential future improvements to the identification performance is to incorporate a non-homogeneous base rate. In the current prototype, SAFER uses a homogeneous base rate for all device models, essentially assuming each device is equally likely in the absence of evidence. If, for example, certain device models are no longer in production, one can express this by providing a lower base rate for these device models and dividing the probability mass among the other candidates.

5 RISK METRICS

Due to a high device variety in large networks, retrieving firmware images and looking up vulnerabilities in a manual fashion is a time-consuming task. However, we consider those steps essential

[11]For legibility, the figure shows only a subset of 10 instead of 57 or 58 possible ranks, as the rest has a negligible contribution.

[12]We recall that 360 out of the initial 2,494 fusion operations are omitted because neither CCD nor WPD identified a device model.

for comprehensive risk assessments. We show in the following how SAFER automates those steps as it (1) obtains IoT firmware images from vendor websites, (2) analyzes these to extract vulnerable material, and (3) deduces vulnerabilities of the IoT device model, including those resulting from the firmware-contained software. Based on this information, SAFER later computes current and future device risks based on the metrics we introduce in this article.

5.1 Vulnerability Analysis

Assessing the security risk of an IoT device's firmware image includes manual effort, such as gathering vulnerability exposing material, and is often error prone. Besides publicly known and documented vulnerabilities for a specific device, the device's firmware may also include third-party libraries that introduce public vulnerabilities the IoT device's vendor never documented as relevant to its devices. In its step 4 (Figure 1), SAFER conducts an *automated, static analysis of the firmware images* to find such third-party libraries that will later also be considered in the vulnerability retrieval in step 5. This automated analysis is the main part of our third contribution.

5.1.1 Firmware Retrieval. Based on the device model and firmware version derived from SAFER's previous steps, we now retrieve the firmware for analysis. The first challenge is the lack of a central publicly available data source to find firmware images or information about software dependencies of IoT devices. We approached this problem by (1) creating this data source with SAFER, which executes periodic scripts to gather not yet processed firmware images from 20 official manufacturer websites and analyzes them. We made this information public on SAFER's website.[13] This enables SAFER, as well as other researchers, to automatically retrieve firmware images and detect different dependencies, such as third-party libraries. o far, SAFER has populated the firmware component with more than 825 firmware images of various device models. SAFER periodically runs background tasks to check for new firmware images that it consecutively downloads and analyzes. For increased reliability, we focus on official manufacturer websites for firmware retrieval opposed to scraping this from search engines on the Internet. Based on release notes, for example, the automated mapping between device model and associated firmware also becomes more reliable.

5.1.2 Firmware Unpacking. To automatically unpack and analyze firmware images, SAFER utilizes the **Firmware Analysis and Comparison Tool (FACT)**.[14] SAFER includes a variety of improvements to FACT for the IoT context like additional third-party library detection patterns and unpacking mechanisms. SAFER's firmware component takes the device fusion component result as input. It then fetches vulnerabilities of the three most probable device models and firmware version provided by step 3. To allow SAFER the downloading of firmware images, we manually specified the start of vendor support pages in SAFER's initial setup phase. By crawling official IoT vendor support websites, *SAFER automatically fetched a total of 825 firmware images* and uploaded them for static analysis to the firmware analysis component. The average firmware size is 38.18 MiB with an average amount of 218,610 unique files included. The firmware unpacking component unpacked 160.10 GiB of unique files with an average size of 767.91 KiB.

We show the release date distribution of all downloaded firmware images we analyzed in September 2019 in Figure 3. We later use the information about firmware release dates to calculate metrics on how frequent a vendor updates their device models. We introduce the approach analyzing firmware images in the following.

[13] https://safer.network.
[14] https://github.com/fkie-cad/FACT_core/.

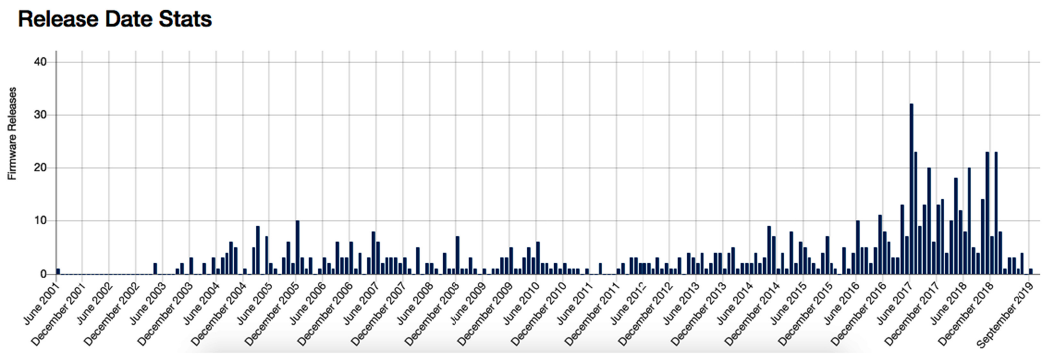

Fig. 3. Release date distribution over all firmware images in the firmware unpacking component.

5.1.3 Firmware Analysis. Once downloaded, actual analysis of firmware images is a non-trivial task, as manufacturers use, for example, various file types, file systems like squashfs, languages like printer job language, or packing mechanisms like zip for a single firmware image. SAFER's firmware component provides extensive functionality in this regard and also provides support for static analysis that is faster, less error prone, more scalable for large datasets, and less resource consuming during the analysis in comparison to dynamic analysis. During our work with SAFER's firmware component, we detected that QNAP[15] encrypts their firmware images, which prevents SAFER from unpacking it. To tackle this, we extended the component with an unpacking plugin for QNAP firmware images using the Data Encryption Standard (DES)[16] along with the publicly known QNAP firmware password. This enables SAFER to decrypt QNAP firmware images and correctly identify the containing libraries. It can be expected that more such adaptation work will become necessary as SAFER learns to analyze more and more different devices that are not yet supported by the component.

To identify software library- and additional vulnerability patterns, the SAFER component applies 60 **"Yet Another Recursive Acronym" (YARA)** rules on unpacked files. YARA has its origin in malware research to describe patterns and is based on regular expressions. Each rule represents a set of strings and one Boolean expression determining the rule logic. Matching patterns in text and binary files along with combining them with Boolean expressions is how we find libraries in firmware images. Since there is no public database to find a sufficient amount of YARA rules for third-party software, we manually investigated software contained in firmware binaries and added new rules to SAFER's firmware component. As this increased the detection rate of software libraries, we contribute those rules back to the community on SAFER's website.[17]

In SAFER's next step 5, SAFER aims to *identify publicly known vulnerabilities* the manufacturer or security researchers filed for the identified device model. as well as the vulnerabilities known for the firmware-contained software.

5.1.4 Software Library Vulnerabilities. Now, SAFER needs to fetch publicly known vulnerabilities for detected libraries. Those vulnerabilities are required to build the base for later security metrics. SAFER utilizes an existing solution called *CVE-Search*[18] to automatically retrieve publicly known vulnerabilities of all libraries and, later on, device model vulnerabilities. SAFER considers

[15]https://www.qnap.com.
[16]https://csrc.nist.gov/publications/detail/fips/46/3/archive/1999-10-25.
[17]https://safer.network.
[18]https://github.com/cve-search/cve-search.

Table 3. Third-Party Software Distribution over All Firmware Images

Software	Usage
OpenSSL	57.3%
OpenSSH	24.1%
Linux Kernel	5.6%
jQuery	4.3%
curl	3.0%
BusyBox	2.9%
udhcp	2.0%
wpa_supplicant	0.8%
SSLeasy	0.3%
Dropbear	0.2%
Others	1.5%

all vulnerabilities registered for identified firmware- and software versions as relevant. CVE-Search provides a local copy of aggregated vulnerability database that retrieves CVE information from a variety of external data sources such as NVD from the National Institute of Standards and Technology (NIST).[19] CVE-Search also enriches retrieved vulnerabilities with helpful information for security experts such as Metasploit[20] plugins. We update SAFER's local vulnerability database every hour to fetch newly registered vulnerabilities. The distribution of included software over all firmware images is listed in Table 3. SAFER identified that more than half of the firmware images include a version of the OpenSSL library. A well-known weakness of this software is the so-called Heartbleed vulnerability, for which SAFER identified 19 prone firmware images.

Device model vulnerabilities. In the vulnerability retrieval process, we distinguish between (1) retrieving vulnerabilities for firmware-contained software of third parties and (2) device model vulnerabilities for software the vendor developed. Analyzing vulnerabilities related to the device model is becoming increasingly important, as recent articles about vulnerable models[21] show. These vulnerabilities are publicly registered at MITRE,[22] which acts as a primary CVE numbering authority. SAFER retrieves all publicly known vulnerabilities for a device model from its local vulnerability database copy through a series of Common Platform Enumeration (CPE) queries. SAFER queries both the device model and firmware version, as well as the third-party libraries detected using the YARA rules from the previous section.

Identifying additional vulnerabilities. As a second step in the vulnerability retrieval process, SAFER focuses on unpacked firmware images. Hard-coded passwords or sensitive cryptographic material shipped in firmware images are another vulnerability source. SAFER's firmware component has a (binary) analysis plugin using regular expressions to detect hard-coded passwords or cryptographic material such as private/public keys and certificates. SAFER identified firmware-contained certificates one uses to perform certificate pinning. SAFER's plugin also detected Secure Shell (SSH) keys developers may used to connect to IoT devices for the sake of testing. Attackers

[19]https://www.nist.gov.
[20]https://www.metasploit.com.
[21]For example, https://www.helpnetsecurity.com/2019/09/17/vulnerabilities-iot-devices/ or https://www.infosecurity-magazine.com/infosec/iot-four-year-router-flaw-1/.
[22]https://cve.mitre.org.

could download the firmware and extract passwords or other sensitive cryptographic material, which are applicable to at least all identical device models. Such secrets could, for example, enable attackers to execute Man-in-the-Middle attacks or connect to the IoT device using a privileged account. SAFER does not address this risk in its device risk score calculation. However, to make users of SAFER aware that attackers can execute such attacks, we indicate on the user interface of assessed IoT devices if, for example, SSH private keys were found. SAFER's user interface suggests call-for-actions to non-technical users based on the found information of this step and the associated risk of the device.

In all unpacked firmware images, SAFER's firmware analysis component identified 408 SSL certificates and 98 private and 32 public keys used for SSL, SSH, or PGP. We assume that public keys and SSL certificates positively impact the security level of a device, where private keys and 168 unique username and password occurrences negatively impact the security level.

As a result, this SAFER component fetches all device model related firmware images and identified publicly known vulnerabilities of the three device models propagated by the device fusion component. The vulnerabilities are based on software libraries, device model, and additional vulnerabilities consisting of key material. With this component, SAFER executed all preliminary steps before triggering the scoring component to calculate different device risk scores per device model.

5.2 Risk Metric Overview

In step 6, SAFER assesses the risk of a device based on found security vulnerabilities in two ways: first, an immediate CDSRI is calculated to signal to users the level of risk that is currently associated with operating this device in the network. SAFER additionally aims to provide an indication on the security problems that may arise from the device in the future. To this end, it investigates historic information on how frequently vulnerabilities for this device model have shown up in the past and how timely they have been addressed. Extrapolating this past behavior into the future leads to a future risk prediction in form of a future VT and a corresponding future PT. The VT represents the median risk associated with a device model, based on all past and future security vulnerabilities. The PT measures how manufacturers have handled patching of past security vulnerabilities and extrapolates from this the likely future availability of patches. SAFER then combines the PT and VT to predict the FDSRI of the identified device. The following sections will address how those metrics for our second contribution are calculated in detail.

5.3 Current Device Security Risk Indicator

To display a single risk indicator representing the current security risk of the IoT device, SAFER needs to consolidate all vulnerabilities found in the previous step. In our previous work [44], we experienced the need of simple comparison approaches like a single risk indicator visualized as a traffic light and, if requested by users, the availability of more detailed information. We argue that risk indicators such as "low," "medium," or "high" help to assess the current risk of one's device at a glance, which eases device comparisons. To calculate such a single risk indicator, SAFER uses CVSS 3.0. SAFER takes all CVSS values of unpatched vulnerabilities into account, which were found in previous steps for the identified device model. SAFER classifies the unpatched vulnerabilities by Equation (8). The CDSRI is based on the highest CVSS value max_{cvss} of unpatched vulnerabilities. We argue that attackers will likely focus on most severe and unpatched vulnerabilities to limit the time and complexity to compromise devices in the network.

SAFER distinguishes its users into two types [44] that require different risk presentations of, for example, the CDSRI. For SAFER's technical user interface, the CDSRI is displayed as a main component of a devices' risk assessment. We received feedback [44] from technical users requesting additional information as, for example, a histogram showing the grouped CVSS severity levels of unpatched vulnerabilities. We added this and more data we used to calculate the CDSRI to help users understand at a glance how many and how severe vulnerabilities were found for the current firmware version. This also includes the age of vulnerabilities, found private keys, and available exploits displayed as additional information on SAFER's technical user interface.

For non-technical users, SAFER shows a traffic light [44] that indicates the CDSRI. It shows green for none or low, yellow for medium, and red for a high or critical device risk. An advantage of SAFER is to help users with securing devices, because it, for example, advises users to update/downgrade their device to a more "secure" firmware version by call-for-actions for non-technical users. SAFER supports such a call-for-update feature because it assesses the CDSRI for every analyzed firmware version of identified device models. We introduced this feature to SAFER because we identified that the most recent firmware version is not always the most secure version. We consider SAFER's firmware version suggestion a beneficial feature to raise awareness and, as a result, reduce the risk level of devices by informed users.

$$\text{CDSRI} = \begin{cases} \text{None} & max_{cvss} = 0.0 \\ \text{Low} & 0.1 \leq max_{cvss} \leq 3.9 \\ \text{Medium} & 4.0 \leq max_{cvss} \leq 6.9 \\ \text{High} & 7.0 \leq max_{cvss} \leq 8.9 \\ \text{Critical} & 9.0 \leq max_{cvss} \leq 10.0 \end{cases} \tag{8}$$

Besides calculating the risk for the currently installed firmware version with the CDSRI, SAFER also predicts how a device's risk will evolve in the future. We consider it an important aspect to be aware of currently "secure" devices that potentially evolve into high risk-exposing devices in the future or vice versa. In the following, we discuss how SAFER predicts those future risk indicators.

5.4 Future Risk Prediction

SAFER's VT and PT extrapolate past observations into a novel prediction of future security risks a device may pose due to yet unknown vulnerabilities. To calculate future trends for IoT devices, SAFER tries to predict the severity level of potential future vulnerabilities and patch times of vulnerability patches published by vendors. To predict future severity levels for the VT, SAFER takes all previously retrieved vulnerabilities into account.

Future patch times for the PT are predicted based on time intervals of patched vulnerabilities—from the date they became publicly known until the vendor fixed them. We use past patching behavior in Section 5.4.2 as a base to predict the future patch times. One can recognize those patch times in Figures 4 and 5 of the appendix. For vulnerability and patch prediction, SAFER applies and compares different models including Facebook's Prophet [56], simple moving average, and different **Auto Regressive Integrated Moving Average (ARIMA)** models[23] based on the Box-Jenkins method [8] on its vulnerability severity and patch data.

We define all observed values for either VT (containing severity levels) or PT (containing patch times) as the dataset O. This dataset is split into a training set O_{train} containing 66% and a test set O_{test} containing 34%. We then train the prediction models on O_{train} and predict the potential future values $\hat{Y}$ using these models. We then compare the real values O_{test} with our predictions $\hat{Y}$ using the **mean absolute deviation (MAD)**, which is a measure for statistical dispersion and robust to outliers.

[23]https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html.

We define MAD as follows. Given the set of outcomes O_{test} and the set of predictions $\hat{Y}$, the absolute deviation x_i between the prediction $\hat{y}_i \in \hat{Y}$ and the outcome $o_i \in O_{test}$ is defined as $x_i = |o_i - \hat{y}_i|$. The set of differences $X = \bigcup\{x_i\}$ and the associated median $\tilde{X} = \text{median}(X)$ are then used to define MAD as the absolute difference between this median and the individual deviations—that is,

$$MAD = \text{median}(|x_i - \tilde{X}|), x_i \in X. \tag{9}$$

5.4.1 Vulnerability Trend. To determine the likely severity levels of an IoT device's future vulnerabilities, we calculate the VT.

The VT is based on the potential occurrence and severity of future vulnerabilities. We applied the three prediction models on the VT data and later show that the ARIMA model predicts the VT best. ARIMA is a class of statistical models for analyzing and forecasting time series data. Those three models divide into the "AR" part, the "I" part, and the "MA" part. The autoregression model (the "AR" part of ARIMA) uses the dependent relationship between an observation and a number of consecutive (lagged) observations. The integrated model (the "I" part of ARIMA) subtracts observations from previous observations to make the time series stationary. The moving average model (the "MA" part of ARIMA) uses the dependency between an observation and a residual error of lagged observations.

ARIMA predicts future vulnerabilities v_{future}, which are based on CVSS values of vulnerabilities from the past (v_{past}). We combine both sets—containing past and future vulnerabilities—to calculate the VT per device model, which results in $X^{vulnerabilities}$ as shown in Equation (10).

$$X^{vulnerabilities} = v_{past} \cup v_{future} \tag{10}$$

$$vt_{cvss} = \text{median}(X^{vulnerabilities}) \tag{11}$$

$$vt = \begin{cases} \text{Low} & 0.0 \leq vt_{cvss} \leq 3.9 \\ \text{Medium} & 4.0 \leq vt_{cvss} \leq 6.9 \\ \text{High} & 7.0 \leq vt_{cvss} \leq 10.0 \end{cases} \tag{12}$$

CVSS values of vulnerabilities indicate the severity of every vulnerability in a range from 0.0 to 10.0. Using the median of all vulnerabilities a device has experienced and may experience based on ARIMA predictions provides us with vt_{cvss} as shown in Equation (10). This metric is a robust estimate of the average severity of a device's vulnerabilities.

To classify the VT value in a more human understandable risk format, we use the thresholds of CVSS version 2.0 as shown in Equation (12) for our final *vulnerability trend* vt. We use CVSS version 2.0 instead of a more recent version 3.x because we identified in prior work [44] that SAFER's users require few and clearly distinguishable categories as, for example, for the VT. CVSS version 2.0 enables us to use "low," "medium," and "high" as VT categories instead of using CVSS version 3.x that implements two additional categories.

We used *pmdarima*,[24] which allowed us to analyze the set of CVSS values per device model and identify the best-performing ARIMA parameters automatically to predict future values. We then trained the prediction models using the best ARIMA parameters on the training set of 38 device models and forecast the potential future severity levels $\hat{Y}$. Afterward, we calculated the VT category for the predicted data and compared it with the VT category for the test set data containing real observations. The best prediction model, which we use in the following to predict

[24] https://pypi.org/project/pmdarima/.

$\hat{Y}$ and the VT category, is the ARIMA model with 100% accuracy. The second best prediction model is the simple moving average model with 94.74%.

Predicting the severity level for future vulnerabilities, we achieve better accuracy with ARIMA (1.31 CVSS MAD, $\tilde{X}_{all\_models}$: 0.03 to 0.40 CVSS) than with a simple moving average (1.36 CVSS MAD, $\tilde{X}_{all\_models}$: 0.17 to 0.41 CVSS) or Prophet (2.39 CVSS MAD, $\tilde{X}_{all\_models}$: 0.37 to 1.11 CVSS). The median time frame for all 38 device models that we predicted on was 219 months. This is the time from the first to the most recently registered vulnerability. Those 219 months contain on average a total of 160 CVEs per device model. Thus, the average training set size of 144 months contains 105 CVEs, and the ARIMA model predicts on average 55 CVEs for 75 months of the test set.

Since the VT data has a range from 0.0 to 10.0, it has less numeric variation compared to PT's patch intervals ranging from a few days to multiple years. Thus, we argue that using an ARIMA model—being a linear model with less variation—to predict CVSS values of future vulnerabilities achieves better results. Figure 6 of the appendix shows an ARIMA VT prediction example, and Figure 7 shows a Prophet VT prediction for the same data. Based on those figures and the preceding prediction errors, one can see that ARIMA prediction is more accurate to predict the VT than the one of Prophet, which attests to our preceding argumentation. To conclude, achieving an accuracy rate of 100% using ARIMA models to identify the VT category of 38 device models is a robust indicator of the risk that devices expose now and potentially in the future. We introduce the PT in the following as the last requirement for SAFER to estimate a device's future security risk indicator.

5.4.2 Patch Trend. Next, the PT identifies how long devices on average are left vulnerable to security vulnerabilities in relation to the time needed by potential attackers of different skill levels to exploit such a vulnerability.

First, the PT determines on average how long a manufacturer took to patch vulnerabilities in their firmware images once they became publicly known. For every detected vulnerability, SAFER measures the time from being publicly registered in a vulnerability database until the manufacturer patches the vulnerable software. SAFER determines those time spans by utilizing two approaches that we introduce later in this section.

Second, SAFER utilizes the patch intervals from the past to predict future time spans the vendor might need to patch new vulnerabilities. We again evaluate all prediction approaches to forecast future time spans. Last, the PT classifies past and future patch intervals into three categories (fast, medium, slow) based on the time needed for different adversary types to exploit vulnerabilities. The first step for the PT is to identify past patch intervals. We show in the following from which sources SAFER determines if a vendor patched a vulnerability.

Patch intervals using release notes. If IoT manufacturers add CVE identifiers of fixed vulnerabilities in their firmware release notes, SAFER can calculate the PT precisely, as CVEs also always include a date of creation. Therefore, SAFER obtains the creation date from all CVE identifiers of vulnerabilities contained in firmware images and also of all included libraries. It then matches this information with the release notes of the IoT manufacturer to detect if and when which vulnerability got patched. Performing such an analysis, one can calculate an average time to patch for a device model, and one can even generate a trend for how this duration changes over time. A patch is considered if (1) the vendor introduced a software version not prone to CVEs of a prior software version or (2) by removing the entire software.

Patch intervals using library version changes. Since not all IoT manufacturers carefully document patched CVEs in release notes, SAFER also implements an alternative patch interval calculation.

For this, SAFER obtains all CVEs for every available firmware version of a device model including all contained libraries. For every CVE, SAFER analyzes consecutive firmware images for version changes in the vulnerable library. SAFER considers two cases as a heuristic to define the firmware release date as the patch date of a vulnerability. First, if the library is not included in the firmware anymore, SAFER considers all CVEs for this software as patched because the attack vector is removed. Second, if a future firmware image contains another library version. In this case, SAFER determines if the library version found in a consecutive firmware image is not marked as vulnerable by the investigated CVE. We identified such library version changes in many firmware images of, for example, Hewlett-Packard Lights-Out-Management models or Mobotix CCTV cameras.

For both alternative PT calculations, vulnerability patch times for all library vulnerabilities and model vulnerabilities result in patched and unpatched vulnerabilities, which SAFER takes as metrics to calculate the model's PT.

To validate the performance of our patch time predictions, we trained all prediction models on the patch intervals of the training set. The patch intervals are defined by the passed days from when a vulnerability became publicly known until the vendor released a patch for it. We recall that the predictions are based on an average of 219 observed months per device model. This interval starts with the first registered vulnerability and ends with the last entry of the training set. We identified that vendors on average require 94 days to patch a vulnerability. Vendors patch within the training set less often (107 days) than within the observed test set (66 days).

This is a fact that other researchers [4, 51] also detected. They identified that vulnerability disclosure shortens the time for software vendors to release patches. The authors also point out that vulnerability researchers changed from reporting vulnerabilities publicly to inform vendors about found vulnerabilities. This enables vendors to provide patches before a vulnerability goes public. The authors state that since 2008, vendors have provided patches for more than 80% of total vulnerabilities until their disclosure dates [51]. We believe that this is a reason we identified shorter patching intervals in more recent patch data of SAFER.

For predicting vulnerabilities, Johnson et al. [26] point out that historical data allowed forecasting with reasonable accuracy for products such as Linux, Microsoft Office, Google Chrome, Oracle Java, and PHP, among others, meaning that there is a relationship between past and future vulnerability data. More importantly, recent work states that linear models like "ARIMA can be recommended for predicting IT security vulnerabilities, as they consistently achieve low forecasting errors" [64]. In general, related work considers linear models to predict future metrics. Thus, we refer to related work and assume that past (vulnerability/patch) behavior correlates in a similar way to future behavior, which leads us to implement this in SAFER's predictions.

The trend of vendors patching more often over time (or vice versa) is an important aspect that prediction models have to include in their predictions. We stress that in our experiment, Prophet adapted to such changes best. The best prediction model for PT, Prophet, shows a prediction error of 14.31 days MAD compared to the test set. The median over all errors $\tilde{X}$ for Prophet's device model predictions varies between 0.02 and 15.19 days. We compare our results with the second best prediction model ARIMA with a MAD of 24.12 days. $\tilde{X}$ for all 38 device models ranges from 2.78 to 44.12 days. Predicting patch intervals is not trivial, as the data has a high amount of variation and ranges from a few days up to years. One can see the high variation in Figure 4 of the appendix, which shows an ARIMA PT prediction example. We compare it with a Prophet prediction on the same dataset in Figure 5. We argue that Prophet has a better adaption of the high variation in the data than ARIMA or simple linear prediction models.

After prediction, SAFER holds the two sets of past and future patch times required to calculate the PT. In the following, we introduce the background information to classify the PT.

Classifying the PT. We assume that most attackers learn about new vulnerabilities by subscribing to publicly available vulnerability sources[25] and that attackers require a varying amount of time to develop exploit code. This implies that the "zero-day time" (i.e., the time during which the attack is not officially known and no CVE is registered) is not covered by our analysis. We classify attackers into different skill levels, namely security professionals (c1), advanced security experts (c2), and script kiddies (c3). The time needed for experienced security professionals (c1) to develop exploit code for a given vulnerability takes a median of 22 days according to a study by the RAND Corporation [13]. We also investigated on a publicly available repository[26] listing exploit code as well as exploit plugins for Metasploit to measure the time when a vulnerability was publicly registered until the exploit code was uploaded to this repository. We assume that script kiddies (c3) do not have further security knowledge and can execute these exploit plugins only. We identified that it takes a median of 414 days after the vulnerability was publicly announced until an exploit code becomes available on this repository.

Hence, we adopt these reference values, including the interval in between representing advanced security experts (c2), as thresholds to generate the SAFER PT in Equation (15).

$$X^{patch\_time\_spans} = pts_{past} \cup pts_{future} \tag{13}$$

$$pt_{median} = \text{median}\{X^{patch\_time\_spans}\} \tag{14}$$

$$patch\_trend = \begin{cases} \text{Fast} & 0 < pt_{median} \leq c1 \\ \text{Medium} & c1 < pt_{median} \leq c3 \\ \text{Slow} & pt_{median} \geq c3 \end{cases} \tag{15}$$

pts_{past} of Equation (13) represents past patching behavior for a device model for which SAFER can use two approaches. The first approach calculates pts_{past} by using the information of CVE identifiers that vendors mark as fixed in firmware release notes.[27] If vendors do not mention fixed CVEs in release notes, SAFER uses its second approach to calculate pts_{past}. This approach estimates patches by identifying software libraries that changed to non-vulnerable versions in consecutive firmware images. Hence, $X^{patch\_time\_spans}$ of Equation (13) combines the set of all past patch intervals with SAFER's predicted time spans that a manufacturer potentially needs to patch vulnerabilities in the future. Applying the median to this set and classifying it according to Equation (15) implicitly shows the adversary type that will potentially be able to infiltrate the device before a vendor provides a patch.

To calculate the future PT category, first we use the training data containing the days a vendor required to patch the firmware images of a device model and apply the prediction models on this data. This lets SAFER estimate the time needed for a vendor to patch their device model in the future. Second, we calculate the PT category based on the predicted data and compare it with the calculated category of the test set that contains real observations. In our experiment, the two best prediction models to predict the PT category are ARIMA with 47.37% accuracy and Prophet with 100% accuracy. This correlates with the previously discussed MAD values per prediction model. To consider the variation of the PT data in prediction models, we configured Prophet with the multiplicative model and set the parameter *changepoint-prior-scale* to 1.7. Due to achieved results,

[25]For example, https://www.seclists.org/fulldisclosure/ or https://www.github.com/offensive-security/exploitdb.
[26]https://github.com/offensive-security/exploitdb.
[27]http://ftp.axis.com/pub_soft/MPQT/M2026-LE_Mk_II/8_40_3/M2026-LE_Mk_II_8_40_3_release_notes.txt: 8.10.1 Corrected security vulnerability CVE-2017-9798.

Table 4. Future Device Security Risk Indicators

		Patch Trend		
		Fast	**Medium**	**Slow**
Vulnerability Trend	**Low**	Low	Low	Medium
	Medium	Low	Medium	High
	High	Medium	High	Critical

we use the Prophet prediction algorithm to forecast future model patch times. The PT combined with the VT result in the FDSRI, which we introduce in the following.

5.4.3 Future Device Security Risk Indicator. Besides knowing the current risk, it is also relevant for IoT device owners to have an estimate of what risk a device might pose to their networks in the future. We use the VT and PT in a heuristic to provide this novel FDSRI. The FDSRI is based on the VT and PT, which both contain past observations and future predictions for device models.

We show the possible FDSRI combinations in a risk matrix (Table 4). A medium patching manufacturer with unpatched low-risk vulnerabilities for device models may face advanced security experts. Those are able to exploit low-risk vulnerabilities like information disclosure, but most probably do not exploit a major vulnerability. We consider this as a *low FDSRI*, even if the manufacturer does not patch vulnerabilities quickly. We define a *medium future risk* for device models with, for example, high rated and recently registered vulnerabilities under the consideration of fast patching manufacturers. We argue that at this moment, only skilled attackers can exploit such recent vulnerabilities. We consider that those vulnerabilities are likely to be patched by the vendor in the near future due to the fast PT. We define a *high FDSRI* to device models that expose, for example, high-risk vulnerabilities in combination with vendors that patch vulnerabilities rather slow. This increases the exposed risk of a device model over time since it is not likely that the vendor can cope with the evolving device risk. The worst-case FDSRI is a slow patch behavior combined with mostly high-risk vulnerabilities left unpatched. In this case, script kiddies are potentially able to exploit major vulnerabilities without any security knowledge and are able to take over a device. We rate this case as a *critical FDSRI*. We predict those combinations for users to assist and warn them of the future risks a device can expose to a network, even if latest firmware versions are applied at release date.

To determine the FDSRI per device model, SAFER needs to combine the device model's PT and VT. Table 4 shows the possible combinations we use to calculate the FDSRI. We evaluated our approach by first generating the VT and PT for the train set and test set including the FDSRI. Afterward, SAFER predicted the patch intervals and future severity levels for the test set we used to calculate the PT and VT. Ultimately, we calculated the FDSRI using the predicted trends and compared the trend with the real observations.

The main requirement for the FDSRI is the correct classification of the VT and PT. SAFER achieves this by combining different prediction models for both trends. Moreover, the Prophet prediction model—achieving best results for PT—was parameterized in an evaluation based on SAFER's patch data to match analyzed device models. The ARIMA prediction model used for the VT gets individually parameterized in an automated fashion based on the data of each device model. This lets SAFER predict the correct VT and PT categories as shown in Sections 5.4.1 and 5.4.2. By stating a correct prediction per device model, we define that SAFER predicted the identical category of the test set containing real data ahead of the training set. We use the test set only for validation and do train SAFER's prediction algorithms on the training set only containing prior

Table 5. Assessed Devices of the Organization Grouped by Device Model

Type	CDSRI	Models	VT	Models	PT	Models	FDSRI	Models
CCTV	Low	4	Low	4	Slow	0	Low	4
	Medium	0	Medium	0	Medium	0	Medium	0
	High	0	High	0	Fast	4	High	0
	Critical	0					Critical	0
IP-2-X	Low	1	Low	2	Slow	0	Low	2
	Medium	0	Medium	0	Medium	0	Medium	0
	High	0	High	0	Fast	2	High	0
	Critical	1					Critical	0
IPMI	Low	4	Low	8	Slow	0	Low	8
	Medium	0	Medium	0	Medium	0	Medium	0
	High	0	High	0	Fast	8	High	0
	Critical	4					Critical	0
NAS	Low	3	Low	3	Slow	0	Low	3
	Medium	0	Medium	0	Medium	0	Medium	0
	High	0	High	0	Fast	3	High	0
	Critical	0					Critical	0
Printer	Low	3	Low	15	Slow	0	Low	15
	Medium	0	Medium	0	Medium	15	Medium	0
	High	0	High	0	Fast	0	High	0
	Critical	12					Critical	0
Telepresence	Low	2	Low	4	Slow	0	Low	6
	Medium	0	Medium	2	Medium	1	Medium	0
	High	0	High	0	Fast	5	High	0
	Critical	4					Critical	0

data. We state that SAFER—implementing the preceding mechanisms to predict CDSRI, PT, VT, and FDSRI—is able to identify both the current and future risk correctly for the 38 device models. We point out that those assessed device models refer to a total of *240 physical IoT devices in our organization.*

By using SAFER, we identified that 21 out of 38 assessed IoT device models at CERN have in their current setup (CDSRI) a high likelihood to expose critical vulnerabilities to the network. The CDSRI is a user-centric indicator representing the risk a device exposes in its current firmware setup managed by the device owner. Contrary to the CDSRI, the FDSRI is a vendor-centric indicator that considers the behavior of how timely a vendor addresses vulnerability patches over, for example, multiple years. Considering such behaviors over time by the FDSRI lets SAFER calculate different FDSRI results for devices with a currently critical CDSRI.

The FDSRI shows that CERN's device models—showing a critical CDSRI at the moment—are likely to expose a low risk in the future. This refers to (1) the vendor patching behavior for 22 out of 38 device models that SAFER identified as fast patching; in addition, (2) vendors increase their security awareness that SAFER identifies by a decrease of high-risk vulnerabilities. SAFER predicts a low VT for 36 out of 38 device models. We show a full overview of all risk metrics for the 38 device models in Table 5.

To conclude, we showed throughout Section 5.4 how SAFER calculates its FDSRI and both depending trends PT and VT. We state that SAFER's category predictions achieve 100% accuracy for

PT, VT, and FDSRI classifications. To make our prediction errors more transparent, we mention them for our PT (14.31 days MAD) and VT (1.31 CVSS MAD) predictions.

6 DISCUSSION

6.1 Limitations

Although SAFER performed well in our assessments, we list a few limitations of our prototype. As mentioned in Section 4.3.2, WPD needs manual effort if, for example, major firmware releases change the position of the model and firmware information on a devices' web page. Thus, we plan to use current findings of WPD to train a machine learning classifier or use natural language processing to help us identify more devices with less manual effort.

To enhance the correct device detection, SAFER applies subjective logic to fuse the results from individual mechanisms achieving an identification rate of 92.55% for 2,134 fusion operations. We achieve this identification rate on a reduced set of fusion operations, due to omitting 360 fusion operations where CCD and WPD did not identify a device.

The vulnerability- and patch detection approach of SAFER considers entire software libraries and is based on information SAFER retrieves from public sources such as the NVD. Thus, SAFER cannot identify zero day vulnerabilities or, for example, source code patterns leading to potential vulnerabilities. The same applies to patches that require source code analysis. Individual libraries may contain multiple vulnerabilities where not all of them are fixed in successive patches, and we state that SAFER misses such cases because it does not investigate in a fine-grained level as, for example, individual lines of code.

SAFER considers the "created" date of a CVE as the date the CVE was publicly registered. SAFER cannot identify if at this date the CVE contained vulnerability information or temporary placeholder information. We assume that when a CVE gets registered for a product, the vendor knows about the vulnerability and is already able to work on patches.

SAFER's category predictions achieve 100% accuracy for PT, VT, and FDSRI classifications. To make our prediction errors more transparent, we mention them for our PT (14.31 days MAD) and VT (1.31 CVSS MAD) predictions. Thus, limitations rise for more fine-grained classifications of the VT (Equation (12)) and PT (Equation (15)).

6.2 Comparing SAFER with Related Work

Related work focuses on subsets of tasks to secure IoT devices. We developed with SAFER a highly automated encompassing solution with so-called components where each focuses on a different task. An advantage of SAFER is the use of a multitude of identification mechanisms to identify devices and not being bound to a single mechanism as related work [6, 19, 54]. SAFER retrieves and extracts firmware images of various vendors using static analysis being less resource intense than other related work [14] and enabling higher automation capabilities. SAFER analyzes firmware images for contained software libraries and retrieves their publicly known vulnerabilities. Additionally, SAFER retrieves publicly known device model vulnerabilities (as in the work of Miettinen et al. [38]), which leads in combination with SAFER's found firmware vulnerabilities to a comprehensive dataset for the risk scoring component.

The majority of vulnerability prediction approaches require the public availability of source code for all software versions. Massacci and Nguyen [37], for example, compare—exemplary for Firefox—18 different vulnerability approaches of related work with their own approach, whereas the 12 best-performing ones focus on vulnerability prediction. Most prediction approaches focus on the source code itself as, for example, using source code metrics, the version at discovery, and references to the code base. Other vulnerability prediction works require a specific amount of past software patches in the source code (e.g., [21]) to be present on which a machine model can learn on. Other works are limited to specific programming languages that they can parse and process (e.g., [15]). The last group does not only require source code availability but also a large code base (e.g., [25]) to perform predictions on. Even compared to other related work [17, 34, 47, 53, 62, 63] to find vulnerabilities in, for example, source code, SAFER uses only public sources to fetch known vulnerabilities. Related work [18, 33, 60, 61] doing this as well achieved accurate results. We state that by using SAFER's approach to retrieve vulnerabilities, SAFER is not limited in detecting vulnerabilities for specific programming languages, closed-source software, and so on, and does not require a large source code base as related work.

SAFER's risk scoring component uses the CVSS standard (in comparison to modified CVSS versions of related work [23, 48, 58, 59]) to generate universal and comparable metrics. This is a relevant fact to make device comparisons for different device categories possible.

Moreover, the last component calculates the CDSRI and predicts the FDSRI for device models. The user interface [44] of SAFER has different risk visualizations for technical and non-technical users. Using SAFER, one can make informed decisions about the device's security in its current setup (CDSRI) and how it is estimated to evolve in the future (FDSRI).

Ultimately, the entire framework was tested in CERN's large-scale network on which not all solutions of related work (e.g., [5, 38]) are suitable.

7 SUMMARY AND FUTURE WORK

7.1 Summary

In this article, we have presented and evaluated the SAFER framework, which aims to automate the analysis of IoT networks with respect to the security risk that IoT devices in this network may pose. To achieve this, SAFER first scans the network and tries to identify devices regarding brand, model, and firmware version using multiple different identification mechanisms. As our first contribution, we have presented and evaluated two such mechanisms: an enhanced version of CCD that is based on unique clock characteristics and the new WPD that evaluates information from a device's web page. We also presented a fusion approach based on subjective logic. With this approach, we could correctly identify 531 out of 572 (93%) of the IoT devices in our dataset, whereas WPD alone achieved only 77.09% and CCD 21.90%, respectively, showing the benefit of the fusion approach.

SAFER's firmware analysis component, which is based on an enhanced FACT framework and an automated crawling process, was then used to derive 825 firmware images for IoT devices from vendor pages and automatically analyze them for included libraries and various vulnerabilities. This was followed by the steps of vulnerability enrichment and scoring. Scanning vulnerability databases like the NVD, SAFER retrieves CVEs for firmware images and contained libraries to automatically determine known vulnerabilities for the DUT and its software. These automation steps add to the third contribution.

This information is then used to calculate a number of risk indicators that represent contribution 2: the CDSRI is based on the identified vulnerabilities that are present in the devices at the time of analysis and represent the risk of the device to be exploited now. The front-end visualizes this score to the user but also provides information on available firmware updates that might mitigate this risk. In contrast, the FDSRI estimates how problematic a device—which could even be secure from the CDSRI perspective—might become in the future because of a rich history of vulnerabilities or lacking of timely patches from its vendor. FDSRI is calculated based on the VT and PT, which are established using prediction models such as Prophet or ARIMA.

Although CDSRI is a straightforward risk indicator, we evaluated the FDSRI through an experiment with 38 different device models and showed that the development of risk category can be accurately predicted based on historic data. As SAFER was actively used in a realistic and large network at CERN with thousands of IoT devices and dozens of different device models, our evaluations provide a clear indication of both the feasibility and scalability of our approach. SAFER therefore is a suitable tool to establish IoT security awareness in large-scale networks as an *extendable, scalable, and highly automated risk assessment framework for IoT devices*. However, development and evaluation of SAFER is not concluded, and we see a number of ways to extend SAFER in the future, which we discuss next.

7.2 Future Work

For our future work, we plan to automate the tasks of SAFER's setup phase for which some manual work is still required. Although we foresee that an active community can cover tasks relevant for assessing an unknown device, future work should look deeper into automating these tasks as much as possible. We now discuss some relevant aspects.

7.2.1 CCD Data Acquisition. CCD's time frame of 48 hours to acquire input before performing a device identification is relatively long in comparison to faster approaches such as WPD. We identified that simply fetching more timestamp scans over time enabled us to reduce the timestamp retrieval interval from 48 hours to 4 hours and 20 minutes. We evaluated that with 10,000 scans per device model, CCD only requires a scan to contain 100 instead of 576 timestamp values by still achieving a similar (<1.0%) identification accuracy. We believe that gathering more timestamp scans by a community-based approach lets us decrease CCD's identification time frame even further, but this has to be evaluated.

7.2.2 Firmware Retrieval and Analysis. SAFER relies on the availability of firmware images to analyze and determine contained software. Manufacturers may actively limit access to firmware or obfuscate it to hinder reverse-engineering. In this case, vendors or trustworthy users may conduct the analysis and upload the resulting data to the SAFER repository. Such a central device repository would enable users to gain more accurate risk scores and identify new devices more quickly.

Alternatively to analyzing firmware images, we already implemented a prototype component for SAFER that parses license statement files.[28] In those files, vendors frequently state the licenses of contained libraries as demanded, for example, by the GNU General Public License. Those license statement files often include the software's name and its exact version and is another valuable information source to SAFER where an analysis of the firmware binary is not necessary. This and similar extensions will extend the scope of SAFER significantly.

To let researchers use the results of our work, we already published on SAFER's website[29] all analyzed firmware images and our YARA rules to detect IoT-related software within firmware images. We plan to release the entire SAFER framework in 2022 to let others benefit from SAFER's possibilities by applying SAFER to their networks.

APPENDICES

A SUBJECTIVE LOGIC BACKGROUND

Subjective logic summary. The fundamental object in subjective logic is a *subjective opinion* ω_X^A held by actor A over a random variable X. It is a triplet of the *belief* function $\mathbf{b}_X^A$ and the *base rate*

[28]http://ftp.axis.com/pub_soft/MPQT/Q8722-E/latest/thirdpartysoftwarelicenses_Q8722-E.txt.
[29]https://safer.network.

function $\mathbf{a}_X^A$, which are defined over the finite domain $\mathbb{X}$ of the random variable X, along with a number u_X^A that represents *uncertainty*. The triplet is typically written as

$$\omega_X^A = \left(\mathbf{b}_X^A, u_X^A, \mathbf{a}_X^A\right).$$

The following constraints apply to all opinions:

$$1 = u_X^A + \sum_{x \in \mathbb{X}} \mathbf{b}_X^A(x), \text{ where } 0 <= u_X^A <= 1 \text{ and } \mathbf{b}_X^A, \mathbf{a}_X^A : \mathbb{X} \to [0, 1].$$

In some instances, it can be useful to define the *certainty* c: this is the sum of all beliefs, which by the preceding definitions is equivalent to the inverse of the uncertainty: $c_X^A = \sum_{x \in \mathbb{X}} \mathbf{b}_X^A(x) = 1 - u_X^A$.

Fusion operators. Subjective logic defines various fusion operators to combine opinions of multiple actors. This enables subjective logic to be used in a variety of applications; the “correct” fusion operator is different depending on the desired outcome in the case of conflicting opinions[30] (e.g., a jury decision or the choice of a group activity). In this work, we use the CBF operator, which represents the case where evidence from each opinion is additive. This choice is appropriate for the SAFER scenario, as each mechanism provides independent evidence. For a detailed discussion of different operators, refer to the work of Jøsang [27]. Given a set of opinions,[31] the operator is defined as follows [28]. For a set of actors $\mathbb{A}$, the CBF fusion result $\omega_X^{\diamond(\mathbb{A})}$ has the following belief function [28][32]:

$$\mathbf{b}_X^{\diamond(\mathbb{A})}(x) = \frac{\sum_{A \in \mathbb{A}} \left(\mathbf{b}_X^A(x) \prod_{A_j \neq A} u_X^{A_j}\right)}{\sum_{A \in \mathbb{A}} \left(\prod_{A_j \neq A} u_X^{A_j}\right) - (|\mathbb{A}| - 1) \prod_{A \in \mathbb{A}} u_X^A}. \tag{16}$$

The uncertainty is then defined as

$$u_X^{\diamond(\mathbb{A})} = \frac{\prod_{A \in \mathbb{A}} u_X^A}{\sum_{A \in \mathbb{A}} \left(\prod_{A_j \neq A} u_X^A\right) - (|\mathbb{A}| - 1) \prod_{A \in \mathbb{A}} u_X^A}. \tag{17}$$

As stated in the work of Jøsang et al. [28], the base rate for all inputs is normally the same, and therefore the base rate of the fused opinion is also the same.

[30]In data fusion in general, it is still an open research question whether a generic fusion operator exists. This question is outside the scope of this article.

[31]Note that this definition is for multinomial opinions, which are defined over a domain with at least three disjunct items (i.e., $|\mathbb{X}| > 2$) and where belief is always assigned to a specific outcome. In some applications, the use of more complex hyperopinions enables the representation of belief in an arbitrary subset outcomes, without specifying their relative probability. The disadvantage is that the storage complexity of hyperopinions is $O(2^{|\mathbb{X}|})$, which is why we do not use them here.

[32]For brevity, we only provide equations for the non-dogmatic case, since in SAFER we ensure that no dogmatic opinions are generated. *Dogmatic* opinions are opinions with $u_X^A = 0$, implying infinite evidence.

B ADDITIONAL DATA

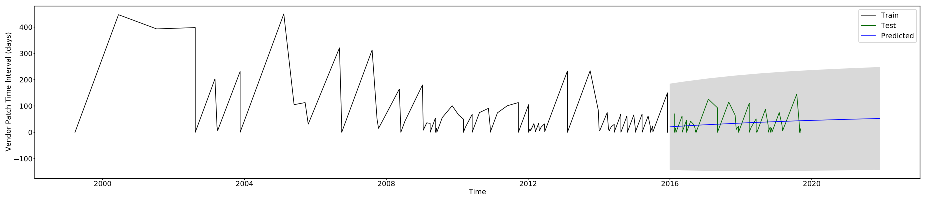

Fig. 4. PT prediction using ARIMA.

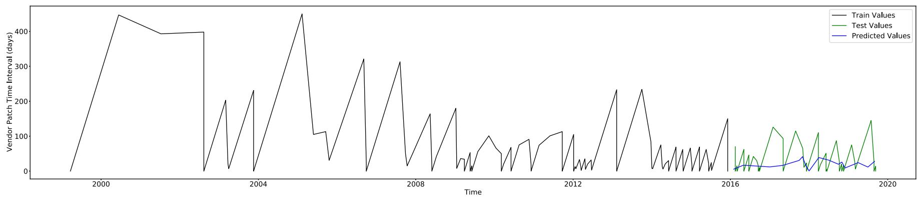

Fig. 5. PT prediction using Prophet.

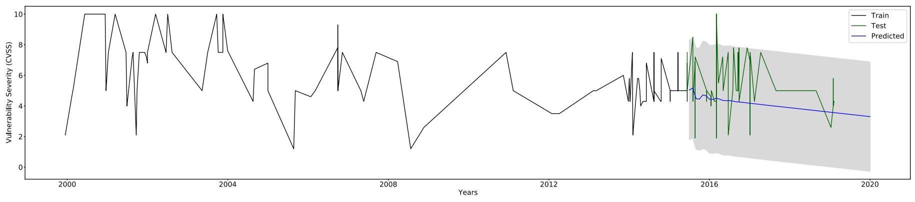

Fig. 6. VT prediction using ARIMA.

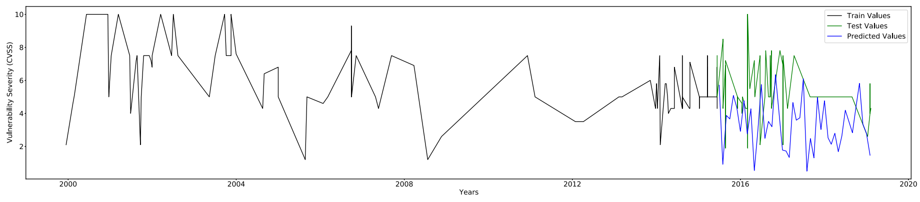

Fig. 7. VT prediction using Prophet.

REFERENCES

[1] Sharad Agarwal, Pascal Oser, and Stefan Lueders. 2019. Detecting IoT devices and how they put large heterogeneous networks at security risk. *Sensors* 19, 19 (2019), 4107.

[2] Bako Ali and Ali Ismail Awad. 2018. Cyber and physical security vulnerability assessment for IoT-based smart homes. *Sensors* 18, 3 (2018), 817.

[3] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. Sok: Security evaluation of home-based IoT deployments. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP'19)*. IEEE, Los Alamitos, CA, 1362–1380.

[4] Ashish Arora, Ramayya Krishnan, Rahul Telang, and Yubao Yang. 2010. An empirical analysis of software vendors' patch release behavior: Impact of vulnerability disclosure. *Information Systems Research* 21, 1 (2010), 115–132.

[5] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. Behavioral fingerprinting of IoT devices. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security.* 41–50.

[6] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. IoTSense: Behavioral fingerprinting of IoT devices. *arXiv preprint arXiv:1804.03852* (2018).

[7] David A. Borman, Robert T. Braden, and Van Jacobson. 1992. *TCP Extensions for High Performance.* RFC 1323. Internet Engineering Task Force. https://doi.org/10.17487/RFC1323

[8] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. 2011. *Time Series Analysis: Forecasting and Control.* Vol. 734. John Wiley & Sons.

[9] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32.

[10] Saikat Chakraborty, Rahul Krishna, Yangruibo Ding, and Baishakhi Ray. 2021. Deep learning based vulnerability detection: Are we there yet. *IEEE Transactions on Software Engineering.* Preprint.

[11] Jiongyi Chen, Wenrui Diao, Qingchuan Zhao, Chaoshun Zuo, Zhiqiang Lin, XiaoFeng Wang, Wing Cheong Lau, Menghan Sun, Ronghai Yang, and Kehuan Zhang. 2018. IoTFuzzer: Discovering memory corruptions in IoT through app-based fuzzing. In *Proceedings of the 2018 Network and Distributed Systems Security Symposium (NDSS'18).*

[12] Kyle Coffey, Richard Smith, Leandros Maglaras, and Helge Janicke. 2018. Vulnerability analysis of network scanning on SCADA systems. *Security and Communication Networks* 2018 (2018), Article 3794603.

[13] RAND Corporation. 2017. Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits. Retrieved September 26, 2019 from https://www.rand.org/pubs/research_reports/RR1751.html.

[14] Andrei Costin, Apostolis Zarras, and Aurélien Francillon. 2016. Automated dynamic firmware analysis at scale: A case study on embedded web interfaces. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security.* 437–448.

[15] Hoa Khanh Dam, Truyen Tran, Trang Pham, Shien Wee Ng, John Grundy, and Aditya Ghose. 2017. Automatic feature learning for vulnerability prediction. *arXiv preprint arXiv:1708.02368* (2017).

[16] Stefan Dietzel, Rens van der Heijden, Hendrik Decke, and Frank Kargl. 2014. A flexible, subjective logic-based framework for misbehavior detection in V2V networks. In *Proceeding of 2014 IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks.* IEEE, Los Alamitos, CA, 1–6.

[17] Ruian Duan, Ashish Bijlani, Yang Ji, Omar Alrawi, Yiyuan Xiong, Moses Ike, Brendan Saltaformaggio, and Wenke Lee. 2019. Automating patching of vulnerable open-source software versions in application binaries. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'19).*

[18] Michel Edkrantz, Staffan Truvé, and Alan Said. 2015. Predicting vulnerability exploits in the wild. In *Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing.* IEEE, Los Alamitos, CA, 513–514.

[19] Xuan Feng, Qiang Li, Haining Wang, and Limin Sun. 2018. Acquisitional rule-based engine for discovering Internet-of-Things devices. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security'18).* 327–341.

[20] Xuan Feng, Xiaojing Liao, XiaoFeng Wang, Haining Wang, Qiang Li, Kai Yang, Hongsong Zhu, and Limin Sun. 2019. Understanding and securing device vulnerabilities through automated bug report analysis. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security'19).*

[21] Aayush Garg, Renzo Degiovanni, Matthieu Jimenez, Maxime Cordy, Mike Papadakis, and Yves Le Traon. 2020. Learning to predict vulnerabilities from vulnerability-fixes: A machine translation approach. *arXiv preprint arXiv:2012.11701* (2020).

[22] Gemini George and Sabu M. Thampi. 2018. A graph-based security framework for securing industrial IoT networks from vulnerability exploitations. *IEEE Access* 6 (2018), 43586–43601.

[23] Oscar M. Guillen, Ralf Brederlow, Ralph Ledwa, and Georg Sigl. 2014. Risk management in embedded devices using metering applications as example. In *Proceedings of the 9th Workshop on Embedded Systems Security.* 1–9.

[24] Danny Yuxing Huang, Noah Apthorpe, Gunes Acar, Frank Li, and Nick Feamster. 2019. IoT inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *arXiv preprint arXiv:1909.09848* (2019).

[25] Matthieu Jimenez, Mike Papadakis, and Yves Le Traon. 2016. Vulnerability prediction models: A case study on the Linux kernel. In *Proceedings of the 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM'16).* IEEE, Los Alamitos, CA, 1–10.

[26] Pontus Johnson, Dan Gorton, Robert Lagerström, and Mathias Ekstedt. 2016. Time between vulnerability disclosures: A measure of software product vulnerability. *Computers & Security* 62 (2016), 278–295.

[27] Audun Jøsang. 2016. *Subjective Logic: A Formalism for Reasoning under Uncertainty.* Springer International, Cham, Switzerland. https://doi.org/10.1007/978-3-319-42337-1

[28] A. Jøsang, D. Wang, and J. Zhang. 2017. Multi-source fusion in subjective logic. In *Proceedings of the 2017 20th International Conference on Information Fusion (Fusion'17).* IEEE, Los Alamitos, CA, 1–8. https://doi.org/10.23919/ICIF.2017.8009820

[29] Patrick Gage Kelley, Joanna Bresee, Lorrie Faith Cranor, and Robert W. Reeder. 2009. A "nutrition label" for privacy. In *Proceedings of the 5th Symposium on Usable Privacy and Security*. 1–12.

[30] Patrick Gage Kelley, Lucian Cesca, Joanna Bresee, and Lorrie Faith Cranor. 2010. Standardizing privacy notices: An online study of the nutrition label approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1573–1582.

[31] Ronny Ko and James Mickens. 2018. DeadBolt: Securing IoT deployments. In *Proceedings of the Applied Networking Research Workshop*. 50–57.

[32] Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. 2005. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing* 2, 2 (2005), 93–108.

[33] Patrick Kwaku Kudjo, Jinfu Chen, Solomon Mensah, Richard Amankwah, and Christopher Kudjo. 2020. The effect of Bellwether analysis on software vulnerability severity prediction models. *Software Quality Journal* 28, 4 (2020), 1413–1446.

[34] Bingchang Liu, Liang Shi, Zhuhua Cai, and Min Li. 2012. Software vulnerability discovery techniques: A survey. In *Proceedings of the 2012 4th International Conference on Multimedia Information Networking and Security*. IEEE, Los Alamitos, CA, 152–156.

[35] Yining Liu, Keqiu Li, Yingwei Jin, Yong Zhang, and Wenyu Qu. 2011. A novel reputation computation model based on subjective logic for mobile ad hoc networks. *Future Generation Computer Systems* 27, 5 (2011), 547–554.

[36] Franco Loi, Arunan Sivanathan, Hassan Habibi Gharakheili, Adam Radford, and Vijay Sivaraman. 2017. Systematically evaluating security and privacy for consumer IoT devices. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. 1–6.

[37] Fabio Massacci and Viet Hung Nguyen. 2010. Which is the right source for vulnerability studies? An empirical analysis on Mozilla Firefox. In *Proceedings of the 6th International Workshop on Security Measurements and Metrics*. 1–8.

[38] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT sentinel: Automated device-type identification for security enforcement in IoT. In *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS'17)*. IEEE, Los Alamitos, CA, 2177–2184.

[39] Mujahid Mohsin, Muhammad Usama Sardar, Osman Hasan, and Zahid Anwar. 2017. IoTRiskAnalyzer: A probabilistic model checking based framework for formal risk analytics of the Internet of Things. *IEEE Access* 5 (2017), 5494–5505.

[40] Mukrimah Nawir, Amiza Amir, Naimah Yaakob, and Ong Bi Lynn. 2016. Internet of Things (IoT): Taxonomy of security attacks. In *Proceedings of the 2016 3rd International Conference on Electronic Design (ICED'16)*. IEEE, Los Alamitos, CA, 321–326.

[41] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum, and Nasir Ghani. 2019. Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2702–2733.

[42] Michele Nitti, Roberto Girau, and Luigi Atzori. 2013. Trustworthiness management in the Social Internet of Things. *IEEE Transactions on Knowledge and Data Engineering* 26, 5 (2013), 1253–1266.

[43] Jason R. C. Nurse, Sadie Creese, and David De Roure. 2017. Security risk assessment in Internet of Things systems. *IT Professional* 19, 5 (2017), 20–26.

[44] Pascal Oser, Sebastian Feger, Paweł W. Woźniak, Jakob Karolus, Dayana Spagnuelo, Akash Gupta, Stefan Lüders, Albrecht Schmidt, and Frank Kargl. 2020. SAFER: Development and evaluation of an IoT device risk assessment framework in a multinational organization. *arXiv:2007.14724 [cs.CR]* (2020).

[45] Pascal Oser, Frank Kargl, and Stefan Lüders. 2018. Identifying devices of the Internet of Things using machine learning on clock characteristics. In *Proceedings of the International Conference on Security, Privacy, and Anonymity in Computation, Communication, and Storage*. 417–427.

[46] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: Analysing the rise of IoT compromises. In *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT'15)*.

[47] Henning Perl, Sergej Dechand, Matthew Smith, Daniel Arp, Fabian Yamaguchi, Konrad Rieck, Sascha Fahl, and Yasemin Acar. 2015. VCCFinder: Finding potential vulnerabilities in open-source projects to assist code audits. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 426–437.

[48] Yanzhen Qu and Philip Chan. 2016. Assessing vulnerabilities in Bluetooth Low Energy (BLE) wireless network based IoT systems. In *Proceedings of the 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity'16), IEEE International Conference on High Performance and Smart Computing (HPSC'16), and IEEE International Conference on Intelligent Data and Security (IDS'16)*. IEEE, Los Alamitos, CA, 42–48.

[49] Saša Radomirovic. 2010. Towards a model for security and privacy in the Internet of Things. In *Proceedings of the 1st International Workshop on Security of the Internet of Things*.

[50] Vinay Sachidananda, Shachar Siboni, Asaf Shabtai, Jinghui Toh, Suhas Bhairav, and Yuval Elovici. 2017. Let the cat out of the bag: A holistic approach towards security analysis of the Internet of Things. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*. 3–10.

[51] Muhammad Shahzad, Muhammad Zubair Shafiq, and Alex X. Liu. 2012. A large scale exploratory analysis of software vulnerability life cycles. In *Proceedings of the 2012 34th International Conference on Software Engineering (ICSE'12)*. IEEE, Los Alamitos, CA, 771–781.

[52] Yun Shen and Pierre-Antoine Vervier. 2019. IoT security and privacy labels. In *Annual Privacy Forum*. Springer, 136–147.

[53] Yonghee Shin, Andrew Meneely, Laurie Williams, and Jason A. Osborne. 2010. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. *IEEE Transactions on Software Engineering* 37, 6 (2010), 772–787.

[54] Sandra Siby, Rajib Ranjan Maiti, and Nils Ole Tippenhauer. 2017. IoTScanner: Detecting privacy threats in IoT neighborhoods. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*. 23–30.

[55] Prashast Srivastava, Hui Peng, Jiahao Li, Hamed Okhravi, Howard Shrobe, and Mathias Payer. 2019. FirmFuzz: Automated IoT firmware introspection and analysis. In *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*. 15–21.

[56] Sean J. Taylor and Benjamin Letham. 2018. Forecasting at scale. *American Statistician* 72, 1 (2018), 37–45.

[57] Mauricio Tellez, Samy El-Tawab, and M. Hossain Heydari. 2016. IoT security attacks using reverse engineering methods on WSN applications. In *Proceedings of the 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT'16)*. IEEE, Los Alamitos, CA, 182–187.

[58] Huan Wang, Zhanfang Chen, Jianping Zhao, Xiaoqiang Di, and Dan Liu. 2018. A vulnerability assessment method in Industrial Internet of Things based on attack graph and maximum flow. *IEEE Access* 6 (2018), 8599–8609.

[59] Ruyi Wang, Ling Gao, Qian Sun, and Deheng Sun. 2011. An improved CVSS-based vulnerability scoring mechanism. In *Proceedings of the 2011 3rd International Conference on Multimedia Information Networking and Security*. IEEE, Los Alamitos, CA, 352–355.

[60] Mark A. Williams, Roberto Camacho Barranco, Sheikh Motahar Naim, Sumi Dey, M. Shahriar Hossain, and Monika Akbar. 2020. A vulnerability analysis and prediction framework. *Computers & Security* 92 (2020), 101751.

[61] Shuang Wu, Congyi Wang, Jianping Zeng, and Chengrong Wu. 2020. Vulnerability time series prediction based on multivariable LSTM. In *Proceedings of the 2020 IEEE 14th International Conference on Anti-Counterfeiting, Security, and Identification (ASID'20)*. IEEE, Los Alamitos, CA, 185–190.

[62] Yang Xiao, Bihuan Chen, Chendong Yu, Zhengzi Xu, Zimu Yuan, Feng Li, Binghong Liu, et al. 2020. MVP: Detecting vulnerabilities using patch-enhanced vulnerability signatures. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security'20)*. 1165–1182.

[63] Zhengzi Xu, Bihuan Chen, Mahinthan Chandramohan, Yang Liu, and Fu Song. 2017. SPAIN: Security patch analysis for binaries towards understanding the pain and pills. In *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE'17)*. IEEE, Los Alamitos, CA, 462–472.

[64] Emrah Yasasin, Julian Prester, Gerit Wagner, and Guido Schryen. 2020. Forecasting IT security vulnerabilities—An empirical analysis. *Computers & Security* 88 (2020), 101610.

[65] Zhiyuan Zheng, Allen Webb, A. L. Narasimha Reddy, and Riccardo Bettati. 2018. IoTAegis: A scalable framework to secure the Internet of Things. In *Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN'18)*. IEEE, Los Alamitos, CA, 1–9.

Received January 2021; revised November 2021; accepted December 2021