SoK: Understanding the state of IoT-specific vulnerabilities via CVE characterization with LLIoT

Tina Rezaei, Suzan Bayhan, Andrea Continella, Jeroen van der Ham-de Vos, and Roland van Rijswijk-Deij
{*t.rezaei, s.bayhan, a.continella, j.vanderham r.m.vanrijswijk*}*@utwente.nl, University of Twente*

***Abstract*—Following the expansion of IoT systems, spanning from devices to cloud backends, reported IoT CVE vulnerabilities have increased at an alarming pace. Since most IoT attacks exploit known vulnerabilities, understanding known vulnerabilities is vital for defense and security research. In this work, we systematize the prior research on studying IoT vulnerabilities, revealing the absence of consistent IoT definitions, reliable and scalable classification methodologies, and high-quality IoT CVE datasets. To overcome these limitations, we design LLIoT, a novel and LLM-assisted approach that systematically and automatically distinguishes IoT-specific CVEs at large scale, enabling in-depth understanding of IoT vulnerabilities. First, leveraging the systematization knowledge from the literature, we derive a four-layer IoT ecosystem taxonomy and define classification criteria for distinguishing IoT CVEs. Then, using an expert-validated ground-truth dataset, we demonstrate that LLMs can reliably distinguish IoT from non-IoT CVEs with a high accuracy of 95%, outperforming humans by avoiding cognitive errors and gaps in domain knowledge. Applying LLIoT to CVEs from 2013–2024, we build a dataset of 15,116 IoT-specific vulnerabilities, of which 8,368 are newly classified with respect to previous datasets. Using this dataset, which we share with the research community for further research and reproducibility, we characterize how IoT vulnerabilities differ from traditional IT vulnerabilities. Upon our observation, we provide actionable recommendations for responsible stakeholders.**

***Index Terms*—IoT security, vulnerability analysis, IoT CVE dataset**

1. Introduction

The Internet of Things continues to experience drastic growth, with a forecast of 31 billion devices by 2030 [1]. Therefore, IoT devices have become increasingly attractive targets for attackers, resulting in a steady rise in reported vulnerabilities [2]–[4]. Large-scale empirical studies show that IoT vulnerabilities now span the entire technology stack, from resource-constrained firmware to cloud back-ends, undermining confidentiality, integrity and availability on a global scale [5]. Moreover, as shown by Feng et al. [6], IoT attacks almost exclusively use known vulnerabilities for exploitation. Therefore, studying known IoT vulnerabilities is critical for understanding the weaknesses in the IoT development cycle, preventing exploitation, and designing new defense mechanisms.

The National Vulnerability Database (NVD)[1] serves as the primary repository for publicly disclosed vulnerabilities and is a valuable resource for cybersecurity analysis. However, despite thousands of published CVEs, the community lacks a reliable way to identify which ones affect IoT systems, how their properties differ from traditional IT vulnerabilities, where the most critical weaknesses lie and what mitigations are required. Previous studies rely on Common Platform Enumeration (CPE) metadata [7], [8]—a structured identifier that specifies the affected vendor, product, and version —or keyword-based filtering followed by manual inspection [9]–[14] to collect IoT-related CVEs. However, these studies are limited in scale and suffer from high false positives, restricting in-depth analysis. These limitations stem from dictionary limitations, evolving IoT terminology, incomplete CPE metadata (typos and aliases in vendor and product strings), and the high cost of manual review.

Meanwhile, recent advancements in LLMs offer new possibilities. LLMs exhibit strong contextual reasoning, perform well on classification tasks without labeled data, and show strong performance in vulnerability detection and security analysis [15]–[18]. However, it is unclear whether LLMs can reliably distinguish IoT from non-IoT CVEs at scale, and no prior work has systematically explored their effectiveness for this purpose.

To address these limitations, we systematize the knowledge on IoT vulnerabilities and operationalize this systematization by designing LLIoT, which systematically identifies IoT-specific CVE vulnerabilities at scale using LLMs and analyzes the landscape of IoT-specific vulnerabilities. We first review prior work on IoT taxonomy and formalize a four-layer definition of the IoT ecosystem comprising device, communication, application, and cloud-backend layers and develop classification rules to distinguish IoT from non-IoT CVEs. Afterwards, to evaluate the effectiveness of LLMs for the IoT CVE classification, we construct a high-quality ground truth dataset reviewed by human experts. Provided the ground-truth labels, we evaluate state-of-the-art LLMs for IoT vs non-IoT classification. LLMs achieve up to 95% accuracy, demonstrating their suitability for this task. Moreover, comparing human and LLM reasoning, LLMs exhibit fewer cognitive errors on ambiguous phrasing and bridge domain knowledge gaps in humans.

Given the high performance of LLMs, we scale the LLM classification and classify all CVEs from 2013–2024, evaluate the consistency and reliability of the resulting labels, and benchmark against two previous datasets. The labels exhibit high consistency (92% unanimity), and our conducted manual inspection shows that the resulting dataset is of higher quality and contains more

1. http://nvd.nist.gov/

IoT-specific CVEs than both baselines. Using this dataset, we further characterize IoT vulnerabilities to understand how their properties differ from traditional non-IoT vulnerabilities and where the most pressing weaknesses lie. Among other trends, we observe that IoT CVEs skew toward lower attacker effort (e.g., no required user interaction or privileges) yet higher impact on confidentiality, integrity, and availability relative to non-IoT. We also observe memory safety issues as the most common weakness in the IoT sector followed by web-centric weaknesses showing potential increase recently. Upon our observation, we identify root causes of most common weaknesses in the IoT and provide actionable recommendations.

The major contributions of this work are as follows:

- We systematize the IoT ecosystem through a 4-layer IoT ecosystem definition and define IoT vs. non-IoT CVE classification rules. Therefore, unlike previous studies, we provide a comprehensive classification guideline to guide consistent labeling.
- We create a high quality ground-truth labeled dataset of IoT and non-IoT CVEs through expert review, accompanied by documented human reasoning. This dataset enables evaluation of LLM performance on distinguishing IoT and non-IoT vulnerabilities and analyzing the reasoning errors in both human and LLMs to highlight sources of misclassifications. We also release the full ground-truth dataset, including human reasoning, for public use.
- We systematically classify CVEs from 2013 to 2024 and build a dataset of 15,116 IoT-specific CVEs, which is, to the best of our knowledge, the most comprehensive and accurate labeled dataset of IoT-specific CVEs to date. Compared with prior datasets, we identify an additional 8,368 unique IoT vulnerabilities and improve labeling quality. We make this dataset public for future use by researchers[2].
- We characterize IoT-specific and non-IoT CVEs and reveal changes in their temporal patterns, differences in their severity, and the most common weakness families. Following our findings, we reveal the causes of IoT vulnerabilities and outline actionable recommendations for stakeholders.

2. Systematization of prior work

We organize prior work into three categories. First, we review studies that propose architectural abstractions of IoT systems to analyze vulnerabilities across different components of the IoT ecosystem. This body of work in turn informs the definition of IoT-specific criteria. Second, we examine prior work on IoT CVE datasets and methods for distinguishing IoT-related CVEs from general CVEs. Third, we review studies that analyze IoT vulnerabilities, characterizing their properties, prevalence, and impact using existing datasets or real-world measurements.

IoT taxonomy. Prior IoT taxonomies largely adopt layered architectures but differ in their abstraction level and perspective. A large body of work follows a three-layer taxonomy consisting of device/perception, network/communication and application layers [19]–[27]. These studies provide abstract functional decompositions of IoT systems. While most of them share a similar structure, their definition of each layer differ slightly. For example, while most of them lack cloud services in their definition, Haddad Pajouh et al. [25] incorporate REST APIs and cloud computing services into the application layer, whereas the network layer is defined to include communication protocols (e.g., WiFi, ZigBee, Bluetooth) as well as cloud back-end connectivity.

Some other studies extend the three-layer model by separating service or middleware functionality from application layer, adding a new layer between network and application, often referred to as a service, middleware, data, or support layer [28]–[32]. This intermediate layer typically receives data from the network layer, performs processing and storage, and provides interfaces or APIs to the application layer, and often incorporates cloud and database systems. The application layer thereafter supports domain-specific and user-facing services. Guth et al. [33] propose a related but more platform-oriented reference architecture consisting of device, gateway, middleware, and application layers. The gateway layer provides connectivity and protocol translation between devices and the middleware in case devices cannot directly connect to the middleware. The middleware receives, processes, and manages device data. Alrawi et al. [5] propose an abstract IoT security model with four main components which includes device, mobile app, communication channel, and cloud backend to systematize attack vectors, mitigations, and stakeholders. Similarly, Rondon et al. [34] divide enterprise IoT into four layers of device, communication, monitoring and application, and business (cloud) layer, where the business layer captures cloud-based services and external system integrations.

In contrast to these architecture-oriented taxonomies, Mohanta et al. [35], and Yang et al. [36] adopt a networking-stack perspective, organizing IoT architecture into physical, transport, network, and application layers to analyze protocol-level security threats. Similarly, Di Martino et al. [37] propose a more fine-grained six-layer technological stack comprising sensing, short-range communication, gateway, network, service platforms, and application layers.

Takeaway. Overall, prior IoT taxonomies vary in granularity and perspective. Most prior work treats cloud and backend infrastructure as part of higher-level layers or middleware, rather than modeling them as distinct components despite their growing role in IoT systems. Moreover, from this synthesis, we learn that a clear and explicit representation of the core components of the IoT ecosystem is required for further accurate analysis of IoT vulnerabilities.

IoT CVE distinction. Prior efforts on distinguishing IoT CVEs are quite limited and sources on IoT CVEs are fragmented or domain-specific. A subset of works explicitly aims to construct IoT vulnerability datasets. VarIoT [38] proposes an IoT vulnerability dataset created by aggregating numerous public data sources, including NVD. They focus primarily on vulnerabilities related only to IoT devices and use keyword filtering at multiple levels to distinguish IoT CVEs. However, Alsadi et al. [39] later acknowledge that their dataset has limitations in accurately classifying IoT-related vulnerabilities. Chen et

2. The code and the dataset can be found at: https://github.com/Tina-Rezaei/LLIoT

al. [40] performed keyword-based filtering followed by manual inspection, to filter IoT CVEs which resulted in a dataset of 1,739 IoT-related CVEs. However, their dataset is limited and includes non-IoT vulnerabilities, e.g., CVE-2017-7054, CVE-2016-7596, CVE-2017-2420, CVE-2020-26143, affecting Windows and macOS, despite excluding general-purpose computers from IoT by definition. Kumar et al. [41] analyzed IoT vulnerabilities reported in the CVE database from 2019 to 2023 to uncover prevalent vulnerability types and assess their severity. Their approach, however, relied solely on keyword filtering without manual validation, limiting its precision. Besides scale and accuracy limitations, these studies lack a clear and comprehensive definition of IoT for their classification, and their decision criteria remain ambiguous.

Another line of work derives IoT-related CVEs indirectly from IoT contexts (e.g., security reports, firmware corpora) and extracts or links the relevant CVE IDs [3], [42], [43]. While these works do not aim to build a general IoT dataset, their methodology offers insight into how IoT vulnerabilities are surfaced. For example, Feng et al. [6] adopt a report-centric approach, first collecting vulnerability disclosures from blogs, forums, and mailing lists, and then identifying IoT-related reports using NLP-based techniques. Then they extract CVE identifiers only when present, making CVEs a by-product of the report analysis. Zhao et al. [44] and Oser et al. [45] collect a large set of IoT firmware and devices and extract all their third party components (TPCs) and their version. Then they leverage version check to identify the CVEs matching the version of the detected TPCs. Therefore, the obtained CVEs in these approaches are inherently domain-specific (e.g., firmware TPCs) and concern TPCs that might also be widely used in non-IoT environments.

Finally, another approach explored in the literature relies on structured metadata for classification. Blinowski et al. [46] use CPE identifiers and a machine learning technique (SVM) to classify CVEs into IoT-related categories such as home systems, SCADA, PCS, and mobile platforms. However, their classification mixes both IoT and non-IoT domains, limiting its suitability for isolating IoT-specific vulnerabilities. Moreover, the reported classification performance remains moderate, reflecting the challenges of relying solely on structured CPE metadata for accurate vulnerability categorization. Rivera et al. [47] propose a time-to-fix prediction framework for IoT devices based on survival analysis and machine learning. To filter IoT CVEs, they use the NVD "product type" field, retaining only entries labeled as hardware ("h"). While this treatment provides a systematic approach, it overlooks vulnerabilities in the firmware, operating systems, and application layers that are central to IoT ecosystems.

Takeaway. Overall, efforts to build IoT vulnerability datasets remain significantly limited. Across these works, three core limitations emerge: (i) lack of a consistent definition of IoT for IoT CVE distinction, (ii) reliance on weak heuristics such as keyword filtering or metadata, and (iii) limited scalability and validation. As a result, existing datasets contain substantial false positives and fail to capture large portions of the IoT CVE landscape.

IoT vulnerability analysis. In this part, we review studies that focus on analyzing vulnerabilities specific to IoT. Alsadi et al. [39] analyze factors influencing the exploitation of IoT vulnerabilities using features such as CVSS scores, CWE categories, vendor information, and signals from external sources such as hacking forums, while relying on the VarIoT dataset. Similarly, Perez et al. [48] use the same dataset to investigate vendor-level security performance for IoT-centric and non-IoT centric vendors. Arat et al. [49] propose a graph-based method for risk detection and vulnerability assessment of IoT systems. They evaluate it on a smart-home scenario where multiple CVE entries (with CVSS scores) are assigned to devices to compute node, path, and graph risk levels. Zhao et al. [50] conduct a ten-month, internet-wide measurement of 1,362,906 deployed IoT devices and investigate 73 N-day IoT CVEs via version matching, finding 28.25% of devices still vulnerable and thousands showing signs of botnet infection. Fang et al. [51] proposes a system-level IoT security analysis framework that constructs attack graphs by mapping CVEs to devices and modeling their exploit dependencies. By combining vulnerability information with device interactions, it identifies multi-step attack paths and quantifies their impact using metrics such as attack complexity and blast radius.

Takeaway. Overall, these studies largely rely on existing IoT vulnerability datasets. As a result, their findings are inherently constrained by the quality and coverage of the underlying data. Consequently, dataset limitations propagate to downstream analyses, affecting both the reliability and completeness of their conclusions.

Our systematization of prior work reveals that analyzing IoT vulnerabilities requires a unified architectural abstraction that covers all contributing components. It also further revealed that current identification of IoT-specific CVEs relies on ambiguous IoT definitions and non-scalable and non-accurate methodologies. These limitations obscure the true characterization of IoT vulnerabilities and undermine the validity of subsequent analyses. To address these limitations, we derive a four-layer IoT taxonomy and leverage it to build a scalable, LLM-assisted classification approach. This approach enables a consistent, large-scale identification of IoT-specific CVEs and better understanding of IoT vulnerability characteristics across the broader CVE landscape. We release our datasets and framework to support further research in this area and enable reproducibility.

3. Methodology

In this section, we present our methodology behind LLIoT, which follows three main phases consisting of 6 steps as shown in Fig. 1. We refer to the corresponding steps in the figure as we describe each phase of the study.

Expert review: In the first phase, the goal is to answer how IoT-specific vulnerabilities can be systematically distinguished from non-IoT vulnerabilities. This requires a clear definition of the IoT ecosystem to ensure consistent distinction. Therefore, first (step 1), we present a four-layer IoT taxonomy to define an IoT ecosystem. Based on this taxonomy, we define classification rules for distinguishing IoT-specific and non-IoT CVEs. Afterwards (step 2), we construct a high-quality ground-truth dataset to benchmark LLM performance for the classification task. To build the ground-truth dataset, we create a dictionary of IoT-specific keywords and use it to filter potential IoT

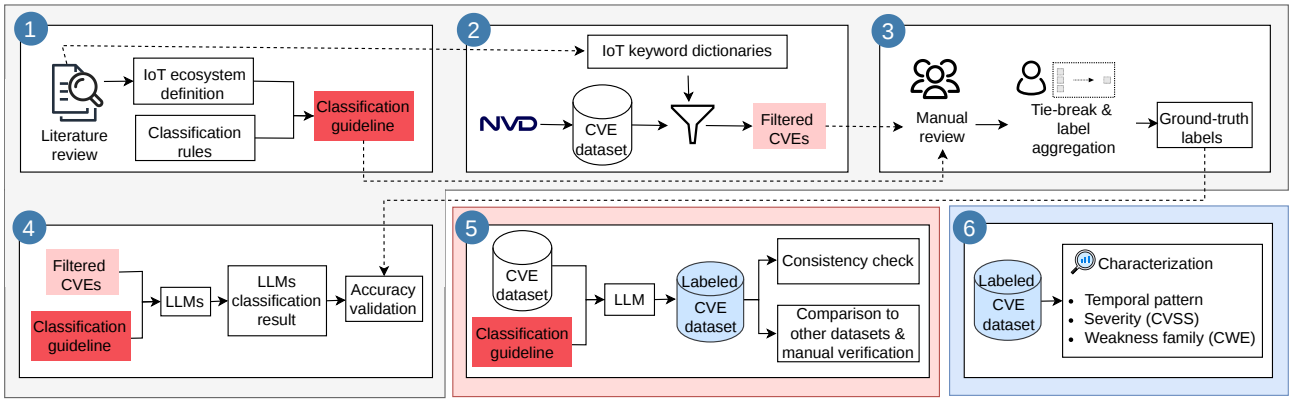


Figure 1: An overview of LLIoT with three phases: expert review (step 1 to step 4), dataset assessment (step 5), and vulnerability characterization (step 6).

CVEs. Then (step 3), we manually review each CVE following the classification guideline. Each CVE is reviewed by three independent experts and disagreements are resolved by a tie-breaker. We use Krippendorff's α [52] to measure the agreement level among reviewers. An α of 1 indicates perfect agreement; values above 0.80 are satisfactory, 0.67–0.79 allow tentative conclusions, values below 0.67 indicate poor agreement, and 0 reflects no agreement [53]. Afterwards (step 4), we apply four leading LLMs to the ground-truth dataset, providing them with the classification guideline to perform the CVE classification. We select closed-source models, GPT-4o and O3, and open-source models, Deepseek-r1-70B and Llama4-scout, for this experiment. Their performance is assessed using accuracy, precision, recall, and F1 score to determine the feasibility of using LLMs for CVE classification. We also reveal common misclassifications made by humans and LLMs and analyze their reasoning errors to highlight sources of misclassifications.

Data collection. We collect all CVE entries from 2013 to 2024 through the NVD and CVE.or API. This results in 224,531 CVEs of which 211,182 are published. For each CVE entry, we extract the description, affected products/vendors from associated CPE entries, CWE id, and CVSS. Table 1 shows the statistics of the collected dataset. Note that since CVSS has evolved over time, a CVE can carry multiple versions of CVSS. To ensure consistency in our analysis, we select CVSS version 3.1 since it is the most prevalent version among all CVEs. Wherever this version is absent, we select the highest available version. As Table 1 shows, most of the selected CVEs (148,142) have CVSS V3.1. Among the remaining CVEs, 42,284 have CVSS V3 and only around 19,000 have CVSS version 2 and 4.

Dataset assessment. In the second phase, our goal is to determine whether LLMs can reliably classify IoT-specific vulnerabilities at scale and to benchmark the quality of the resulting labels against existing datasets. To this end (step 5), we apply the best-performing LLM from phase 1 to the entire CVE dataset from 2013 to 2024 to classify all CVEs at scale. It is of critical importance that labels given by the LLM are consistent and reliable, which means that running the same classification task multiple times with identical input, prompt, and parameters should provide the same labels. Therefore, to measure consistency, we randomly select n CVEs from the labeled CVE dataset

Table 1: Collected data statistics.

Type	Count	Type	Count
All CVEs	224,531	CVEs without CVSS	1,816
Published CVEs	211,182	CVEs with CVSS v3.1	148,142
Rejected CVEs	13,349	CVEs with CVSS v3.0	42,284
CVEs with CPE	211,182	CVEs with CVSS v2	18,267
VulnCheck KEV	4,005	CVEs with CVSS v4	673

and run the LLM classification m times over these selected CVEs. Then we measure consistency through the following metrics:

- Unanimity ratio (U_i). The unanimity ratio measures how often the model assigns the exact same label to a CVE across all repeated classification runs. We select n CVEs and run the classification model m times for each CVE. We mark a CVE as unanimous if all runs assign it the identical label. The unanimity ratio is the fraction of CVEs that meet this criterion.
- Maximum-class agreement (C_i). To further analyze consistency per CVE, we use another metric called maximum-class agreement, which measures the fraction of runs that align with that CVE's most frequent label (1.0 implies unanimity):

$$C_i = \frac{\max_c x_{i,c}}{m}. \quad (1)$$

where $x_{i,c}$ indicates the number of runs in which class c is assigned to CVE i across m runs.

We then compare our dataset with two previous datasets [40], [38] in terms of comprehensiveness and correctness through manual review. Finally, we report the expected error for the entire labeled dataset.

Vulnerability characterization. In this phase, the goal is to study how IoT vulnerabilities differ from traditional non-IoT vulnerabilities and provide actionable recommendations accordingly. We characterize IoT and non-IoT CVEs over their temporal pattern, changes in their severity score (using CVSS score), and the most frequent weakness families (using CWEs), shown as step 6 in Fig. 1. We then investigate the causes of vulnerabilities and propose mitigation techniques for responsible stakeholders.

4. IoT taxonomy and expert review

In this section, we explain the first phase of our study and answer the question of how IoT-specific vulnerabil-

Table 2: IoT-specific keywords and number of matched CVEs per IoT layer.

layer	References	Keywords	#Matches
Device	[54], [55], [56], [57], [58], [59]	Streaming device, Smart TV, Smart speakers, Smart lock, Tower fan, Smart plug, Smart light switch, Thermostat, Smart lamp, Smart doorbell, Smart security camera, Security camera, Smart camera, IP camera, Smart display, Home automation hub, Smart hub, Smart door lock, Smart dimmer switch, Smart outlet, Smart relay switch, Motion sensor, Smart toy, Smart meter, SmartThings hub, Amazon echo.	358
Communication	[60], [61]	ZigBee, BLE, 6LoWPAN, LoRaWAN, MQTT, CoAP, RPL (Routing protocol for low-power and lossy network)	357
Application	[5], [62], [20]	SmartThings, IFTTT, Home Assistant, Alexa, OpenHAB, Domoticz, Apple Homekit/Homekit, Google assist, Google Nest	106
Cloud endpoints	[63]	AWS IoT, Azure IoT, Google Cloud IoT, IBM Watson IoT	30

ities can be systematically distinguished from non-IoT vulnerabilities.

4.1. IoT ecosystem definition

We define the IoT ecosystem as comprising the following four layers:

- *Device layer*: Physical devices that often have internet connectivity, limited power, memory, computational capacity and either collect sensory data or perform actuation tasks. These devices also interact with other devices directly or indirectly using communication technologies with limited or no human intervention [64]. Examples include smart thermostats, Amazon Echo, motion sensors, smart cameras, and smart TVs. *Typical vulnerabilities* of this layer originate from the hardware, firmware, or embedded OSes/software (e.g., FreeRTOS, TinyOS, RIOT-OS).
- *Communication layer*: This layer includes any protocol that facilitates data transfer and connectivity for IoT devices within the local network or outside of the network. It involves specialized low-power protocols (e.g., MQTT, BLE, ZigBee). *Typical vulnerabilities* of this layer include flaws in the protocol design, implementation or configuration bugs in protocol stacks (e.g., unencrypted data exchange, lack of authentication).
- *Application layer*: This layer includes mobile applications and orchestration systems/local controllers that sit above the hardware and enable user-space services for IoT. Examples include IoT-focused platforms and applications/companion applications (e.g., Home Assistant, SmartThings, IFTTT). *Typical vulnerabilities* include flaws in the design or implementation logic of these platforms and applications.
- *Cloud layer*: Remote or public cloud endpoints that provide IoT-specific services, such as device management, remote control, and other IoT-focused functionalities. Examples include AWS IoT, Azure IoT, Google Cloud IoT, IBM Watson IoT. *Typical vulnerabilities* of this layer include flaws in the configuration of the infrastructure, vulnerable services, insecure cloud APIs or tokens, and improper input validation on the server side.

Given the above definition of the IoT ecosystem, we define the following classification rules:

- IoT-specific CVE: Vulnerabilities that affect technologies belonging to the IoT ecosystem, where these technologies are purpose-built for IoT and the deployment context is also IoT. For example, ZigBee and LoRaWAN for communication protocols, or platforms like SmartThings, OpenHAB, and AWS IoT for application and cloud layers. Moreover, the vulnerable product is itself an active, functional element of an IoT system.
- Non-IoT CVE: Vulnerabilities that do not fit within the above definition and affect technologies outside the defined IoT ecosystem. General-purpose hardware/software such as smartphones, computers, or tablets are considered as Non-IoT. Network infrastructures such as routers, modems are also Non-IoT.

Note that we exclude network devices from IoT. The device layer refers specifically to end-user or embedded systems designed for sensing, actuation, or control in physical environments, not generic infrastructure such as routers or switches. Network devices (e.g., enterprise routers, switches) are typically part of traditional IT infrastructure unless they are part of an IoT-specific deployment (e.g., IoT gateways, edge routers in smart homes or industrial setups). Moreover, some technologies originally designed for IoT may also appear in non-IoT contexts. For example, MQTT is widely used in IoT but can also run on servers in the non-IoT context. Thus, the presence of IoT-related technologies in a vulnerability does not ensure that the flaw is IoT-specific. For example, CVE-2019-9749 involves a Fluent Bit MQTT plugin running on servers, where a malformed packet can crash the host. Despite involving MQTT, the flaw impacts server deployments rather than IoT devices specifically. Conversely, technologies not designed for IoT, such as OpenSSL, may still be IoT-relevant if the vulnerability affects IoT deployments. Therefore, classification cannot rely solely on keyword matching. Instead, contextual understanding is necessary to determine the scope and relevance of each CVE to IoT security. Also, it is worth to mention that many technologies are common in both IoT and non-IoT context, such as WiFi, Linux distributions to different extents, BusyBox, OpenSSL. However, this surface is broad, and in this work we focus only on the IoT-specific technologies.

4.2. Ground-truth dataset creation

In this section, we explain the construction of the ground-truth dataset through keyword filtering followed by human review. Since manual review is time-intensive and prone to errors, we focus on a set of 300 CVEs for detailed manual analysis to ensure data quality. This set consists of 150 IoT-specific and 150 non-IoT CVEs.

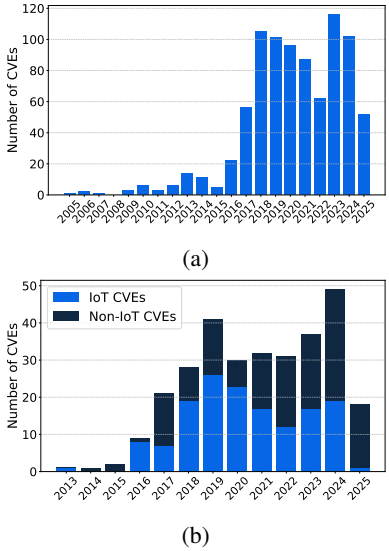
CVE filtering. To build the IoT CVE subset, following prior studies [40], [41], we develop a dictionary of IoT-specific keywords and use it to search CVE descriptions

to filter potential IoT CVEs. To ensure coverage of all four layers of the IoT ecosystem, we create four separate keyword dictionaries, one for each IoT layer defined in Section 4.1. The complete keyword list is presented in Table 2, which is obtained through a review of existing literature. Using these dictionaries, we filter 851 unique IoT CVEs. Among the filtered CVEs, 358 are associated with the device layer, 357 with the communication layer, 106 with the application layer, and 30 with the cloud layer. We then randomly sampled an equal number from each layer to form the 150 IoT-specific CVEs. The 150 non-IoT CVEs are randomly sampled from the rest of the CVEs that were not filtered by our dictionaries.

The temporal distribution of IoT-specific CVEs extracted using the four keyword dictionaries is shown in Fig. 2a. While the earliest filtered IoT CVE dates back to 2005, the number of identified IoT CVEs remains low until 2013. This is expected since IoT was not prevalent during that period. Note that beside the increasing prevalence of IoT over time after 2013, the number of total CVEs has also increased substantially each year. Fig. 2b presents the temporal distribution of the 300 CVEs selected for manual review. This subset was sampled to reflect the same temporal pattern from 2013.

Manual labeling. To validate the 300 selected CVEs, we asked nine reviewers with expertise or familiarity in cybersecurity to manually review CVEs. The reviewers include one security professor, one security expert, and seven PhDs (five in IoT security, cybersecurity, and Internet measurement; two in quantum cryptography for IoT and web privacy). The experts were divided into three groups of three, and each group reviewed 100 CVEs (50 IoT-specific and 50 non-IoT), ensuring that every CVE was assessed by three independent reviewers. Reviewers received the classification guideline, including the IoT ecosystem definition and classification rules from Section 4.1, and were asked to label each CVE as IoT-specific or non-IoT and identify the IoT layer responsible for the vulnerability. To avoid uncertain decisions, we provided an *unsure* option, allowing reviewers to indicate when they could not confidently classify a CVE. This option reduces chance agreement and enables further analysis to improve dataset quality. Also, we asked the reviewers to provide a brief justification for their decision. Moreover, a dedicated web interface was developed to facilitate the review process, displaying each CVE's description and CPEs with a direct link to its NVD page (Appendix A).

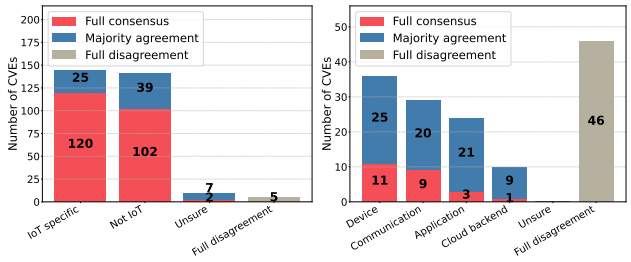
Manual labeling outcomes. Fig. 3 shows the result of manual review. Each CVE label was determined by majority vote among the three reviewers. As shown in Fig. 3a, reviewers identified 146 IoT CVEs, 140 non-IoT CVEs, 9 unsure cases, and 5 CVEs with three different labels. Notably, we observe a large fraction of full consensus for each class of IoT and non-IoT, indicating that all three reviewers assigned the same label in most cases. Only for 14 CVEs (almost 4%) reviewers were mostly unsure or in full disagreement (i.e., three different labels). We noticed that these cases relate to technologies that are borderline and are widely used in both IoT and non-IoT, e.g., the WolfSSL library (a lightweight SSL/TLS library used across resource-constrained devices, desktops, enterprise systems, and cloud services), HarmonyOS (deployed on both smartphones and certain IoT devices), Fast DDS,



(a)

(b)

Figure 2: Temporal distribution of CVEs: (a) CVEs filtered out by all four dictionaries (total: 851), and (b) selected CVEs for manual review (total: 300).



(a) IoT classification (b) Layer-level classification

Figure 3: Reviewers' votes for classification label and vulnerable layer for the selected subset of CVEs.

and BusyBox. We also calculate the inter-rater agreement, measured by Krippendorff's α, which was 0.80, showing a satisfactory level of agreement among the reviewers for this task. Note that unsure values were treated as missing values rather than a separate category as they do not indicate a preference for one label over another and do not constitute a meaningful classification on their own.

Moving on to layer-level classification, we seek an answer to the following question: *Given our taxonomy of four-layered IoT ecosystem definition, how are the vulnerabilities distributed among these layers?* The motivation is to understand if there is a layer that is the weakest in terms of security attributes and determine the actors who should take an action, e.g., whether it is the vendors of those devices or the software engineers developing the app in the cloud backend. However, layer-level classification proved to be challenging. As Fig. 3b shows, full consensus among the three reviewers occurred less frequently than in the IoT classification task. Furthermore, the proportion of cases with complete disagreement, almost 14%, was significantly higher, indicating the difficulty in identifying the exact vulnerable layer for reviewers. The Krippendorff α was 0.28 for layer-level classification, indicating poor agreement. These observations indicate the difficulty of layer-level classification. While determining whether a CVE is IoT-specific is relatively straightforward given context and keyword cues, accurately identifying the vul-

nerable layer is challenging. Several factors contribute to this challenge. First, CVE descriptions often lack sufficient technical detail, forcing reviewers to read external resources such as proof-of-concept exploits to understand the vulnerability's exact location, which is both time-consuming and inconsistent. Second, reviewers may not have comprehensive expertise, especially given the diversity and complexity of IoT architectures. Third, certain vulnerabilities inherently straddle multiple layers, making the precise classification ambiguous. These limitations highlight the need for more nuanced definitions of IoT layers, standardized annotation guidelines, technical detail or enriched metadata within vulnerability databases to support accurate and consistent layer-level classification. Therefore, given the poor agreement level for this task, we cannot accurately identify the layers. We do not rely on the layer-level labels derived from this review and identify improved layer-level attribution as future work.

Eventually, to resolve cases where CVEs were marked as unsure or had full disagreement between reviewers, we consulted an external expert (a full professor with expertise in cybersecurity) to serve as a tie-breaker. This expert reviewed the original labels and the accompanying justifications given by the reviewers to determine the final classification. As a result, every CVE in the manually reviewed set received a single agreed-upon label. This resulted in 148 IoT and 152 non-IoT CVEs.

4.3. LLM validation on the ground-truth dataset

In this section, we evaluate state-of-the-art LLMs on the CVE classification task. We prompt them to classify each CVE as IoT or non-IoT, to identify the vulnerable layer, and to provide a brief rationale. The prompt given to LLMs includes the same classification guideline as given to human reviewers (Appendix B).

LLM performance validation. Table 3, shows the classification result on the manually labeled dataset. As can be seen, commercial models (GPT-4o and O3) outperformed open-source models (Deepseek-r1 and Llama4-scout). GPT-4o achieved the highest overall F1 score of 95.3%, closely followed by O3 (94.59%), demonstrating their effectiveness in distinguishing IoT from non-IoT vulnerabilities. Deepseek-r1 showed lower F1 score of 92.76% and Llama4-scout had the poorest performance (89.94%). The Krippendorff α calculated for the four LLMs as four independent raters, shows a high α value of 0.87 for IoT classification, exceeding the human agreement of 0.80 on the same task. These findings suggest that large language models exhibit high classification performance in IoT CVE labeling. For the layer-level classification, the LLMs achieved an agreement level of 0.42, which is higher than the human score of 0.28. However, as noted earlier, the overall low human agreement on layer-level labels prevents us from reliably evaluating LLM accuracy on this task and the low resulting score of 0.42 of LLMs, once again underscores the inherent difficulty of this task.

Classification errors in humans and LLM. When comparing human and LLM reasoning, we observed distinct strengths and weaknesses in their decision-making processes. In some cases, human reviewers were better able to consider domain-specific context. For example, CVE-2018-19628 describes a vulnerability in Wireshark's ZigBee ZCL dissector. Although the term Zigbee is associated with IoT, human reviewers correctly identified this CVE as non-IoT. While most LLMs could classify this CVE correctly, GPT-4o marked it as IoT since the ZigBee component caused the vulnerability. With a small rephrasing of the classification rule for IoT-specific, GPT-4o could resolve this error. On the other hand, human reviewers were prone to cognitive errors, in which reviewers confuse terminology and immediately come to a the final decision without fully resolving the contextual meaning. For example, CVE-2016-6769 contains the phrase "smart lock", which led all three reviewers to incorrectly label it as IoT-related. However, in this case, "smart lock" refers to an Android software feature used to manage screen unlock settings, rather than a physical smart lock device. Other errors stemmed from differences in technical background. For example, CVE-2022-33211 describes a memory corruption vulnerability in Qualcomm's cellular modem firmware. The word "modem" here refers to the cellular baseband processor inside phones, wearables, or IoT modules, which some human reviewers confused with cable modems and tagged as non-IoT. Such misunderstandings arise from differences in technical background and lack of domain knowledge, which are difficult to avoid given the broad range of technologies represented in CVEs. In contrast, LLMs could correctly identify these cases. Some misclassifications relate to subjective inference about devices that are not widely recognized as IoT but could be considered as IoT, such as GoPro cameras. Also, for several CVEs such as CVE-2024-25507, a SQL injection vulnerability related to RuvarOA (a web-based office automation service), human marked as unsure and noted that not only the description is not informative, but also they could not find anything online to understand what the mentioned technology and vulnerability is. This yields limits of human domain knowledge, especially when faced with not widely recognized or poorly documented technologies. However, LLMs could provide satisfactory information and classify it correctly, likely due to their broad training on enormous and diverse corpora.

Human-LLM disagreement resolution. After manually reviewing discrepancies between LLM and human labels and identifying several human errors, we revisited CVEs misclassified by an LLM compared to the initial human annotations to further validate label quality and re-

Table 3: Performance comparison of different LLMs on IoT vs. non-IoT CVE classification.

Model	Accuracy	Precision	Recall	F1 Score
O3	94.67	93.96	95.24	94.59
GPT-4o	**95.33**	**94.04**	**96.60**	**95.30**
Deepseek-r1-70B	92.67	89.81	95.92	92.76
Llama4-scout	89.33	83.63	97.28	89.94

Table 4: Updated classification performance after expert review and correction of disputed CVE labels.

Method	Accuracy	Precision	Recall	F1 Score
Human Reviewers	97.58	98.6	97.24	97.92
O3	**99.31**	**100**	**99.31**	**99.65**
GPT-4O	97.58	97.28	98.62	97.95
Deepseek-r1-70B	96.19	94.7	98.62	96.62
Llama4-scout	92.39	87.8	99.31	93.2

evaluate classification performance. Overall, 44 samples were flagged for manual inspection. We ask a tie-breaker security expert to review these CVEs along with the reasoning provided by both human reviewers and LLMs. Based on this additional review, we updated the final label assignments for these cases and incorporated the corrections into the ground-truth dataset. It is worth to note that some cases remained genuinely borderline or context-dependent, e,g, WolfSSL and BusyBox. Because these could not be confidently assigned to either class, we excluded them (overall 10 such samples out of 300 CVEs) for our evaluation. We then recalculated the accuracy for LLMs and also calculated human reviewers' accuracy based on their initial annotations. Table. 4 shows the results. As we see, all LLMs had an increase in their performance compared to Table 3 due to previous incorrect annotations by human reviewers. O3 outperformed all other LLMs with a high F1 score of 99.65%, while the performance of human reviewers reaches 97.92%.

Key insight: Considering the massive number of CVEs, not only is manual review impractical, our inspection revealed cognitive errors, subjective variances, background differences and limited domain knowledge of reviewers. LLMs on the other hand, showed resilience to cognitive errors, filled the gap in human domain knowledge regardless of poorly documented CVEs, and achieved 99.31% accuracy on the ground-truth dataset (after resolving labeling errors), enabling timely, large-scale systematic labeling.

5. Large-scale classification of CVEs

Building on the strong performance of LLMs observed in the previous section, in this section, we scale up LLM classification to explore whether LLMs can reliably classify IoT-specific vulnerabilities at a larger scale and benchmark the resulting dataset. Since O3 obtained the best result among all LLMs, we use O3 to classify all CVEs from 2013 to 2024. In total, it took 10 hours[3] to classify all CVEs with the cost of 760[4] Euros.

Consistency analysis. To evaluate labeling consistency after labeling all CVEs, we randomly select 1,000 CVEs (500 IoT and 500 non-IoT) and run the same model (O3) with the same prompt over this set 10 times.

The mean unanimity rate ($\bar{U}$) is 92%, proving that for the majority of selected CVEs, the LLM produces the same label across all runs. We calculate this metric separately for the IoT and non-IoT subsets, which yields a score of 86% and 98%, respectively. This substantial difference indicates that disagreement is concentrated in the IoT sector and shows that the LLM model is less certain for labeling IoT samples, while non-IoT samples are labeled with a very high confidence. The 95% bootstrap confidence interval for this metric is reported in Table 5. Then, to better understand the consistency level for individual CVEs, we calculate the mean maximum-class agreement ($\bar{C}$) across all C_i. We obtain $\bar{C} = 0.98$ overall,

3. Note that the time varies significantly based on API rate limits and the degree of request concurrency.

4. Note that the cost could be reduced via techniques such as the batch API.

Table 5: Mean unanimity rate $\bar{U}$ and mean maximum-class agreement $\bar{C}$ with 95% bootstrap CIs for the LLM classification task.

	$\bar{U}$	95% bootstrap CI	$\bar{C}$	95% bootstrap CI	n
Overall	0.92	[0.90, 0.94]	0.98	[0.97, 0.98]	1000
IoT	0.86	[0.83, 0.88]	0.96	[0.95, 0.97]	500
Non-IoT	0.98	[0.97, 0.99]	0.99	[0.995, 0.999]	500

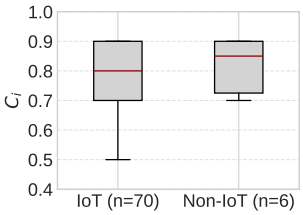


Figure 4: Distribution of maximum-class agreement (C_i) for non-unanimous samples in the IoT and non-IoT subsets.

and $\bar{C} = 0.96$ for IoT vs. $\bar{C} = 0.99$ for non-IoT, showing a high level of per-sample consistency and confirming that even when labels are not unanimous, they are typically near-consensus. The corresponding 95% bootstrap CI is shown in Table 5. A box plot of C_i for non-unanimous items for the IoT and non-IoT subsets, Fig. 4 further visualizes the disagreement distribution. Although inconsistency is more prevalent in IoT samples, the C_i values lie mainly between 0.7 and 0.9, which shows a near consensus. These results indicate high labeling stability under a fixed prompt.

We also treat each of the ten runs as an independent rater and compute Krippendorff's α. We obtain $\alpha = 0.94$, indicating high agreement overall. Note that we do not compute Krippendorff's α separately for IoT and non-IoT because each subset is dominated by a single label, making α undefined or uninformative.

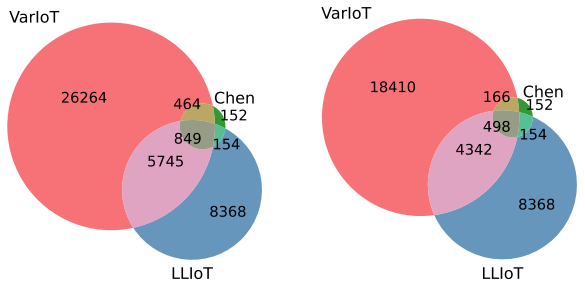
This observation led us to manually analyze non-unanimous samples. We noticed that most of these cases are related to technologies that are hard to classify as either IoT or non-IoT and are genuinely borderline, such as WolfSSL, chipsets, industrial controllers, TLS, and industrial gateways. These technologies are used in both IoT and non-IoT context. For these cases, assigning a deterministic IoT/non-IoT label is not straightforward, and it depends on the intended definition of IoT and its use case. However, outside of this boundary region, the LLM is highly consistent. We emphasize that consistency assesses stability, not correctness. We will evaluate the correctness of the dataset in the next part.

Key insight: The consistency analysis of LLMs over 1,000 CVEs, exhibited strong agreement, with a 92% unanimity rate and 98% maximum-class agreement. However, LLMs suffer from inconsistency in labeling borderline cases.

Comparison to other datasets. To assess the comprehensiveness of our dataset, we compare it with two public datasets of IoT CVEs, VarIoT [38] and Chen et al. [40], over the same period (2013 till the end of 2024). Both works provide a limited and somewhat different definition of IoT for labeling CVEs, which in turn affects their

Table 6: Number of IoT CVEs in previous studies, distinguished by category.

Method	Network	Industrial	Home&Office	Camera	Embedded	Vehicle	Wearable	Medical	Other	Total
VarIoT	9,871	2,477	131	103	97	32	8	5	20,598	33,322
LLIoT coverage	1735 (17.5%)	1950 (78.7%)	122 (93.1%)	93 (90.2%)	68 (70.1%)	19 (59.3%)	7 (87.5%)	5 (100%)	2,595 (12.5%)	6,594 (19.7%)
Chen et al. [40]	350	36	191	275	92	36	14	7	618	1,619
LLIoT coverage	90 (25.7%)	33 (91.6%)	175 (91.6%)	238 (86.5%)	78 (84.8%)	23 (63.9%)	13 (92.9%)	7 (100%)	346 (55.9%)	1,003 (61.9%)



(a) Including network devices. (b) Excluding network devices.

Figure 5: Overlap of IoT CVEs among the three approaches.

classification results. VarIoT focuses primarily on vulnerabilities related only to IoT devices and uses filtering at multiple levels to identify IoT CVEs. In contrast. Chen et al. [40] define IoT vulnerabilities as issues in IoT devices, networks, and applications, which is more closely aligned with our definition. They follow keyword-based filtering followed by human review to identify IoT CVEs. Note that while our dataset excludes network devices, the other two studies classify network devices as IoT.

Fig. 5a shows the overlap of IoT CVEs in the three datasets. In total, 849 CVEs are common to all three datasets, and 7364 CVEs are marked as IoT by at least two datasets. Our proposed method identifies more than 8,000 IoT vulnerabilities that are not reported by either prior works. Chen et al. identify only 152 CVEs not found by the others. The VarIoT dataset reports approximately 33,000 IoT vulnerabilities in total, about 26,000 of which are unique to VarIoT. Much of this apparent excess is explained by VarIoT's broader inclusion of network devices (around 10,000). Chen's dataset likewise includes a large portion of network device vulnerabilities (350 CVEs). Therefore, to have a fair comparison, Fig.5b shows the overlap of CVEs after excluding network-device CVEs (as tagged in VarIoT and Chen) from all three datasets. Under this adjustment, the disagreement with Chen shrinks to 318 CVEs, and the gap with VarIoT decreases to around 18,500. However, even with excluding the network-device vulnerabilities the gap with VarIoT remains substantial.

To better compare these datasets and uncover the reason of the gap with VarIoT and Chen datasets, in Table 6 we show the number of CVEs per category (provided in their dataset) and also show the coverage of our dataset within the same categories. In both datasets network devices dominate other categories. While we exclude general network devices from our IoT-specific scope, a small number of entries do appear there, which mostly stem from industrial IoT gateways. Across the remaining categories, our dataset covers the majority of vulnerabilities, and where coverage is low, the category itself is small. Notably, the gap between our dataset and the other two datasets, particularly with VarIoT, is attributed to the "other" category, including around 20,500 CVEs in VarIoT and around 600 in Chen's dataset, raising concerns about labeling consistency.

Therefore, we validate the quality of the labels in the three datasets through manual analysis. For each dataset, we randomly select 100 CVEs from those uniquely identified by that dataset and not reported by the others (e.g., for LLIoT, we sample 100 CVEs from the 8,368 CVEs not reported by either prior work). This yields a conservative analysis, since CVEs labeled as IoT by at least two datasets are highly likely to be genuinely IoT-related.

Our manual review of VarIoT's CVEs found 43 involving network devices, 6 involving chipsets, and 15 involving operating systems that can appear in some IoT deployments. Under the broader scope of VarIoT and their definition, these could be considered IoT. However, the remaining 36 CVEs were clearly non-IoT, affecting products such as Google Chrome, WordPress plugins, Perl, Adobe Flash Player, macOS, Safari, and various smartphone applications. This suggests that a significant portion of VarIoT's "other" category is still related to network devices and a notable portion is misclassified as IoT, which explains the gap between our dataset and theirs. For the Chen dataset, our manual analysis revealed 13 related to routers, and 59 related to chipsets or various drivers in Android and Linux. Since Android's major usecase is primarily for smartphones, they are excluded in our dataset but they align with Chen's broader scope. The remaining 28 CVEs were non-IoT, including issues in Windows, wired speakers, desktop and non-IoT web apps. Therefore, the main reason for disagreement with Chen's dataset is mainly due to their broader scope of IoT CVEs and some degree of misclassification. The list of samples reviewed for both datasets is shown in Appendix C. VarIoT and Chen rely heavily on keyword based filtering. Many CVEs match the keywords but in non-IoT contexts, e.g. CVE-2020-25282, CVE-2017-0430, which causes a high false positive rate. Although Chen's dataset is accompanied by manual inspection (1,600 samples), manual analysis still shows false positives potentially due to the error-proneness of time-consuming and exhausting manual review at scale.

The manual inspection on our dataset revealed a diverse and largely IoT-specific set of vulnerabilities, including 22 vulnerabilities related to various IoT devices (cameras, infusion pump, smart lock, printer, drone, smart TV, wearables), 5 related to OSes specific to IoT devices (RTOS, OpenHarmony), 20 related to industrial firmware/controller, 9 related to chipsets, 3 EV charging stations, 10 industrial/IoT/OT software & platforms, 4 industrial controllers/RTUs, 9 industrial communication libraries/protocol stacks (e.g., CoAP), 3 building automation, 1 harmonyOS, 2 digital signage systems, 4 consumer IoT devices, 4 embedded JS engines/IoT SDK. 4 CVEs

are more IoT-related than being IoT-specific, involving PLC engineering software, robotics simulation software, and web management software. The reviewed samples and more details can be found in Appendix C. Some CVEs may be more IoT-related than strictly IoT-specific. They may not directly affect an active component of an IoT system or may also impact non-IoT environments, yet they maintain clear contextual relevance to IoT. It is worth to mention that unlike prior studies, which often include clearly unrelated CVEs (e.g., browser, desktop, or smartphone vulnerabilities) under the IoT category, our dataset consistently includes only those vulnerabilities that are either IoT-specific or at minimum closely related to IoT technologies.

We also provide the expected error in the whole population of 211,182 labeled CVEs using a 95% Wilson confidence interval based on the O3 model's accuracy observed on a subset of 290 samples (99.3% accuracy, 288/290 correct, Table 4). Modeling predictions as Bernoulli trials (correct vs. incorrect), for k = 288 correct out of n = 290, the 95% Wilson confidence interval places the accuracy at [97.52%, 99.81%], and the expected error rate between 0.19% and 2.48% over 211,182 CVEs.

Key insight: We identified 15,116 IoT-specific CVEs, including an additional 8,368 unique IoT CVEs not discovered by previous datasets. While classification of CVEs as IoT highly depends on what is presumed as IoT, our investigation revealed considerable numbers of false positives in prior datasets, around 30% false positive in the reviewed subset. To the best of our knowledge, our dataset represents the most comprehensive and reliable collection of IoT-specific CVEs to date.

6. IoT vs. non-IoT characteristics

After labeling the whole dataset, we characterize IoT and non-IoT vulnerabilities to understand the state of IoT-specific vulnerabilities. More specifically, we examine: How do IoT and non-IoT vulnerabilities evolve over time? How do they differ in severity? Which weaknesses are most prevalent in each class? And to what extent do IoT and non-IoT CVEs have public exploits?

6.1. IoT vs. non IoT temporal trends

The resulting labeled dataset of our approach identifies 15,116 IoT CVEs and 196,066 non-IoT CVEs in total. Fig. 6 shows a histogram of IoT and non-IoT CVEs from 2013 to 2024 using bar plots, along with the relative increase in each year compared to 2013, shown as line plots (right axis). As expected, non-IoT dominates in absolute counts throughout the period as a result of a larger number and variety of non-IoT technologies and devices. However, the IoT sector is far smaller, resulting in far fewer vulnerabilities, starting from 300 CVEs in 2013 and rising to 2,200 in 2024. Although the number of non-IoT CVEs is drastically higher, around 35,000 in 2024, and both sectors trend upward, the relative growth of IoT CVEs is steeper. By the end of 2024, the number of IoT CVEs is nearly 8.5× its 2013 count, compared to approximately 6× for the non-IoT.

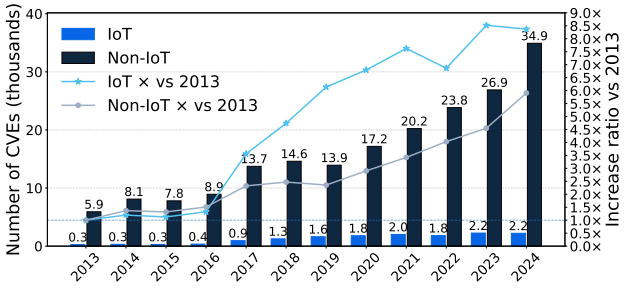


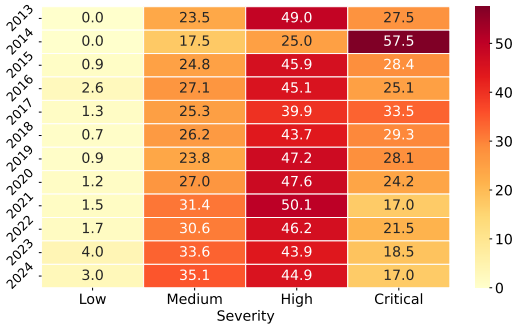
Figure 6: Number of IoT and non-IoT CVEs from 2013 to 2024 along with the increase ratio relative to 2013.

While from 2013 to 2016 the number of IoT CVEs remains relatively stable, after 2016, it experiences a steep upward trend, with the sharpest increase occurring in 2017, which is more than 3× relative to 2013 and more than double compared to its previous year. We speculate that this rise is due to various reasons; first, 2016 was the beginning of the spread of world-wide IoT-tailored malware, starting with the Mirai botnet [65] that launched record-breaking DDoS attacks. The release of Mirai source code caused subsequent variants and a surge in attacks on IoT, e.g., Satori [66], Hajime [67], Reaper [68], and Dark Nexus [69]. Therefore, we speculate that the discovery of such botnets shifted the attention towards IoT and motivated more extensive scanning for IoT vulnerabilities. Although earlier IoT botnets existed, they were less disruptive [70]–[72]. Second, in 2016, the CVE program began actively expanding the number of organizations participating as CVE Numbering Authorities (CNAs) [73]. The expanding CNA network increased CVE issuance volume, which contributes to this rise.

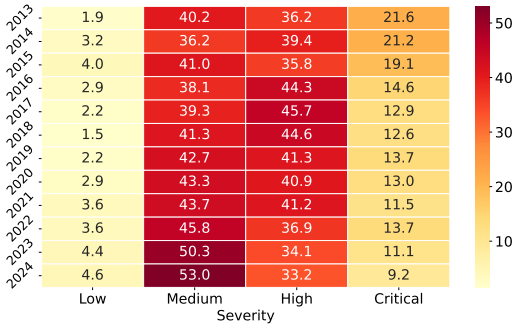
While non-IoT vulnerabilities show a sustained rise since 2013, they also experience a sharp increase in 2017, which may similarly be linked to the CNA expansion. Another significant observation relates to the contrast between the trend of IoT and non-IoT in the last two years. While the number of non-IoT CVEs surges from 26,000 to nearly 35,000, from 2023 to 2024, IoT CVEs remain stable at around 2,200. A likely reason could be related to the number of vendors that remained nearly constant in the IoT sector (around 900) while it rose from 7,500 to almost 10,000 in the non-IoT sector.

6.2. Severity score analysis

Figs 7a and 7b show the yearly distribution of CVSS severity levels for IoT and non-IoT CVEs. We selected CVSS version 3.1 when available, otherwise the highest available version is chosen. In IoT, the high and critical share together decrease from 76% in 2013 to 62% in 2024. This decrease is mainly due to the decrease in the critical share, while the high level share consistently averages around 45% in each year and always dominates throughout almost all years. Between 2013 and 2019, critical IoT CVEs often exceed medium, but this trend reverses from 2020 onward, and the critical share, despite fluctuations, trends downward. Meanwhile, the medium share increases and reaches 35% in 2024. Non-IoT CVEs follow a similar downward trend in high and critical share combined, decreasing from around 60% in 2013 to 42% in 2024 with both share decreasing. Notably,



(a) IoT CVEs.



(b) Non-IoT CVEs.

Figure 7: CVSS over years for IoT and non-IoT CVEs.

since 2019, the medium-severity shows a steady increase and consistently exceeds the high-severity in prevalence, reaching a 53% share in 2024. This observation could be attributed to the discovery of more vulnerabilities as a result of more scanning tools and attempts, resulting in the discovery of more mid-severity vulnerabilities. Overall, IoT shows higher risk scores, highlighting their potential higher risk compared to non-IoT vulnerabilities. Although both groups show a general move toward more medium-severity cases, IoT retains a substantially larger fraction of high and critical vulnerabilities, underscoring the need for stronger mitigation strategies in IoT systems.

Given higher CVSS scores in IoT CVEs, we further analyze which components of CVSS differ the most between IoT and non-IoT. Fig. 8 shows the percentage-point difference in the distribution of CVSS components' values between IoT and non-IoT CVEs. The largest gap appears in the User Interaction (UI) component, where a significantly higher portion of IoT CVEs require no user interaction (+20.3 pp), while the "Required" category appears more often in non-IoT CVE (-20.3 pp). This indicates that IoT vulnerabilities are more often autonomous and remotely exploitable. It also corroborates the lack of security mechanisms in IoT compared to non-IoT. Moreover, in the Privileges Required (PR) component, IoT CVEs are more likely to require no privileges (+12 pp), while non-IoT CVEs skew toward low or high privilege requirements. These two metrics, PR and UI, contribute significantly to the CVSS score, which partially justifies the higher CVSS score in IoT compared to non-IoT. For impact metrics, IoT CVEs show higher proportions of high confidentiality (C), integrity (I), and availability (A) impact. In fact, due to lack of security mechanisms, once an attacker gets into the system, they can often take full control of the device and its data.

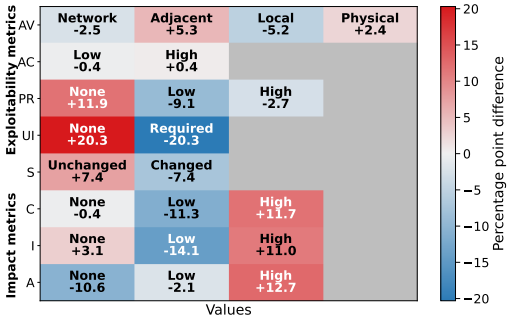


Figure 8: Value difference (IoT% - non-IoT%) of CVSS base score components between IoT and non-IoT CVEs (red = IoT higher, blue = non-IoT higher).

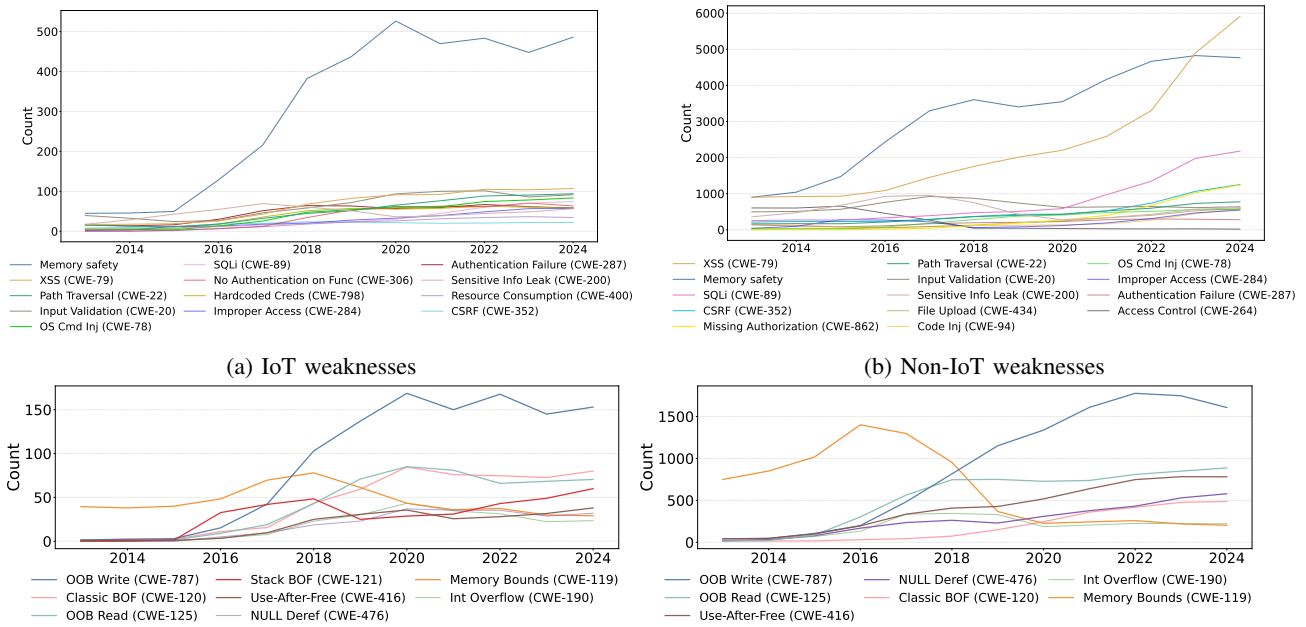
Key insight: The severity of vulnerabilities in the non-IoT sector has been shifting from high and critical towards medium level, with medium-level accounting for 53% of all vulnerabilities in 2024. While IoT vulnerabilities follow the same trend, they are still far behind. In 2024, the medium level severity in the IoT sector covers only 35% of vulnerabilities, while high and medium levels comprise 62% of the total. Overall, IoT CVEs show a higher severity score compared to the non-IoT sector. Also, IoT vulnerabilities tend to be lower-effort, higher-impact, and more exploitable with less user involvement, all contributing to their elevated severity score.

6.3. Analysis on the weaknesses

Each CVE may be associated with one or more CWE(s) that determine the weakness family. Fig. 9a and Fig. 9b show the counts of the most common weaknesses in each year in IoT and non-IoT, respectively, which helps to see the trend of each weakness family over the years.

CWEs are organized hierarchically, determining weaknesses more precisely by going from high-level weaknesses in root nodes (e.g., improper access control) to more specific ones (e.g., improper authentication) in leaf nodes. For visualization, we aggregate memory safety issues that appear in our top-20 list into a single "Memory safety" bucket: Out-of-Bounds write (CWE-787), Out-of-Bounds read (CWE-125), Buffer copy without checking size of input (CWE-120), Stack-based buffer overflow (CWE-121), Use after free (CWE-416), Integer overflow (CWE-190), NULL dereference (CWE-476) and Memory-bound (CWE-119). While this aggregation introduces an imbalance in granularity level between memory safety and other weaknesses, we acknowledge that our analysis remains unchanged. We expand the memory safety weaknesses in Fig. 9c and Fig. 9d, respectively. In fact, within the memory safety bucket, out-of-bound writing is the most prevalent weakness, ranking first and second among the top-20 weaknesses in IoT and non-IoT sectors.

As seen in Fig. 9a. in the IoT sector, memory-safety issues are the dominant weaknesses by far, surging significantly from 2016 onward. Among memory safety issues, out-of-bound write is the most prevalent by far. This weakness is common, largely due to the widespread usage of the C language in the IoT firmware, which is



(a) IoT weaknesses

(b) Non-IoT weaknesses

(c) IoT memory safety weaknesses

(d) Non-IoT memory safety weaknesses

Figure 9: (a) and (b) shows the most common weaknesses in IoT and non-IoT CVEs, respectively. (c) and (d) expands the aggregated memory safety weaknesses in IoT and non-IoT, respectively.

highly susceptible to such vulnerabilities [74], [75] and also poor memory protection mechanisms. Next in prevalence are input validation (CWE-20), XSS (CWE-79), and path traversal (CWE-22). Input validation weaknesses occur when software accepts data without enforcing the expected properties. This weakness can lead to a wide variety of more specific weaknesses, e.g., SQL injection (CWE-89), XSS, OS command injection (CWE-78). XSS is inherently web-based. Path traversal is also predominantly seen in web (but it also arises in file-handling services or device APIs that assemble filesystem paths from user input). Taken together, the appearance of these weaknesses among the top-20 weaknesses could reflect the growing use of web-based interfaces in IoT devices to provide user interaction and remote management capabilities. Observing other related weaknesses such as SQLi (CWE-89) and CSRF (CWE-352) in the top-20 weaknesses, which are mostly web-related as well, further contribute to this trend. In contrast, in the non-IoT sector, XSS shows the steepest rise and becomes the single most prevalent weakness from 2023 onward, affecting around 6,000 CVEs in 2024. Together with upward trends in SQLi, CSRF and path traversal, this makes web-based vulnerabilities the most critical issue in the non-IoT sector and indicates both the ubiquity of web applications and the impact of large-scale automated discovery (scanning tools, bug bounties). Next, memory safety is the most critical class of weaknesses. Among them, the out-of-bound write (CWE-787) ranks highest and is the second most frequently observed weakness overall, after XSS.

The next dominant class of weaknesses in the IoT sector relate to authentication issues, which include improper authentication (CWE-287), no authentication on critical functions (CWE-306), and improper access control. Authentication issues are far less common in the non-IoT top 20 CWEs. Instead, the non-IoT sector involves missing authorization (CWE-862), even showing a rising trend, while it is absent from the IoT top 20. This is because non-IoT systems often involve multi-tenant architectures and fine-grained role-based access controls, which makes them more susceptible to missing authorization vulnerabilities. IoT systems, on the other hand, tend to lack this granularity, often supporting only a single administrative role. Furthermore, non-IoT systems benefit from abundant computational resources, allowing them to implement complex and flexible authentication mechanisms. In contrast, IoT systems often employ simplified or even missing authentication and access control. This is mostly influenced by design practices focused on quick deployment, neglecting security practices, and by resource constraints (CPU, memory, power) that limit the feasibility of stronger controls.

The Sensitive Information Leakage weakness (CWE-200) appears in both IoT and non-IoT CVEs and shows a declining trend in each. A likely reason is changes in the assignment of CWEs. As noted, CWE uses a hierarchical tree, and broad umbrella categories (e.g., Memory Bounds, Input Validation) are discouraged in favor of more specific children such as Out-of-Bounds Write/Read or Use-After-Free, which better indicate the actual weakness. Therefore, as labeling practices matured, this weakness is less frequently assigned, which could be the potential reason for the observed decrease. Moreover, both sectors show an increasing trend for OS command injection (CWE-78), which ranks higher in IoT than in the non-IoT. This could suggest IoT's frequent reliance on shell-based utilities (e.g., invoking BusyBox/Unix commands from admin UIs) where user-controlled parameters are passed to system calls, creating more and easier command injections.

Hard-coded credential (CWE-798) and uncontrolled resource consumption (CWE-400) weaknesses are seen in the top 20 weaknesses of only IoT. The Use of hardcoded-credentials (passwords, API keys), is a bad practice still prevalent in IoT systems to simplify initial

setup. In contrast, non-IoT systems more frequently rely on external configuration files or integrate with centralized credential management. Uncontrolled resource consumption (excessive CPU, memory, or network usage) tends to be more visible in IoT devices, which operate on resource-constrained hardware with limited resource control mechanisms. In contrast, non-IoT systems benefit from mechanisms such as resource monitoring, process isolation, and throttling mechanisms, which reduces the risk of such weaknesses.

Conversely, code injection (CWE-94) and file upload (CWE-434) are only seen in the top 20 CWEs of non-IoT. Code injection usually arises in interpreted environments that support runtime code interpretation (e.g., PHP), which is more common in non-IoT systems. In contrast, IoT systems mostly rely on low-level and compiled languages for performance and resource efficiency, limiting the possibility of code injection attacks. Similarly, file upload functionality (profile pictures, documents) is most common with web based applications, frequently used in non-IoT systems. This feature, when implemented insecurely, may allow a malicious file to be uploaded and executed on the web server. Usually, these kinds of vulnerabilities happen with back-end programming languages that are interpreted on the fly, e.g., PHP. IoT devices rarely expose such upload interfaces, and when a web server is present, it typically serves static content with minimal dynamic processing, making such weaknesses far less common.

Key insight: In the IoT sector, memory-safety weaknesses dominate, with web-centric weaknesses becoming increasingly common in recent years. This shows that IoT devices tend to expose web interfaces for supporting remote control and offering user interfaces. In non-IoT, web-centric weaknesses dominate by a wide margin due to the ubiquitous web applications and scanning tools, and memory-safety issues rank second. Moreover, the architectural differences in IoT systems (single tenancy, reliance on low-level programming languages, limited resources) have led to differences from non-IoT in their top-20 weaknesses. These include more authentication issues, lack of authorization issues, hard-coded credentials and uncontrolled resource consumption weaknesses.

6.4. Analysis on known exploited vulnerabilities

Known exploited vulnerabilities (KEV) refer to a list of vulnerabilities that are known to be exploited in the wild. Therefore, vulnerabilities listed in KEV should preferably be prioritized for patching. There are various data sources provide a KEV list. CISA maintains a Known Exploited Vulnerabilities (KEV) publicly and even enforces fixing CVEs in KEV in a timely manner. Similar to CISA, VulnCheck provides a KEV list, but with broader list of vulnerabilities. At the time of writing this study, VulnCheck KEV published 4005 CVEs, while CISA KEV has 1399 CVEs. As all CVEs in CISA KEV are contained in VulnCheck KEV, here we rely on VulnCheck[5] for our analysis. We identified 209 IoT CVEs as well as 3099 non-IoT CVEs from our dataset listed in VulnCheck KEV.

5. https://www.vulncheck.com/kev

7. Discussion

In this section, based on the insights observed from our study, we discuss underlying causes and responsible stakeholders, and provide actionable recommendations. We also list the limitations of our study and directions for future work.

7.1. Recommendations

Kill the low-hanging-fruit vulnerabilities. We observed hard-coded credentials as one of the common weaknesses only observed in the IoT top-20 weaknesses. Moreover, it is not unlikely that this weakness has gone under-reported. In fact, manufacturers usually rely on one credential for all products to make mass production easier and simplify troubleshooting during assembly. Two main mitigation options exist as also advocated by ETSI [76]. First, manufacturers should generate device-specific credentials. This approach is likely costly for manufacturers since it involves 2 additional steps in the production pipeline: 1) securely provisioning the credential on the device, 2) printing or embedding the credential in the product packaging or manual. Second, enforcing a mandatory credential change during the initial setup by the user, ensuring that the default credential becomes immediately invalid. This procedure requires digital literacy and might lead to customer service issues if users forget their credentials and hold the manufacturer responsible for this. To solve such cases, vendors should provide a secure reset process such as a physical reset mechanism that allows users to set new credential and maintain the usability of the device.

Eliminate memory safety issues. As we observed, memory safety issues are the most prevalent weaknesses in the IoT sector, also ranked second in non-IoT, and are constantly reported as the major issue in software/firmware vulnerability [77]. IoT systems are largely dependent on low-level, memory-unsafe languages due to strict resource constraints and legacy codebases. A key effective mitigation is the adoption of memory safe languages, which shifts the safety burden from developers to the language and the development environment. A case study on Android vulnerabilities showed that by transitioning to memory safe languages such as Java and Rust, memory safety issues, which accounted for 76% of Android vulnerabilities in 2019 plummeted to 24% by 2024 [78]. Therefore, such a transition in the IoT sector must be taken into action by IoT device manufacturers, firmware developers, and IoT application developers. Several memory safe languages and frameworks suitable for embedded environments include Rust [79], Ada [80], and MicroPython [81]. For cases where memory-unsafe languages remain necessary, developers should follow established secure coding standards (e.g., MISRA-C, CERT C) and use static and dynamic code analysis tools to find potential security vulnerabilities such as flawfinder, CodeQL, CommSCA, and Infer [82], [83]. Additionally, open-source component maintainers are another responsible stakeholder, as IoT systems rely heavily on open-source operating systems and libraries such as FreeRTOS, Zephyr RTOS, Contiki-NG, Mbed OS, and OpenThread. While open-source component maintainers are encouraged to support memory-

safe rewrites, IoT vendors who integrate these projects should also invest in such memory-safe transition efforts.
Prevent web-based issues. The appearance of several web-based weaknesses (XSS, command injection, SQLi) in the top-20 IoT weaknesses suggests that this family may grow in prominence as embedded web interfaces remain common for device configuration and management. As a fundamental mitigation, firmware and application developers should prevent these interfaces from being exposed to the public internet by blocking WAN access and restricting them to the local network. Network operators should block unsolicited inbound traffic by default through NAT or firewall rules. For remote management, IoT devices should instead rely on secure cloud-based remote management, placing responsibility on cloud providers to enforce strong authentication and secure session handling.
Legislation and enforcement. Recent regulatory efforts aim to ensure that manufacturers and other stakeholders implement minimum cybersecurity safeguards before products reach the market. The EU Cyber Resilience Act (CRA) [84], [85], which was adopted in December 2024 and comes into full force in 2027, mandates minimal cybersecurity standards for all software and connected hardware at every stage—design, development, deployment, and maintenance throughout their entire lifetime. Similarly, the Radio Equipment Directive (RED) [86] imposes additional cybersecurity requirements on wireless and radio-enabled devices, including ensuring that the equipment does not harm networks and protects user data and privacy. Therefore, all stakeholders in the supply chain are required to adhere to security-by-design practices, and products that do not meet these standards may not be legally placed on the EU market.

7.2. Limitations

In this section we discuss various aspects of our work and possible future directions.
Data source constraints. In our study we only rely on NVD CVE text and CPEs that both can be incomplete or inconsistent (aliases, typos, missing metadata). Although LLMs mitigate sparse descriptions better than humans on average, errors can persist when input data is limited or misleading. Other data sources and security advisories can be merged to enrich context, improve label consistency, and reduce ambiguity.
Reproducibility. While our experiments demonstrate that state-of-the-art LLMs can achieve high classification accuracy (up to 95%), their performance is sensitive to prompt design and model choice. Although we release our prompts and the dataset, as LLMs continue to evolve, reproducibility and long-term consistency may vary depending on the model version, API configuration, or fine-tuning updates. While we currently rely on zero-shot prompting, future work may explore fine-tuned models with few shots for improved stability and transparency.
Borderline technologies. As we observed in our analysis, while LLMs could effectively identify IoT CVEs, they are not consistent for borderline technologies, such as chipsets, gateways, light-weight libraries, and industrial controllers. However, inconsistencies in human and LLM labels for borderline technologies do not reflect shortcomings for either approach. It highlights the inherent ambiguity and the limitations of binary classification. Jin et al. [3] revealed mobile library vulnerabilities affecting IoT that are used in both IoT and non-IoT systems. Outside this boundary region, the LLM is highly consistent and our manual inspection confirms label quality, indicating that LLM-assisted labeling is effective when the IoT ecosystem is well defined. However, to overcome such inconsistencies, future work may benefit from more meta data, post classification treatment or multi-label classification to better capture this complexity.

7.3. Future work

In addition to the above listed future work directions to address the limitations of LLIoT, we believe that LLIoT dataset and the prompt openly shared in Appendix B can facilitate new research toward a better understanding of IoT vulnerabilities and ultimately mitigation strategies for the identified risks. By releasing our dataset publicly, we enable a wide range of further analyses, including: real-world measurements of IoT vulnerability prevalence in deployed systems, time-to-patch analysis, patch prioritization studies, vendor/product investigation, exploitation-likelihood comparisons with EPSS, and security assessments for specific device classes. Our work also gives insight on the usage of LLMs for CVE classification and lowers the barriers for finding IoT CVEs.

8. Conclusion

This work systematized the knowledge on IoT CVE vulnerabilities. We first studied the prior effort on IoT taxonomies for studying IoT vulnerabilities, practical approaches to distinguish IoT CVE vulnerabilities and analysis of IoT CVEs. This systematization revealed fundamental limitations in existing approaches, including inconsistent definitions of IoT, ambiguous criteria for identifying IoT-specific vulnerabilities, and reliance on incomplete or heuristic-driven datasets. To address these challenges, we operationalized this systematization through LLIoT, an LLM-assisted approach for systematically identifying IoT-specific CVEs in the NVD at scale. By formalizing a four-layer IoT ecosystem and constructing a high-quality, expert-validated ground-truth dataset of IoT CVEs, we showed that LLMs can reliably distinguish IoT from non-IoT CVEs. We applied the classification to CVEs from 2013–2024, and identified 15,116 IoT-specific CVEs, including 8,368 previously unreported by existing datasets. Our comparison against existing IoT CVE datasets highlighted improved labeling quality and coverage, making our dataset, to the best of our knowledge, the most comprehensive collection of IoT-specific CVEs to date. Using this dataset, we conducted a large-scale analysis of IoT vulnerabilities, finding that IoT vulnerabilities have increased significantly since 2016 and tend to have higher severity, broader impact, and lower attacker effort compared to non-IoT. We discussed causes of observed IoT weaknesses and provided a call to action for responsible stakeholders. By releasing our dataset, prompt, and methodology openly, we aim to support reproducible research and enable deeper, data-driven understanding of IoT security.

References

[1] "Iotstatistia." https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/. Accessed: April 2025.

[2] X. Feng, X. Zhu, Q.-L. Han, W. Zhou, S. Wen, and Y. Xiang, "Detecting vulnerability on iot device firmware: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 1, pp. 25–41, 2022.

[3] X. Jin, S. Manandhar, K. Kafle, Z. Lin, and A. Nadkarni, "Understanding iot security from a market-scale perspective," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1615–1629, 2022.

[4] X. Wang, Y. Sun, S. Nanda, and X. Wang, "Looking from the mirror: Evaluating {IoT} device security through mobile companion apps," in *28th USENIX security symposium (USENIX security 19)*, pp. 1151–1167, 2019.

[5] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *2019 IEEE symposium on security and privacy (sp)*, pp. 1362–1380, IEEE, 2019.

[6] X. Feng, X. Liao, X. Wang, H. Wang, Q. Li, K. Yang, H. Zhu, and L. Sun, "Understanding and securing device vulnerabilities through automated bug report analysis," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 887–903, 2019.

[7] "Common Platform Enumeration (CPE)." https://nvd.nist.gov/products/cpe. Accessed: October 20, 2025.

[8] D. Tovarňák, L. Sadlek, and P. Čeleda, "Graph-based cpe matching for identification of vulnerable asset configurations," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 986–991, IEEE, 2021.

[9] R. Kaksonen, K. Halunen, and J. Röning, "Vulnerabilities in iot devices, backends, applications, and components," in *Proceedings of the 9th International Conference on Information Systems Security and Privacy*, Scitepress, 2023.

[10] D. Papp, Z. Ma, and L. Buttyan, "Embedded systems security: Threats, vulnerabilities, and attack taxonomy," in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pp. 145–152, ieee, 2015.

[11] C. A. Rivera A, A. Shaghaghi, D. D. Nguyen, and S. S. Kanhere, "Is this iot device likely to be secure? risk score prediction for iot devices using gradient boosting machines," in *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pp. 115–127, Springer, 2021.

[12] B. Xiang, Y. Zhang, F. Liu, H. Huang, Z. Lin, and M. Yang, "Pig in a poke: Automatically detecting and exploiting link following vulnerabilities in windows file operations," 2025.

[13] I. Sayar, A. Bartel, E. Bodden, and Y. Le Traon, "An in-depth study of java deserialization remote-code execution exploits and vulnerabilities," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 1, pp. 1–45, 2023.

[14] X. Tan, Y. Zhang, C. Mi, J. Cao, K. Sun, Y. Lin, and M. Yang, "Locating the security patches for disclosed oss vulnerabilities with vulnerability-commit correlation ranking," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3282–3299, 2021.

[15] X. Zhou, T. Zhang, and D. Lo, "Large language model for vulnerability detection: Emerging results and future directions," in *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, pp. 47–51, 2024.

[16] X. Yin, C. Ni, and S. Wang, "Multitask-based evaluation of open-source llm on software vulnerability," *IEEE Transactions on Software Engineering*, 2024.

[17] C. Vasilatos, D. J. Mahboobeh, H. Lamri, M. Alam, and M. Maniatakos, "Llmpot: Dynamically configured llm-based honeypot for industrial protocol and physical process emulation," in *2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P)*, pp. 963–979, IEEE, 2025.

[18] J. Akhoundali, H. Hamidi, K. Rietveld, and O. Gadyatskaya, "Eradicating the unseen: Detecting, exploiting, and remediating a path traversal vulnerability across github," in *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, pp. 542–558, 2025.

[19] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[20] R. Shokeen, B. Shanmugam, K. Kannoorpatti, S. Azam, M. Jonkman, and M. Alazab, "Vulnerabilities analysis and security assessment framework for the internet of things," in *2019 Cybersecurity and Cyberforensics Conference (CCC)*, pp. 22–29, IEEE, 2019.

[21] S. Jamshidi, N. Shahabi, A. Nikanjam, K. W. Nafi, F. Khomh, and C. Fung, "The role of large language models in iot security: A systematic review of advances, challenges, and opportunities," *Internet of Things*, p. 101735, 2025.

[22] S. Rizvi, A. Kurtz, J. Pfeffer, and M. Rizvi, "Securing the internet of things (iot): A security taxonomy for iot," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 163–168, IEEE, 2018.

[23] Y. Y. Shaheen, M. J. Hornos, and C. Rodríguez-Domínguez, "A systematic mapping study on privacy in iot-based systems," *International Journal of Information Security*, vol. 24, no. 6, p. 236, 2025.

[24] M. binti Mohamad Noor and W. H. Hassan, "Current research on internet of things (iot) security: A survey," *Computer Networks*, vol. 148, pp. 283–294, 2019.

[25] H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, M. Aledhari, and H. Karimipour, "A survey on internet of things security: Requirements, challenges, and solutions," *Internet of Things*, vol. 14, p. 100129, 2021.

[26] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization," *Computer networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[27] M. Abbasi, M. Plaza-Hernandez, J. Prieto, and J. M. Corchado, "Security in the internet of things application layer: requirements, threats, and solutions," *IEEE Access*, vol. 10, pp. 97197–97216, 2022.

[28] Y. R. Siwakoti, M. Bhurtel, D. B. Rawat, A. Oest, and R. C. Johnson, "Advances in iot security: Vulnerabilities, enabled criminal services, attacks, and countermeasures," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11224–11239, 2023.

[29] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE internet of things journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[30] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Jin, "Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice," *Journal of Hardware and Systems Security*, vol. 2, no. 2, pp. 97–110, 2018.

[31] V. Adat and B. B. Gupta, "Security in internet of things: issues, challenges, taxonomy, and architecture," *Telecommunication Systems*, vol. 67, no. 3, pp. 423–441, 2018.

[32] P. I. R. Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the internet of things: Challenges, threats and solutions," *Internet of things*, vol. 5, pp. 41–70, 2019.

[33] J. Guth, U. Breitenbücher, M. Falkenthal, F. Leymann, and L. Reinfurt, "Comparison of iot platform architectures: A field study based on a reference architecture," in *2016 Cloudification of the Internet of Things (CIoT)*, pp. 1–6, IEEE, 2016.

[34] L. P. Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac, "Survey on enterprise internet-of-things systems (e-iot): A security perspective," *Ad Hoc Networks*, vol. 125, p. 102728, 2022.

[35] B. K. Mohanta, D. Jena, U. Satapathy, and S. Patnaik, "Survey on iot security: Challenges and solution using machine learning, artificial intelligence and blockchain technology," *Internet of things*, vol. 11, p. 100227, 2020.

[36] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[37] B. Di Martino, M. Rak, M. Ficco, A. Esposito, S. A. Maisto, and S. Nacchia, "Internet of things reference architectures, security and interoperability: A survey," *Internet of Things*, vol. 1, pp. 99–112, 2018.

[38] M. Janiszewski, A. Felkner, P. Lewandowski, M. Rytel, and H. Romanowski, "Automatic actionable information processing and trust management towards safer internet of things," *Sensors*, vol. 21, no. 13, p. 4359, 2021.

[39] A. A. Al Alsadi, M. Vermeer, T. Sasaki, K. Yoshioka, M. Van Eeten, and C. Gañán, "Bits and pieces: Piecing together factors of iot vulnerability exploitation," in *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, pp. 1032–1049, 2025.

[40] X. Chen, C. Yang, Y. Nan, and Z. Zheng, "An empirical study of high-risk vulnerabilities in iot systems," *IEEE Internet of Things Journal*, 2024.

[41] A. Kumar and C. Fung, "An analysis on cve vulnerabilities of the internet of things," in *2024 7th Conference on Cloud and Internet of Things (CIoT)*, pp. 1–6, IEEE, 2024.

[42] J. Song, S. Wan, M. Huang, J. Liu, L. Sun, and Q. Li, "Toward automatically connecting iot devices with vulnerabilities in the wild," *ACM Transactions on Sensor Networks*, vol. 20, no. 1, pp. 1–26, 2023.

[43] A. Allen, A. Mylonas, S. Vidalis, and D. Gritzalis, "Security evaluation of companion android applications in iot: The case of smart security devices," *Sensors*, vol. 24, no. 17, p. 5465, 2024.

[44] B. Zhao, S. Ji, J. Xu, Y. Tian, Q. Wei, Q. Wang, C. Lyu, X. Zhang, C. Lin, J. Wu, *et al.*, "A large-scale empirical analysis of the vulnerabilities introduced by third-party components in iot firmware," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 442–454, 2022.

[45] P. Oser, R. W. van der Heijden, S. Lüders, and F. Kargl, "Risk prediction of iot devices based on vulnerability analysis," *ACM Transactions on Privacy and Security*, vol. 25, no. 2, pp. 1–36, 2022.

[46] G. J. Blinowski and P. Piotrowski, "Cve based classification of vulnerable iot systems," in *International Conference on Dependability and Complex Systems*, pp. 82–93, Springer, 2020.

[47] C. A. R. A, X. Chen, A. Shaghaghi, G. Batista, and S. Kanhere, "Predicting iot device vulnerability fix times with survival and failure time models," 2025.

[48] S. R. Pérez, M. Van Eeten, and C. H. Gañán, "Patchy performance? uncovering the vulnerability management practices of iot-centric vendors," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 1198–1216, IEEE, 2024.

[49] F. Arat and S. Akleylek, "A new method for vulnerability and risk assessment of iot," *Computer Networks*, vol. 237, p. 110046, 2023.

[50] B. Zhao, S. Ji, W.-H. Lee, C. Lin, H. Weng, J. Wu, P. Zhou, L. Fang, and R. Beyah, "A large-scale empirical study on the vulnerability of deployed iot devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1826–1840, 2020.

[51] Z. Fang, H. Fu, T. Gu, P. Hu, J. Song, T. Jaeger, and P. Mohapatra, "Towards system-level security analysis of iot using attack graphs," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1142–1155, 2022.

[52] J. M. Ford, "Content analysis: An introduction to its methodology," *Personnel psychology*, vol. 57, no. 4, p. 1110, 2004.

[53] G. Marzi, M. Balzano, and D. Marchiori, "K-alpha calculator–krippendorff's alpha calculator: a user-friendly tool for computing krippendorff's alpha inter-rater reliability coefficient," *MethodsX*, vol. 12, p. 102545, 2024.

[54] T. Alladi, V. Chamola, B. Sikdar, and K.-K. R. Choo, "Consumer iot: Security vulnerability case studies and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, pp. 17–25, 2020.

[55] B. D. Davis, J. C. Mason, and M. Anwar, "Vulnerability studies and security postures of iot devices: A smart home case study," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10102–10110, 2020.

[56] B. A. Visan, J. Lee, B. Yang, A. H. Smith, and E. T. Matson, "Vulnerabilities in hub architecture iot devices," in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 83–88, IEEE, 2017.

[57] S. Manandhar, K. Moran, K. Kafle, R. Tang, D. Poshyvanyk, and A. Nadkarni, "Towards a natural perspective of smart homes for practical security and safety analyses," in *2020 ieee symposium on security and privacy (sp)*, pp. 482–499, IEEE, 2020.

[58] "Amazon best sellers smart home." https://www.amazon.com/Best-Sellers-Smart-Home/zgbs/smart-home. Accessed: April 2025.

[59] "Amazon best sellers home automation." https://www.amazon.com/Best-Sellers-Smart-Home/zgbs/smart-home. Accessed: April 2025.

[60] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[61] I. Nadir, Z. Ahmad, H. Mahmood, G. A. Shah, F. Shahzad, M. Umair, H. Khan, and U. Gulzar, "An auditing framework for vulnerability analysis of iot system," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 39–47, IEEE, 2019.

[62] I. Zavalyshyn, A. Legay, A. Rath, and E. Rivière, "Sok: Privacy-enhancing smart home hubs," *Proceedings on Privacy Enhancing Technologies*, 2022.

[63] "Gartner cloud iot platforms." https://www.gartner.com/reviews/market/global-industrial-iot-platforms/vendor/alibaba-cloud/product/cloud-iot-platforms/alternatives. Accessed: June 2025.

[64] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of things: Security vulnerabilities and challenges," in *2015 IEEE symposium on computers and communication (ISCC)*, pp. 180–187, IEEE, 2015.

[65] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110, 2017.

[66] "Satori IoT botnet." https://shorturl.at/6bDMo. Accessed: October 2, 2025.

[67] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin, "Measurement and analysis of hajime, a peer-to-peer iot botnet," in *Network and Distributed Systems Security (NDSS) Symposium*, 2019.

[68] "Reaper IoT botnet." https://www.wired.com/story/reaper-iot-botnet-infected-million-networks/. Accessed: October 2, 2025.

[69] "Dark Nexus IoT Botnet." https://digital.nhs.uk/cyber-alerts/2020/cc-3425. Accessed: October 2, 2025.

[70] P. Victor, A. H. Lashkari, R. Lu, T. Sasi, P. Xiong, and S. Iqbal, "Iot malware: An attribute-based taxonomy, detection mechanisms and challenges," *Peer-to-peer Networking and Applications*, vol. 16, no. 3, pp. 1380–1431, 2023.

[71] L. McNulty and V. G. Vassilakis, "Iot botnets: Characteristics, exploits, attack capabilities, and targets," in *2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pp. 350–355, IEEE, 2022.

[72] N. Elsayed, Z. ElSayed, and M. Bayoumi, "Iot botnet detection using an economic deep learning model," in *2023 IEEE World AI IoT Congress (AIIoT)*, pp. 0134–0142, IEEE, 2023.

[73] "CVE Continues to Grow." https://www.cve.org/about/history?utm_source=chatgpt.com. Accessed: October 20, 2025.

[74] S. Bhatkar, D. C. DuVarney, and R. Sekar, "Address obfuscation: An efficient approach to combat a broad range of memory error exploits," in *12th USENIX Security Symposium (USENIX Security 03)*, 2003.

[75] K. V. English, I. Obaidat, and M. Sridhar, "Exploiting memory corruption vulnerabilities in connman for iot devices," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 247–255, IEEE, 2019.

[76] "ETSI EN 303 645." https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf. Accessed: 10 Nov 2025.

[77] "Exploring Memory Safety in Critical Open Source Projects." https://www.cisa.gov/sites/default/files/2024-06/joint-guidance-exploring-memory-safety-in-critical-open-source-projects-508c.pdf. Accessed: November 10, 2025.

[78] "In practice on Android." https://security.googleblog.com/2024/09/eliminating-memory-safety-vulnerabilities-Android.html. Accessed: October 20, 2025.

[79] "Rust Programming Language." https://rust-lang.org/. Accessed: 10 Nov 2025.

[80] "Ada Programming Language." https://www.adacore.com/languages/ada. Accessed: 10 Nov 2025.

[81] "Micro Python." https://micropython.org/. Accessed: 10 Nov 2025.

[82] S. Lipp, S. Banescu, and A. Pretschner, "An empirical study on the effectiveness of static c code analyzers for vulnerability detection," in *Proceedings of the 31st ACM SIGSOFT international symposium on software testing and analysis*, pp. 544–555, 2022.

[83] J. Zheng, L. Williams, N. Nagappan, W. Snipes, J. P. Hudepohl, and M. A. Vouk, "On the value of static analysis for fault detection in software," *IEEE transactions on software engineering*, vol. 32, no. 4, pp. 240–253, 2006.

[84] "Cyber Resilience Act (CRA)." https://eur-lex.europa.eu/eli/reg/2024/2847/oj. Accessed: October 20, 2025.

[85] "Cyber Resilience Act- links." https://www.european-cyber-resilience-act.com/Cyber_Resilience_Act_Links.html. Accessed: October 20, 2025.

[86] "Radio Equipment Directive (RED)." https://single-market-economy.ec.europa.eu/sectors/electrical-and-electronic-engineering-industries-eei/radio-equipment-directive-red_en. Accessed: October 20, 2025.

Appendix A. Web interface for manual review

Fig. 10 shows our web interface designed for reviewers to facilitate their reviewing process. Each user was given their own credentials to login to the portal and review their assigned CVEs.

Appendix B. LLM prompt

The prompt given to LLMs contains the same IoT ecosystem definition and classification rules presented to human participants, ensuring that both groups operate under an identical conceptual framework. The complete prompt template used for all model inferences appears in Figure 11.

Appendix C. Manual analysis

Table 7 shows the result of manual inspection of 100 samples randomly selected from VarIoT from the "other" category. However, our manual inspection was limited to only 100 samples out of almost 18000. To further prob potential misclassifications, we conducted a keyword-based search within VarIoT's "other" category, focusing on terms observed during manual review (e.g., Adobe Flash Player, WordPress, Safari, Google Chrome, Perl). Our keyword search revealed large numbers of clearly non-IT CVEs: 671 related to Adobe flash player, 9 related to Perl, 62 related to wordpress, 61 related to Google chrome, 236 related to Cisco WebEx, 19 related to Cisco Unified MeetingPlace app, and 801 related to Safari that are clearly non-IoT. In addition 528 CVEs were associated with routers and 269 were related to switches.

Table 9 shows the results of a manual review of 100 CVEs randomly drawn from Chen's dataset that are exclusive to Chen (not observed by other datasets). Finally, Table 11 presents the results of a manual evaluation of 100 CVEs randomly sampled from those that appear only in our dataset.

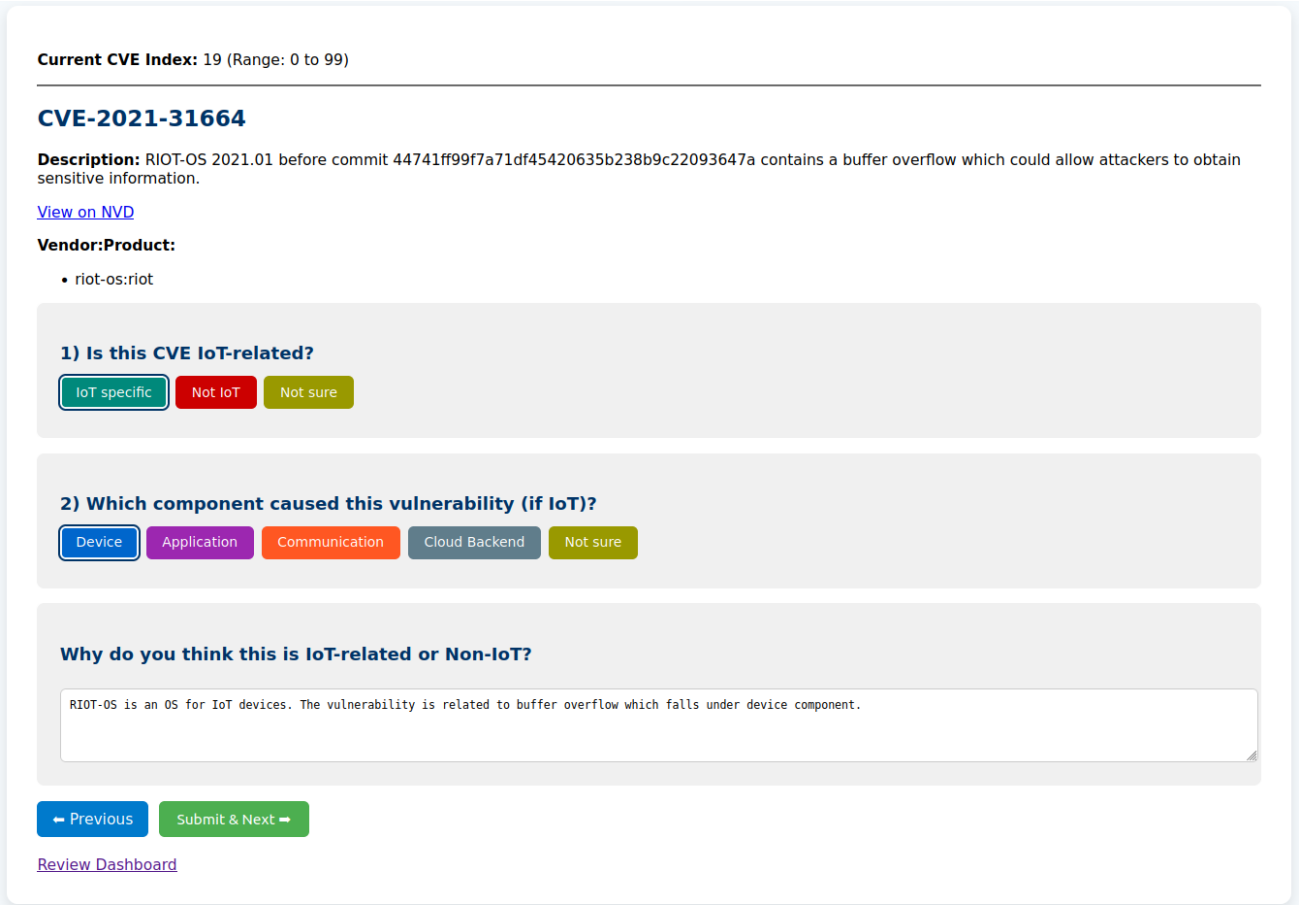


Figure 10: The interface used by reviewers to assess assigned CVEs. For each CVE, the description and associated CPEs are displayed, along with a link to the corresponding NVD page for further reference and information.

Prompt

```
We define the IoT ecosystem as comprising the following four layers:
1-Device layer: Physical devices that often have limited power, memory,
    computational capacity and either collect sensory data or perform actuation
    tasks. These devices also interact with other devices directly or indirectly
    using communication technologies with limited or no human intervention. Examples
     include smart thermostats, Amazon Echo, motion sensors, smart cameras, and
    smart TVs. Typical vulnerabilities of this layer originate from the hardware,
    firmware, or embedded OSes/software (e.g., FreeRTOS, TinyOS, RIOT-OS).
2-Communication layer: This layer includes any protocol that facilitates data
    transfer and connectivity for IoT devices within the local network or outside of
     the network. It involves specialized low-power protocols (BLE, ZigBee). Typical
     vulnerabilities of this layer include flaws in the protocol design,
    implementation or configuration bugs in protocol stacks (e.g., unencrypted data
    exchange, lack of authentication).
3-Application layer: This layer includes mobile applications and orchestration
    systems/local controllers that sit above the hardware and enable user-space
    services for IoT. Examples include IoT-focused platforms and applications/
    companion applications (Home Assistant, SmartThings, IFTTT). Typical
    vulnerabilities include flaws in the design or implementation logic of these
    platforms and applications.
4-Cloud backend layer: Remote or public cloud endpoints that provide IoT-specific
    services, such as device management, remote control, and other IoT-focused
    functionalities. Examples include AWS IoT, Azure IoT, Google Cloud IoT, IBM
    Watson IoT. Typical vulnerabilities of this layer include flaws in the
    configuration of the infrastructure, vulnerable services, insecure cloud APIs or
     tokens, and improper input validation on the server side.

Given the above definition of the IoT ecosystem, CVE classification rules are:
1)IoT-specific CVE: vulnerabilities that affect technologies belonging to IoT
    ecosystem, where these technologies are purpose-built for IoT and the deployment
     context is IoT. Moreover, rhe vulnerable product is itselft an active,
    functional element of an IoT system.
2)non-IoT CVE: When there is insufficient or no clear evidence of its relevance to
    IoT usage, or it does not significantly impact the IoT ecosystem, then it is
    nonIoT. General-purpose hardware/software such as smartphone, computers, or
    tablets are considered as non-IoT. Network infrastructures such as routers,
    modems are also non-IoT.

Classify the following CVE using the above rules.
Description: {description}
Affected Products: {products}

Respond only with a JSON object in the format below-no commentart, no markdown:
    {{
      "label": "IoT" or "nonIoT",
      "layer": "device" or "communication" or "application" or "cloud backend" or "
          unsure",
      "reason": "Your explanation and reasoning",
      "category": "only a general category of your choice based on the affected
          products",
      "sector": "industrial iot" or "automative iot" or "smart cities iot" or "
          consumer iot" or "healthcare iot" or "smart finance" or "other".
    }}"
```

Figure 11: Prompt used for automated IoT-related CVE classification.

Table 7: Result of manual inspection of 100 samples randomly taken from VarIoT's "other" category.

Category	**CVE IDs**
Network infrastructure	CVE-2021-34765, CVE-2021-44168, CVE-2019-5986, CVE-2022-41030, CVE-2023-25087, CVE-2023-3450, CVE-2023-38924, CVE-2023-52026, CVE-2024-30598, CVE-2024-35397, CVE-2024-37186, CVE-2024-39759, CVE-2024-44558, CVE-2024-48629, CVE-2024-51018, CVE-2013-2341, CVE-2013-2681, CVE-2013-3098, CVE-2016-1457, CVE-2018-14712, CVE-2020-25988, CVE-2017-18704, CVE-2017-18735, CVE-2017-5262, CVE-2021-1226, CVE-2023-22839, CVE-2013-0600, CVE-2013-4875, CVE-2014-4347, CVE-2018-1429, CVE-2018-11420, CVE-2013-4628, CVE-2018-7120, CVE-2019-1806, CVE-2019-1949, CVE-2015-2904, CVE-2021-37730, CVE-2022-20674, CVE-2022-20830, CVE-2022-22163, CVE-2022-24142, CVE-2022-25219, CVE-2022-29525
Chipsets	CVE-2015-1137, CVE-2015-9156, CVE-2022-32966, CVE-2015-9181, CVE-2020-12350, CVE-2018-11998
HarmonyOS/EMUI	CVE-2021-40005, CVE-2022-22256, CVE-2022-39004, CVE-2022-41588
Apple platforms (iOS / tvOS / watchOS / macOS)	CVE-2017-13854, CVE-2017-2476, CVE-2017-7020, CVE-2018-4328, CVE-2019-6209, CVE-2020-3880, CVE-2020-9805, CVE-2020-9976, CVE-2021-1881, CVE-2021-30703, CVE-2022-32792
Google chrome	CVE-2013-0896
WordPress plugin	CVE-2019-18668
Perl	CVE-2020-10878
Linux	CVE-2019-19053
Desktop app	CVE-2015-4233, CVE-2017-16745, CVE-2017-12361, CVE-2019-1772, CVE-2020-3413, CVE-2022-41664, CVE-2023-24978
NUC	CVE-2020-8742
Adobe flash player	CVE-2015-3118, CVE-2015-8453
OpenSSL	CVE-2015-0204, CVE-2016-2108
HTTP server	CVE-2017-16166
Object storage server	CVE-2021-21287
MacOS and Safari	CVE-2021-1805, CVE-2021-30930, CVE-2017-2540, CVE-2013-0971, CVE-2016-1764, CVE-2015-1155, CVE-2015-3751
Smartphones	CVE-2014-6848, CVE-2015-5857, CVE-2021-30875, CVE-2016-6728, CVE-2021-37113, CVE-2017-17313, CVE-2017-8208, CVE-2019-5244, CVE-2019-15466
Video conferencing software	CVE-2014-3340

Table 9: Result of manual inspection of 100 CVEs randomly selected from Chen's "other" category.

Category	**CVE IDs**
Network infrastructure / routers	CVE-2019-0063, CVE-2022-29729, CVE-2022-38789, CVE-2022-46740, CVE-2021-37273, CVE-2022-47052, CVE-2022-22236, CVE-2014-9222, CVE-2020-22655, CVE-2020-22656, CVE-2020-22658, CVE-2021-20121, CVE-2020-25858
Chipsets	CVE-2016-2453, CVE-2016-8420, CVE-2016-8421, CVE-2016-8452, CVE-2016-8456, CVE-2016-8465, CVE-2017-0430, CVE-2017-0523, CVE-2017-0526, CVE-2017-0527, CVE-2017-0566, CVE-2017-0572, CVE-2017-0621, CVE-2017-0705, CVE-2017-0786, CVE-2017-0787, CVE-2017-0791, CVE-2017-13172, CVE-2019-20612, CVE-2020-26555, CVE-2021-33139, CVE-2021-36922, CVE-2021-36923, CVE-2021-36925, CVE-2022-20041, CVE-2022-20043, CVE-2022-20046, CVE-2022-32658, CVE-2022-32659, CVE-2022-36338, CVE-2019-14620, CVE-2021-3011, CVE-2021-30344, CVE-2021-1887, CVE-2020-12321
Wi-Fi/sensor/Bluetooth modules in Android and Linux	CVE-2016-0820, CVE-2016-10292, CVE-2016-6691, CVE-2016-8453, CVE-2016-8457, CVE-2016-8466, CVE-2017-0567, CVE-2017-0568, CVE-2017-0569, CVE-2017-0571, CVE-2017-0788, CVE-2020-24586, CVE-2021-27803, CVE-2017-0789, CVE-2017-0518, CVE-2017-0519, CVE-2019-18618, CVE-2016-3798, CVE-2018-10910, CVE-2022-47367, CVE-2022-25837, CVE-2020-15238
Apple platforms (iOS / tvOS / watchOS / macOS)	CVE-2019-8854
Windows	CVE-2021-0152, CVE-2021-0164, CVE-2021-0151, CVE-2020-0554, CVE-2015-0884, CVE-2021-0169, CVE-2021-0171
Smartphones (Android)	CVE-2016-2501, CVE-2020-25282, CVE-2015-8938, CVE-2016-2488, CVE-2016-3859
Desktop applications	CVE-2022-28762, CVE-2020-10858, CVE-2020-28421, CVE-2019-15742, CVE-2019-11073, CVE-2019-11074
Software library	CVE-2018-1000045
Wired speakers	CVE-2021-38365, CVE-2021-38546, CVE-2021-38547
Mobile and web apps	CVE-2020-24199, CVE-2020-24196, CVE-2021-46067, CVE-2014-7689, CVE-2020-12270, CVE-2020-12856

Table 11: Result of manual inspection of 100 CVEs randomly selected CVEs specific to LLIoT.

Category	**CVE IDs**
Camera	CVE-2024-45173, CVE-2024-22771, CVE-2023-6321, CVE-2023-31595, CVE-2023-31678, CVE-2023-41918, CVE-2015-2099, CVE-2015-2860, CVE-2015-8039, CVE-2019-7338, CVE-2019-9682, CVE-2020-23826, CVE-2021-21941, CVE-2021-42109
Printer	CVE-2023-0855, CVE-2024-4782, CVE-2022-2888
Smart lock	CVE-2023-33371, CVE-2023-7009
Smart tv	CVE-2021-34403
Wearable	CVE-2020-27486
Infusion pump	CVE-2022-43557
Drone	CVE-2023-6948
Image scanning device	CVE-2024-47946
RTOS	CVE-2023-5779, CVE-2021-3455, CVE-2023-6249, CVE-2023-4263, CVE-2024-43696
Chipsets and embedded firmware/-SoC	CVE-2019-10544, CVE-2019-14073, CVE-2020-3660, CVE-2021-30262, CVE-2024-43056, CVE-2023-33104, CVE-2022-38155, CVE-2023-2481, CVE-2023-32829,
Industrial controller/firmware	CVE-2023-1898, CVE-2018-17932, CVE-2023-43848, CVE-2021-34581, CVE-2024-26002, CVE-2021-39243, CVE-2022-47389, CVE-2023-5392, CVE-2022-44354, CVE-2022-31206, CVE-2022-32136, CVE-2022-4224, CVE-2024-24972, CVE-2024-34542, CVE-2022-3010, CVE-2024-11611, CVE-2022-4286, CVE-2023-46386, CVE-2021-41300, CVE-2021-20589
EV charging station controller/interface	CVE-2024-11665, CVE-2021-22773, CVE-2024-8997
Embedded IoT SDK/JS engine	CVE-2024-38158, CVE-2021-29328, CVE-2021-46540, CVE-2023-49552
Industrial communication and networking stack	CVE-2023-26494, CVE-2021-33889, CVE-2024-31030, CVE-2024-28286, CVE-2022-25897, CVE-2024-11588, CVE-2024-43694,
Industrial/IoT/OT software & platforms	CVE-2023-25547, CVE-2023-25548, CVE-2016-5800, CVE-2022-43485, CVE-2022-36635, CVE-2022-26082, CVE-2022-27893, CVE-2023-3000, CVE-2020-18667, CVE-2024-6596
Industrial controllers/RTUs	CVE-2022-30274, CVE-2023-48242, CVE-2020-24705
HarmonyOS	CVE-2024-58110
Consumer IoT Building automation	CVE-2020-10951, CVE-2019-16733, CVE-2024-48786, CVE-2018-11402 CVE-2024-7732, CVE-2024-4292, CVE-2021-22720
Digital signage systems	CVE-2022-36250, CVE-2013-5979,
PLC engineering/robotic simulation/web management software	CVE-2021-43990, CVE-2021-43554, CVE-2020-4805, CVE-2022-24628