"*Belt and suspenders*" or "*just red tape*"?: Investigating Early Artifacts and User Perceptions of IoT App Security Certification

Prianka Mandal, Amit Seal Ami, Victor Olaiya, Sayyed Hadi Razmjo, Adwait Nadkarni
William & Mary
{pmandal, aami, voolaiya, srazmjo, apnadkarni}@wm.edu

Abstract

As IoT security regulations and standards emerge, the industry has begun adopting the traditional enforcement model for software compliance to the IoT domain, wherein Commercially Licensed Evaluation Facilities (CLEFs) certify vendor products on behalf of regulators (and in turn consumers). Since IoT standards are in their formative stages, we investigate a simple but timely question: *does the traditional model work for IoT security, and more importantly, does it work as well as consumers expect it to?* This paper investigates the initial artifacts resultant from IoT compliance certification, and user perceptions of compliance, in the context of certified mobile-IoT apps, i.e., critical companion and automation apps that expose an important IoT attack surface, with a focus on three key questions: (1) are certified IoT products vulnerable?, (2) are vulnerable-but-certified products non-compliant?, and finally, (3) how do consumers perceive compliance enforcement? Our systematic analysis of 11 mobile-IoT apps certified by IOXT, along with an analysis of 5 popular compliance standards, and a user study with 173 users, together yield *17 key findings*. We find significant vulnerabilities that indicate gaps in certification, but which do not violate the standards due to ambiguity and discretionary language. Further, these vulnerabilities contrast with the overwhelming trust that users place in compliance certification and certified apps. We conclude with a discussion on future directions towards a "belt and suspenders" scenario of effective assurance that most users desire, from the status quo of "just red tape", through objective checks and balances that empower the regulators and consumers to reform compliance enforcement for IoT.

1 Introduction

Federal and state governments have only recently begun to grapple with the reality of billions of potentially vulnerable Internet of Things (IoT) products, and have responded with the promise of targeted security and privacy regulations [24, 47, 58]. For instance, the US Senate is considering *The Cyber Shield Act* [59], which provisions security certifications for IoT products in accordance with IoT-specific security standards developed by the National Institute of Standards and Technology (NIST) and other standards bodies. Similarly, a recent US presidential executive order [56] instructs NIST to identify the relevant criteria that defines comprehensive security testing requirements for IoT products. While these policy initiatives are a step in the right direction, their effectiveness on IoT security in practice hinges their actual enforcement.

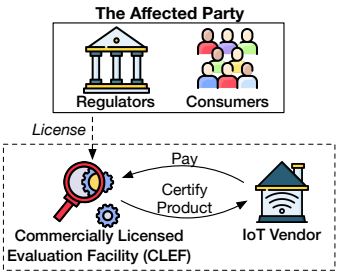

Figure 1: The Product Security Certification Loop

Emerging IoT security standards and regulations outline an enforcement strategy similar to the one used for software security certification, which we term as the *traditional* model, as shown in Figure 1. Under this model, regulators (and by extension consumers) delegate enforcement to several *Commercially Licensed Evaluation Facilities (CLEFs)*. Product vendors contract CLEFs, which then evaluate and certify products. Such delegation enables certification to scale through hundreds of CLEFs that certify millions of products produced by thousands of small and medium-scale vendors. However, this delegation comes at a steep price: as seen in Figure 1, *the affected party*, i.e., regulators and consumers who are the primary *beneficiaries* of compliance, are left (almost) entirely *outside the product security certification loop*.

The limited role of the affected party enables an incentive structure that is skewed *against* effective enforcement. Particularly, vendors have no incentive to select an ideal CLEF that would thoroughly evaluate their product, instead seeking the fastest route to certification, as Lerner and Tirole demonstrate in their study of "Forum Shopping" [43]. Moreover, CLEFs are licensed on the basis of *procedural competence* (e.g., facilities as per ISO 17025 [31]) and not *performance* at finding

vulnerabilities in practice, which reduces their incentive to improve. In effect, the traditional model disenfranchises the affected party, rendering it powerless to directly affect improvement in either CLEFs or vendors.

This paper seeks to motivate a change to this status quo by investigating the current state of *product security certification in IoT*, guided by the following question:

> Does the *traditional* security certification model (i) work in the context of IoT products, and more importantly, (ii) work as well as consumers (i.e., the affected party) expect it to?

This question is timely, as IoT standards are in their formative stages, and re-thinking compliance enforcement for IoT is feasible. To this end, we further distill this question into three ***research questions (RQ$_1$→RQ$_3$)*** that help us investigate the initial outcomes of the traditional certification model applied to IoT, and consumer perceptions regarding it:

RQ$_1$: *Are certified IoT products vulnerable?* What are the characteristics of the vulnerabilities they exhibit? Are they less vulnerable relative to non-certified products?

RQ$_2$: *Are vulnerable but certified IoT products non-compliant?* Do the vulnerabilities in certified products violate relevant IoT standards? If not, despite vulnerabilities, why?

RQ$_3$: *How do consumers, i.e., the affected party, perceive compliance enforcement?* How aware are consumers of security compliance standards? What are their expectations from certified IoT products? Who do they hold responsible for ensuring security compliance, or liable for failures?

With the view of enabling a tractable analysis and generating actionable insights, we scope our investigation of RQ$_1$→RQ$_3$ by targeting an *IoT product-type* that is *central* to the IoT system as a whole: *mobile-IoT apps* [34], i.e., companion apps and third-party automation services help users control devices and manifest an important attack surface of the IoT system. This work makes the following contributions:

- **Analysis of certified mobile-IoT apps (RQ$_1$):** We analyze 11 certified mobile-IoT apps from IOXT [32], focusing on vulnerabilities resulting from *cryptographic API misuse* that critically impact on the secrecy/integrity of IoT data in transit or at rest [37, 38] and are highly relevant to mobile apps. In addition, we target mobile security issues explicitly outlined in standards, namely requesting more permissions than necessary, and leaking private data. Our analysis of crypto-API misuse finds 35 serious vulnerabilities in 9/11 apps, including attempts to ***evade compliance and security checks*** ($\mathcal{F}_1$). Many of the vulnerabilities are not detected by popular crypto-API misuse detectors [40, 49, 53] ($\mathcal{F}_5$), and adversely impact the security of IoT data (e.g., camera feeds, authentication and account credentials) ($\mathcal{F}_2$, $\mathcal{F}_3$). Additionally, the mobile-IoT apps request at least one dangerous permission without justifying it to the user ($\mathcal{F}_7$) and may leak sensitive data to logs ($\mathcal{F}_8$). Repeating the analysis on a comparable set of non-certified apps reveals a similar rate of vulnerabilities ($\mathcal{F}_6$), indicating a potential gap between consumer expectations and reality (see $\mathcal{F}_{14}$).
- **Analysis of security compliance (RQ$_2$):** We evaluate the compliance of the certified (but vulnerable) mobile-IoT apps with IOXT (according to which they were certified), as well as 4 additional standards/guidelines, namely MASVS [11], IOTSF [7], IOTAA [6] and NISTIR 8259A [9]. We find that *it is infeasible to precisely establish non-compliance according to the letter of the standard, despite vulnerabilities mapping to specific criteria*, for three key reasons. First, certain broad criteria give the appearance of making vulnerabilities non-compliant (i.e., an "effective" standard), while a literal interpretation of the criteria may allow vulnerable code to be justified as compliant ($\mathcal{F}_9$). Second, the test cases elaborated in certain criteria provide significant discretion to testers through ambiguous terminology ($\mathcal{F}_{10}$). Finally, we find that the IOXT standard includes language (such as "*where possible*") that makes essential criteria discretionary for app developers ($\mathcal{F}_{11}$).
- **User survey and qualitative analysis of common perceptions and expectations (RQ$_3$):** We perform a survey with 173 IoT users (denoted as **P1→P173**) to gauge their awareness of IoT security compliance regulations, expectations from certified products and stakeholders, as well as perceptions regarding vulnerabilities, compromises, and accountability. We find that users are generally not informed on IoT security compliance standards ($\mathcal{F}_{12}$), but nevertheless, a overwhelmingly trust that certified apps are secure ($\mathcal{F}_{14}$), in direct contrast to the status quo discovered in Findings $\mathcal{F}_1$→$\mathcal{F}_3$. This belief is not shaken even after users are faced with scenarios of compliance failures ($\mathcal{F}_{16}$). Finally, users hold all stakeholders (i.e., the app developers, CLEFs, and standards bodies) equally responsible for correctly enforcing standards ($\mathcal{F}_{15}$), but hold developers disproportionately more liable for compliance failures ($\mathcal{F}_{17}$).

Our study leads to ***17 key findings ($\mathcal{F}_1$→$\mathcal{F}_{17}$)*** that not only demonstrate certification failures with serious impact on IoT security, but also challenge user perceptions. That is, most users trust the assurance enabled by certification, believing in a *"belt and suspenders"* scenario (**P144**) (Section 4). However, a minority consider it to be *"just red tape"* (**P11**) without additional guarantees, and Sections 2 and 3 show that this assessment matches the status quo. Finally, we discuss (Section 5) how we can change the status quo towards a "belt and suspenders" scenario, through objective checks and balances that validate the security posture of CLEFs, and empower the affected party to reform the product certification loop for IoT.

2 Mobile-IoT App Analysis (RQ$_1$)

To develop a preliminary understanding of the IoT security compliance ecosystem, we identified 30 unique CLEFs licensed to perform assessments for popular IoT standards such as IOTSF [7] and IOTAA [6]. From an analysis of the doc-

umentations and websites of these CLEFs, we observe that 25/30 CLEFs provide certification/assessments for mobile-IoT apps, which is expected, given that mobile-IoT apps serve as the primary UIs for controlling, configuring, and automating IoT devices, and vulnerabilities in them could provide adversaries with privileged access to devices and other components of the holistic IoT system. This importance assigned to mobile-IoT apps by CLEFs, and their general accessibility relative to firmware, motivates our focus. To identify target products for our analysis, we conducted a semi-automated search for certified mobile-IoT apps (see Appendix A.1), and found that the only set of mobile-IoT apps available for our investigation of **RQ**$_1$ is the list from IOXT [32], a popular CLEF as well as an alliance that administers the IOXT standard.

2.1 Methodology

As of January 2023, IOXT certified 32 Android apps, of which, 11 can be classified as mobile-IoT as they directly automate/-control an IoT device or automation platform. We evaluate these 11 mobile-IoT apps, listed in Table 3 in Appendix A. Our analysis focuses on vulnerabilities resulting from cryptographic misuse, followed by additional security-relevant criteria such as over-privilege and data leaks.

Recall that our goal is to uncover gaps in compliance enforcement, and not to measure the state of crypto-misuse in general. Hence, we do not seek coverage of all vulnerabilities, but rather to find (i) common/mundane vulnerabilities that certifiers should have caught, and (ii) highly complex/evasive examples [15] that would be hard to detect. A systematic, manual, analysis is best-suited for these goals, as it allows us to find common vulnerabilities faster without combing through FPs generated by automated tools (e.g., CryptoGuard alone raised 546 vulnerability alarms in the 11 apps, see the online appendix [20]), and, to find complex vulnerabilities that crypto-detectors fail to detect [15]. Further, we use automated tools for tasks that are impractical to perform manually, such as finding multi-hop data leaks to storage or logs. We now describe the core steps of our analysis methodology.

1. Selecting crypto-API misuse cases for analysis: Prior work by Ami *et al.* has developed a large-scale taxonomy of cryptographic API misuse cases, containing 105 such cases [15]. For a tractable analysis, we sampled a set of 10 misuse cases (see Table 7, Appendix A), which satisfy two key criteria: (i) the misuse is commonly found in apps and hence is frequently discussed in literature [15], and (ii) is considered by at least one of the three automated tools that we initially used, i.e., CryptoGuard, CogniCrypt, and MobSF.

2. Vulnerability analysis and confirmation: We disassembled each app using jadx [33], and analyzed it for each of the misuse cases in Table 7. To find vulnerabilities that can be unequivocally said to be in the certified app, versus auxiliary code, we focused on the app's core/main package, instead of the external libraries and SDKs. We searched for the key API invocation(s) specific to each misuse, and confirmed the reachability of the vulnerable API call, validating control flows upto 4 hops back. We considered nuances that may make a misuse not be vulnerable, such as using MD5 for non-security purposes, and did not treat such cases as vulnerabilities.

3. Post-analysis to identify potential impact on IoT: We performed additional analysis to understand the implications of the discovered vulnerabilities, by examining the surrounding code semantics to understand how vulnerable code is used, and what IoT-related function it performs. We also examined the source/destination of the data to determine the significance of the data to IoT security/privacy. We did not establish impact unless we could do so without any doubt, considering code obfuscation and incomplete decompilation. This conservative approach, in a manner similar to the vulnerability analysis itself, led us to make fewer but certain determinations of the potential impact of the vulnerabilities on IoT security.

4. Permission analysis: Criteria such as IOXT SD113 ("Android Permissions Requested - Only Necessary Permissions") require mobile-IoT apps to request only the permissions they need. As IOXT does not define "necessary", and as least privilege is subjective, we adopt a definition from prior work that uses "informed consent" to adjudicate least privilege (e.g., Whyper [50], SmartAuth [57]). That is, we define "necessary permissions" as those that the developer justifies to the user in (1) the app description or (2) its privacy policy (obtained from Google Play or by running the app if needed). For each app, we extracted the runtime (dangerous) permissions using `adb dumpsys`, and compared this set with those described in text resources to identify unjustified/unnecessary permissions.

5. Sensitive data leak analysis: Standards for mobile-IoT apps also describe how sensitive data should not leak to the logs (e.g.,IOXT SD111 - No sensitive data is logged) or external storage (e.g., IOXT SD109: Store sensitive data only within the application container). We define a sensitive data leak as a data flow from a sensitive source, i.e., Android APIs associated with a dangerous permission, to a relevant sink, i.e., android logs and external storage, and leverage FlowDroid [21] to detect such leaks. We customized FlowDroid to focus on the sinks relevant to this analysis (i.e., log and storage APIs), and expanded the sources to include the relevant APIs connected with dangerous permissions. We analyzed each app with a timeout of 24 hours per app, and manually validated the sources/sinks of the reported data leaks/violations, but were unable to validate the flows given heavy obfuscation.

6. Sampling and analyzing non-certfied mobile-IoT apps: Finally, we compare the state of certified vs non-certified mobile-IoT apps to answer a simple question: *are certified mobile-IoT apps more or less vulnerable than a comparable set of non-certified apps?* For this, we repeated the methodology described previously on a comparable set of 11 non-certified mobile-IoT apps, sampled from the 37k mobile-IoT apps developed by Jin *et al.* [34], as elaborated in Appendix A.

2.2 Results: Crypto-API Misuse Analysis

We discovered 35 vulnerabilities stemming from crypto-API misuse in 9/11 certified mobile-IoT apps, counting only one instance of every vulnerability. Table 1 shows the distribution of the vulnerabilities across specific crypto-API misuse cases.

Most vulnerabilities were expressed in their simplest forms, e.g., the use of `AES` without specifying the block chaining mode, which results in `ECB` being used by default, was found in several apps as follows: `Cipher cipher = Cipher.getInstance("AES")`. However, Listing 1 below shows an intriguing use of `ECB` that seems as if it was written to look inconspicuous, to ***evade compliance/quality checks***:

```
//The string operations below result in: "AES/" + "E" +
     "C" + "B" + "/NoPadding" = "AES/ECB/NoPadding".
this.ALGO= "AES/" + ((char)
     ("AES/GCM/NoPadding".charAt(4) - 2)) +
     "AES/GCM/NoPadding".charAt(5) + ((char)
     ("AES/GCM/NoPadding".charAt(6) - 11)) +
     "/NoPadding";
Cipher cipher = Cipher.getInstance(this.ALGO);
```

Listing 1: A complex instantiation of `AES` in `ECB` mode in TUYA, made to look like the `GCM` mode instead.

While prior work has theorized an evasive-developer threat model [15], this paper is the first to report a real instance of such evasive code in mobile-IoT apps.

Finding 1 ($\mathcal{F}_1$) **–** Some mobile-IoT apps evade compliance checks by disguising vulnerable code as compliant, which indicates a serious challenge for CLEFs, and a pressing need to perform a *hostile review* of products.

The app in which several instances of this vulnerability were found, i.e., TUYA, is a major IoT development platform that provides a "no code" development interface and an SDK for IoT developers to build OEM and smart home "hub" apps. Any vulnerabilities in TUYA potentially impact millions, given that its interface/SDK is used by over *580,000 IoT developers*, and its app installed by over *5 million users*.

Upon further analysis, we identified 6 vulnerabilities that clearly impacted a security/privacy-sensitive IoT function or data. Particularly, several instances of vulnerable encryption in TUYA, mainly `AES` in `ECB` mode (including the evasive use above), are used when potentially pairing with BLE devices, or in classes used to enable local communication. We also found that TUYA uses "`AES/CBC/PKCS5Padding`, susceptible to padding oracle attacks [2], to potentially send/store "security alert images" from devices such as security cameras (also confirmed from TUYA's official documentation [4]).

Finding 2 ($\mathcal{F}_2$) **–** Some certified mobile-IoT apps use vulnerable encryption when transmitting/receiving sensitive audio/video data to/from devices such as cameras.

In a similar vein, we found vulnerable implementations of the `TrustManager` interface that accept all certificates. For instance, the MSMARTHOME app (100k+ installs) uses such a `TrustManager` in a key class that is used for authentication and account management, including for tasks such as account creation and user login. While prior work has already demonstrated how mobile-IoT apps may be subjected to man-in-the-middle (MiTM) attacks to steal authentication tokens [37,38]; this set of vulnerabilities takes it further, by enabling attackers to steal user credentials and hijack the account and all associated devices. We found a similarly vulnerable `TrustManager`, and an overridden `HostnameVerifier` that trusts all hosts, used to set up general SSL/TLS connections in WYZE.

Finding 3 ($\mathcal{F}_3$) **–** Some certified mobile-IoT apps override `TrustManagers` and `HostnameVerifiers` in ways that make critical communication for user authentication and account management vulnerable to MiTM attacks.

As we describe in Section 2.5, WYZE confirmed that they were aware of the vulnerabilities, and had addressed them in the new version, as the certified versions are seldom the latest. To validate the possibility that newer versions of apps may contain fixes to vulnerabilities found in certified versions, we evaluated the latest version (as of November 2022) of all 9/11 vulnerable certified apps. Our analysis demonstrates that for the vast majority, i.e., 6/9 vulnerable apps, the newer versions are just as vulnerable, if not more. The only exceptions are DALS CONNECT and EUREKA, for which some (i.e., for DALS CONNECT) or all (i.e., for EUREKA) vulnerabilities were not found in the new versions, and MSMARTHOME app, which moved the vulnerable methods into native code hence could not be similarly analyzed. In WYZE, we found that the developers had simply refactored the code around the vulnerability, without fixing it, which contradicts the vendor's response. Similarly, in NETHOME PLUS, we found that a previously unreachable vulnerability (`MD5`) was now reachable.

Finding 4 ($\mathcal{F}_4$) **–** The latest versions of vulnerable certified mobile-IoT apps are generally similarly or more vulnerable than the (older) certified versions.

Prior to our systematic manual analysis, we had also automatically analyzed the certified apps with CryptoGuard, CogniCrypt, and MobSF, but chose manual analysis for reasons described in Section 2.1. The tools led to 772 alarms cumulatively. Given this volume of alarms, one might expect the tools to find all (or most) of the vulnerabilities discovered using our manual analysis. However, the tools failed to detect several critical vulnerabilities that we independently found and confirmed, even cumulatively detecting only 22/35 vulnerabilities, with none detecting the evasive instance ($\mathcal{F}_1$).

Finding 5 ($\mathcal{F}_5$) **–** Three automated tools, CogniCrypt, MobSF, and CryptoGuard, do not detect several of the 35 critical vulnerabilities discovered using manual reverse engineering, i.e., 33/35, 28/35, and 15/35 respectively, despite generating 89, 137, and 546 alarms, respectively.

Finally, we found 22 vulnerabilities in 8/11 of a comparable

Table 1: Distribution of the crypto-API misuse vulnerabilities in 9/11 vulnerable certified apps.*

No.	*Misuse case*	Tuya	NetHome Plus	Eureka	Midea Air	MSmartHome	GreenMAX	Wyze	Dals Connect	Google Home	Total
1.	Only AES for encryption	✓									1
2.	AES with ECB for encryption	✓				✓		✓	✓		4
3.	AES with CBC for encryption	✓				✓	✓	✓	✓	✓	6
4.	No clearpassword() call after using PBEKeySpec						✓				1
5.	MD5 hashing	✓	✓	✓	✓	✓			✓	✓	7
6.	SHA1 hashing		✓	✓	✓	✓			✓	✓	6
7.	Trusting all certificates					✓		✓		✓	3
8.	Allowing all hostnames					✓		✓		✓	3
9.	"SSL" as context							✓	✓		2
10.	"TLSv1" as context	✓								✓	2
	5	2	2	2	6	2	5	5	6	35	

*We did not find any vulnerability in TSmartLife and Hubspace.

set of non-certified apps, which indicates that overall our set of certified apps is more vulnerable than non-certified apps, given that we previously found 35 vulnerabilities in 9/11 certified apps. However, as shown in Table 5 in Appendix A, for most misuse cases, certified and non-certified apps have a similar rate of vulnerabilities per app, with a few exceptions.

Finding 6 ($\mathcal{F}_6$) – Our equivalent set of certified and non-certified mobile-IoT apps are similarly vulnerable in terms of the crypto-API misuse cases we analyzed for.

We emphasize that $\mathcal{F}_6$ may not generalize to the broader ecosystem of certified and non-certified mobile-IoT apps, given our limited sample, as we also elaborate in Section 6.

2.3 Results: Permission Analysis

Recall that in the absence of a definition of what a "necessary" permission constitutes, we consider a permission necessary if the app justifies it to the user, i.e., leveraging the notion of overprivilege based on informed consent from prior work [50, 57]. We found that *all certified mobile-IoT apps* request at least one dangerous/sensitive-data related permission without describing it in their privacy policy or description, as shown in Table 6 in Appendix A. For example, Wyze requests several, sensitive, permissions, e.g., background location, and reading contacts, while claiming "*No data collected*" in the data safety section of its Google Play listing.

Finding 7 ($\mathcal{F}_7$) – 11/11 certified mobile-IoT apps request at least one dangerous permission that is not justified in the app description or privacy policy.

2.4 Results: Sensitive Data Leak Analysis

FlowDroid reported sensitive data leaks to external storage and logs in all certified and non-certified apps except HUBSPACE and GREENMAX DRC.[1] However, upon manual validation of the sources and sinks in the alerts, we discovered that most were false positives, with only a few, relevant, leaks.

Upon manual validation, we confirmed that 4/11 certified mobile-IoT apps log sensitive data, such as location (MSMARTHOME, TSMARTLIFE, WYZE), the phone's MAC address (EUREKA, TSMARTLIFE), and the phone-specific telephony device ID (TSMARTLIFE). Similarly, of the 11 comparable non-certified mobile-IoT apps, we confirmed that 6 apps leak sensitive data to the log, which is not considerably different from certified apps. Also, outside the scope of the permission-protected API-related definition of a data leak in Section 2.1, we found *user-provided passwords being leaked by* EUREKA *and* MIDEA AIR *to the logs*.

Finally, in contrast to logs, FlowDroid did not throw any alerts for leaks to external storage, i.e., we conclude that neither the certified nor non-certified mobile-IoT apps leak data from sensitive APIs to external storage, unless they leverage implicit flows that are beyond FlowDroid's capabilities.

Finding 8 ($\mathcal{F}_8$) – Both certified and non-certified mobile-IoT apps from our set leak privacy-sensitive data such as location, the device ID, and sometimes the user-provided password, to the logs, but not to external storage.

2.5 Vulnerability Disclosure

We have reported all the vulnerabilities in the certified and non-certified apps to the concerned vendors. The vulnerabilities and our reports are available in our online appendix [20]. WYZE responded to the reports and confirmed that they were aware of the vulnerabilities, but also stated that the vulnerabilities were fixed in the new (non-certified) version. As the analysis of the new version contradicts this assertion, and we have followed up with WYZE, and are waiting on their response. TUYA, i.e., the developer whose code contained the evasive behavior, as well as several other instances of weak encryption, simply marked the severity of our reports as "ignored", but did not provide any explanation or respond to our communication. We are following up with IOXT separately regarding this case, and have filed certification disputes.

[1] GreenMAX DRC could not be analyzed due to its hybrid nature, and timed out for Hubspace

Table 2: A mapping of the vulnerabilities identified to the specific criteria in the standards analyzed.

No.	Vulnerabilities	Apps with Vulnerability	Specific criteria from individual standards that pertain to each vulnerability				
			IoXt	MASVS/ MASA	NISTIR 8259A	IoTSF	IoTAA
1.	Using only AES for encryption	Tuya	PC1	MSTG-CRYPTO-2, MSTG-CRYPTO-3	Data Protection	2.4.7.15	5.1.1
2.	Using AES with ECB for encryption	MSmartHome, Tuya, Wyze, Dals Connect	PC1	MSTG-CRYPTO-2, MSTG-CRYPTO-3	Data Protection	2.4.7.15	5.1.1
3.	Using AES with CBC for encryption	MSmartHome, Tuya, Wyze, Dals Connect, Google Home, GreenMAX DRC	PC1	MSTG-CRYPTO-2, MSTG-CRYPTO-3	Data Protection	2.4.7.15	5.1.1
4 .	No clearpassword() call after PBEKeySpec	GreenMAX DRC	PC1	MSTG-CRYPTO-3	Data Protection	–	5.1.1
5.	Using MD5 hashing	Dals Connect, Eureka, MSmartHome, Tuya, Wyze, NetHome Plus, Midea Air, Google Home	PC1	MSTG-CRYPTO-2, MSTG-CRYPTO-4	Data Protection	2.4.9.5	5.1.1
6.	Using SHA1 hashing	Dals Connect, Eureka, MSmartHome, NetHome Plus, Midea Air, Google Home	PC1	MSTG-CRYPTO-2, MSTG-CRYPTO-4	Data Protection	2.4.9.5	5.1.1
7.	Trusting all certificates	MSmartHome, Wyze, GoogleHome	SI110	MSTG-NETWORK-3	Logical Access to Interface	2.4.13.4	5.1.1
8.	Allowing all hostnames	Wyze, Google Home	SI113	MSTG-NETWORK-4	Logical Access to Interface	2.4.13.10	5.1.1
9.	Using SSL as context	Dals Connect, Wyze	SI110	MSTG-NETWORK-2	Data Protection	2.4.7.19	5.1.1
10.	Using TLSv1 as context	Tuya, Google Home	SI110	MSTG-NETWORK-2	Data Protection	2.4.7.19	5.1.1
11.	Requesting unnecessary permissions	All Apps	SD113	MSTG-PLATFORM-1	Logical Access to Interfaces	2.4.5.1, 2.4.8.9	5.1.1
12.	Logging sensitive data	Eureka, MSmartHome, TSmartLife, Wyze	SD111	MSTG-STORAGE-3	Data Protection	2.4.9.7	5.1.1
13.	Storing sensitive data outside the app container	NONE	SD109	MSTG-STORAGE-2	Data Protection	2.4.9.7	5.1.1

–IoTSF does not contain any information regarding the use of `PBEKeySpec`, and hence, would not consider apps vulnerable to misuse #4 (i.e., GreenMax DRC) in violation.

3 Security Compliance Analysis (RQ_2)

We now know that certified mobile-IoT apps are not free from vulnerabilities, but the question is, *does being vulnerable also make them non-compliant*? (**RQ_2**). This section seeks to address this question more broadly, i.e., not just in the context of IoXt, but by also evaluating whether the apps would be certifiable under other IoT security standards or guidelines. We also explore the criteria further to understand *why*, despite vulnerabilities, app could still argue compliance, leading to findings that motivate improvements in compliance standards.

3.1 Methodology

We select 5 popular IoT security standards (i.e., 4 besides IoXt), namely: (1) The OWASP Mobile Application Security Verification Standard (MASVS) [11], (2) the IoT Security Foundation (IoTSF) standard [7], (3) the IoT Alliance Australia (IoTAA) security guidelines [6], and finally, (4) NIST's Core IoT Cybersecurity Capabilities Baseline (NISTIR 8259A) [9]. Note that we exclude standards that only govern devices (e.g., GSMA's IoT security standard [5]).

To analyze compliance with respect to each standard, we first systematically transformed the relevant parts of the standard into specific *criteria*, and then compared these criteria with the vulnerabilities from Section 2.2. This analysis is composed of two aspects: determining *what* criteria apply to the vulnerabilities, and using additional information (e.g., test cases) to determine *how* they apply.

To elaborate, we systematically deconstructed each standard into specific criteria that apply to mobile-IoT apps, followed by mapping explicit criteria to the vulnerabilities studied in Section 2 (see Table 2 for the mapping). Particularly, we exhaustively reviewed the standard documents, comparing each criterion to the precise nature of the discovered vulnerability in order to find one that applies. Further, only IoXt provides test cases associated with the criteria in its Test Case Library (v5). We used these test cases to further understand *how* the criteria may apply to the vulnerabilities we detected, analyzing the specified test steps and acceptance requirements. We also considered external references where provided, e.g., for "PC1: Standard Cryptography", where the acceptance and additional requirements specify existing guidelines, such as OWASP MASVS 3.1 – 3.6 and NIST.

3.2 Results: Compliance Analysis

Table 2 maps the vulnerabilities and security/privacy issues identified in our analysis to the applicable criteria in each standard. Given that the spirit of the standards is to facilitate secure products, vulnerabilities in apps that map to specific regulatory criteria should technically be sufficient to make the apps non-compliant. However, we discover that establishing non-compliance is not straightforward, as imprecision, ambiguity, and loopholes in standards may enable vendors to argue compliance by the letter, regardless of vulnerabilities.

1. *Overly broad* criteria: Consider IoTAA, which provides a single, broad, criterion, for the use of cryptography, as follows: *"Ensure devices and associated applications support current generally accepted security and cryptography protocols and best practices. All personally identifiable data in transit and*

in storage must be encrypted using current generally accepted security standards." The NISTIR 8259A baseline is just as broad. It may be somewhat acceptable (although not ideal) for both IoTAA and NISTIR 8259A to be this broad, as they are not standards, but serve as guidelines for the development of more precise standards. Indeed, the MASVS and IoTSF standards are more precise, and discuss specific lists of governmental sources of secure/deprecated ciphers to use/avoid. However, the same cannot be said of IOXT.

To elaborate, in IOXT, the key requirement that governs the use of cryptography is "standard cryptography (`PC1`)". Under `PC1`, apps are required to verify that: (i) "*all choices of cryptography are supported by appropriate industry or government recognized standards or best practices...*", and (ii) "*all choices of cryptography have not been deprecated or reported to be insecure by industry or government organizations*", with the later referring to OWASP MASVS's criteria 3.1−3.6. The overly broad language allows IOXT to claim completeness without being as precise as MASVS and IoTSF, as essentially every vulnerability could be argued as against a recognized best practice. However, we observe that if this broad language is taken literally, it could also allow a developer to argue that certain vulnerable uses are compliant. For example, one might literally interpret `PC1` to argue that `Cipher.getInstance("AES")` is compliant, as this specific invocation is not deprecated, and "AES" is a standard "choice of cryptography", regardless of the eventual vulnerable outcome (i.e., the call results in `ECB` mode by default).

Finding 9 ($\mathcal{F}_9$) – Certain standards criteria are overly broad, making them appear comprehensive. However, a literal interpretation of the same may help developers claim vulnerable code as compliant.

2. *Ambiguous* test cases, *discretion* to testers: Criteria relevant to data security are more precisely defined in IOXT, relative to the broadly described PC1. That is, rather than just broadly requiring apps to collect and process data "as per industry best practices", IOXT further elaborates on what this correct handling should entail, with specific criteria SD113 ("Android Permissions Requested - Only Necessary Permissions"), SD111 ("No sensitive data is logged"), and SD109 ("Store sensitive data only within the application container"). While this increased precision is welcome, we observe that the actual test cases for these specific criteria contain ambiguous language that leaves significant discretion to testers.

For example, for SD113, the tester may accept the app if it "does not request excessive sensitive permissions". However, defining *excessive*, whether quantitatively (i.e., "how many extra permissions is excessive") or qualitatively (i.e., "what permission is unnecessary") is entirely left to the tester. Recall that our analysis defines unnecessary permissions as those not justified to the user, and identifies several apps that request such permissions. However, given that lack of clarity on what "excessive sensitive permissions" entails, it is infeasible to unequivocally deem *any* of the apps non-compliant.

Similarly, SD111 and SD109 do not specify exactly what data is to be considered "sensitive", leaving that aspect to the tester's discretion. Recall that aside from private data such as location and MAC addresses, we also found potential leaks of authentication credential to the logs ($\mathcal{F}_8$), in certified apps. Note that the OWASP MASVS guidelines, which IOXT often refers to, do provide a definition of sensitive data as generally within the bounds of "user credentials and private information", including all information "that must be protected by law or for compliance reasons" [10]. However, IOXT does not require the tester to adhere to this definition, only referring to OWASP MASVS broadly under "additional information". Thus, here too, the ambiguous language prevents us from firmly establishing (non) compliance.

Finding 10 ($\mathcal{F}_{10}$) – Test cases accompanying criteria contain ambiguous phrases (e.g., "excessive permissions"), allowing significant discretion to the tester, preventing an unequivocal determination of compliance.

3. *Loopholes* in the criteria, *discretion* to developers: In contrast to the broad requirements in PC1 (standard cryptography), we find that the correct use of SSL/TLS is clearly specified and in precise detail in IOXT, as a part of the `SI110` criteria, which directly maps to several of the vulnerabilities we studied (i.e., vulnerability #7, #9, #10 in Table 2). The criteria discusses several precise checks (e.g., validating certificate expiration, hostnames). Such precise description is ideal as it makes it easier for developers to adopt and implement the standard, while also making it straightforward for CLEFs to perform checks. However, the following language used to describe how developers should comply with this criteria reduces its weight, by making it largely optional: *Encrypt all network traffic, using verified TLS where possible.*

The last phrase, i.e., ***where possible***, leaves complying with the criteria to the developer's discretion. That is, the developer may consider using SSL/TLS unnecessary/impractical for *any* communication, and yet consider their code to be compliant, as the criteria does not elaborate what "possible" entails.

Finding 11 ($\mathcal{F}_{11}$) – IOXT makes certain precise criteria discretionary for developers to comply with, leaving developers with the flexibility of choosing what communication or data to protect, which may result in vulnerable apps that developers may contest are compliant with the standard.

4 User Perceptions and Expectations (RQ_3)

Recall that this paper seeks to investigate whether the traditional compliance model *works as well as consumers expect it to*. To this end, we conduct a user survey that allows us to explore consumer perceptions on compliance enforcement.

4.1 Methodology

Our methodology is guided by several questions that intuitively emerge when we consider user perceptions regarding compliance enforcement. For instance, what do consumers know and believe about compliance enforcement? Who do they find responsible for enforcing standards correctly? And, who would they hold accountable if things break down? The survey design reflects these questions.

4.1.1 Survey Design

Our survey consists of questions regarding participants' familiarity and experience with mobile-IoT apps, IoT security compliance regulations, their enforcement, and their beliefs and opinion about specific scenarios related to IoT compliance. We constructed 39 questions with a mix of 15 open-ended and 24 close-ended questions, organized as follows (see the online appendix [20] for the survey instrument):

Experience with mobile-IoT apps: Participants were asked about their familiarity and experience with mobile-IoT apps. Further, we provided the list of certified mobile-IoT apps studied in this paper and asked participants about their familiarity with those specific apps. Participants were also given the opportunity to list additional mobile-IoT apps they had used.

Familiarity with IoT security compliance standards: We asked participants if they were familiar with a list of well-known *general*, i.e., non-IoT compliance standards used in the software industry. We then asked them about their familiarity with the IoT security compliance standards studied in Section 3. Lastly, we elicited participants opinions about the entities they think are responsible for the correct enforcement of IoT compliance standards, among the key stakeholders.

Expectations from certified apps and scenario-based questions: After obtaining their general perceptions regarding compliance enforcement, participants were presented with questions regarding their expectations from certified apps. Particularly, we asked participants about their opinions regarding the prevalence of vulnerabilities, developer diligence regarding data security, and the amount of trust they would place when it came to their confidential/private data, in certified vs non-certified apps. We then posed 2 hypothetical scenarios inspired from the vulnerabilities discovered in the certified mobile-IoT apps from our analysis in Section 2, and asked participants to rate the severity and likelihood of these scenarios, as well as the stakeholder(s) that would be liable if the scenario happens in practice. We then again asked them for their preference towards certified vs non-certified apps.

4.1.2 Participant Recruitment

We invited 425 IoT users to participate in our survey through Prolific [12]. Our participant screening criteria included the following: participants must (i) be at least 18 years of age, (ii) reside in the USA, (iii) self-report fluency in English, and (iv) be IoT users (i.e., previously used IoT devices). We received a total of 180 responses, of which we discarded seven responses due to failed attention check questions or finishing the study too quickly (i.e., within 5 minutes in our case, as per Prolific's general guidelines [52]). Our survey took an average of 15 minutes to complete. We paid each participant $5.

4.1.3 Coding and Analysis

We used reflexive thematic analysis to analyze our free-text responses [23]. We used the inductive coding approach to derive codes from open-ended question response data. At first, two coders randomly selected a subset of responses (20-30%) per question for familiarization and to derive preliminary codes. These codes were applied to the selected responses by both coders to resolve differences and for evolving the preliminary codes. Next, the coders split the data equally and applied the codes separately on the split data. During this step, the coders discussed any response that could not be labeled using any of the existing codes and updated the codebook. Finally, the authors held an agreement-disagreement discussion to reach a consensus for each coded response.

4.1.4 Ethical Considerations

Our survey protocol was approved by our Institutional Review Boards (IRB). Participants were informed about the goal of the study before participating, and willingly provided consent to participate, including the consent to disclose anonymized survey responses and quotes. In scenario-based questions where we presented information from our analysis, we anonymized application names and other details, and presented the scenarios as hypothetical, in order to prevent and adverse impact on the apps studied, and in the spirit of responsible vulnerability disclosure.

4.2 Results: User Perceptions

This section discusses the results from our analysis of 173 survey responses. We organize the results across four key themes that we observe: (i) the lack of exposure to compliance standards, (ii) the overwhelming confidence in certification over brand reputation, (iii) the view that all stakeholders except users (i.e., developers, CLEFs, standards bodies) are equally responsible for compliance enforcement, and (iv) the disproportionate blame assigned to developers in case of failures. As we elaborate in Section 6, our qualitative analysis and results draw from self-reported data, which may suffer from biases and a lack of generalizability.

4.2.1 Lack of Exposure to Compliance Standards

We find that participants are generally familiar with mobile-IoT apps, as illustrated in Figure 2. However, participants do not demonstrate familiarity with compliance standards, be they for general software or for IoT security compliance, as shown in Table 8 in Appendix C. Particularly, only 19 participants know of at least one of the listed IoT compliance standards, be it through online materials (8), professional settings (5), personal connections (3), and/or coursework (3).

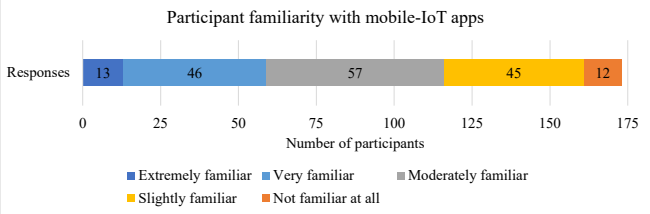

Figure 2: Participant familiarity with mobile-IoT apps

As **P99** states, *"I learned it from google search...aware about from my colleague and then I further looked into it."*

Moreover, since most participants are not familiar with compliance standards, their perspectives are mainly based on their common sense, as **P12** states: *"Common sense is using one that is certified secure."*

When we asked participants whether they had used any mobile-IoT apps that were certified with any of the previously mentioned IoT security standards, 128 (73.99%) participants stated they had not, and 17 (9.83%) were unsure. The rest (i.e., 28 or 16.18%) mentioned several apps they had used that they knew were certified. Of these, 19 mentioned apps that are indeed certified, i.e., 17 mentioned using Google Home (certified by IOXT), 2 mentioned Wyze (certified by IOXT), and 4 mentioned Google Nest (certified by MASA). However, other participants mentioned apps that are not certified by any extant IoT standard to our knowledge, e.g., Ring (2), Alexa(2), Echo (1), Smart Life (1) and Siri (1).

Moreover, when we provided participants with our list of certified apps from IOXT without explicitly telling them that they were certified, participants indicated having used the apps in much larger numbers, as shown in Table 9 in Appendix C. For instance, 86 participants mentioned using Google Home when we provided the list without mentioning certification, whereas previously, only 17 had mentioned Google Home when asked about certified apps they had used.

Finding 12 ($\mathcal{F}_{12}$) – Users are generally not informed of IoT compliance standards, and often unaware of the certified (status of the) mobile-IoT apps they use.

This clear lack of exposure can be attributed to either the fact that IoT compliance standards are still in the early stages of adoption, or, because CLEFs and vendors may not want users to incorporate compliance in their decision-making process, only relying on it as a liability shield; e.g., only 2/11 mobile-IoT apps certified by IOXT (i.e., GOOGLE HOME, TUYA) disclose their certification to users.

4.2.2 Trust in Certification over Brand Reputation

Users have traditionally focused on brand reputation and third-party product reviews while choosing a certain product [26]. Particularly for mobile-IoT apps and IoT products in general, until very recently there was no avenue such as certification to evaluate products, and hence, criteria such as brand image and reputation were the only ones available to consumers.

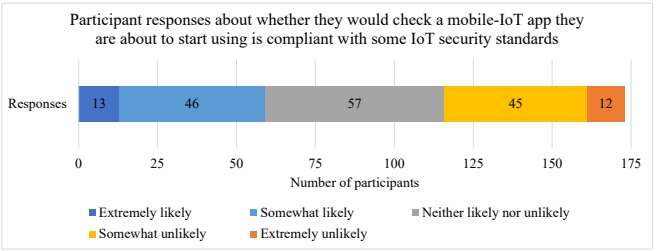

Figure 3: Participant responses about whether they would check if a mobile-IoT app they are about to start using is certified

When asked if they would check whether mobile-IoT apps were in compliance with some IoT security standard/certification before use, 74 (42.77%) participants indicated that they would not be likely to check, generally prioritizing brand reputation and third-party product reviews over certification when assessing an app's security and safety. As **P9** states, *"If the app is from a reputable company and has good reviews I wouldn't have second thoughts."*. Moreover, participants also stated that they rely on popularity and reviews because they are not aware of compliance standards and processes, as **P41** states: *"I wouldn't know where to start looking for this information or how to interpret it. I would instead trust reviews or I guess expert opinions."*

On the other hand, an equal number of the participants (74 or 42.77%) indicated that they are at least somewhat likely to check, mainly because doing so would provide them with additional assurance, i.e., as **P69** puts it, *"...I need some reassurance that the devices and apps I am using are going to keep my information safe."* However, some participants who are somewhat likely to check also focus brand reputation. As **P125** states, *"I generally look, but try to get devices from more trusted brands."* Figure 3 shows the full distribution of the results for this question.

Finding 13 ($\mathcal{F}_{13}$) – While a significant number of users are likely to check the certification status of the mobile-IoT apps they use, mainly for additional assurance, an equal proportion believe brand reputation and popularity to be more valuable.

Further, when presented with scenarios that required them to choose between a certified or non-certified app with identical functionality, an overwhelming majority of participants put their trust in certification and assumed the certified app to be (1) more secure and (2) better designed. To elaborate, when asked whether they considered certified apps or non-certified apps to be more secure (i.e., free of vulnerabilities), 156 (90.17%) chose certified apps, only 1 chose non-certified apps, 6 (3.47%) considered both as equally secure, and 10 (5.78%) considered neither as secure. Similarly, when asked what they would assume regarding the effort taken by developers toward security and protection of user data, an overwhelming majority 127/173 (73.41%) assumed the developers of the certified app to take relatively more efforts, only 1 chose

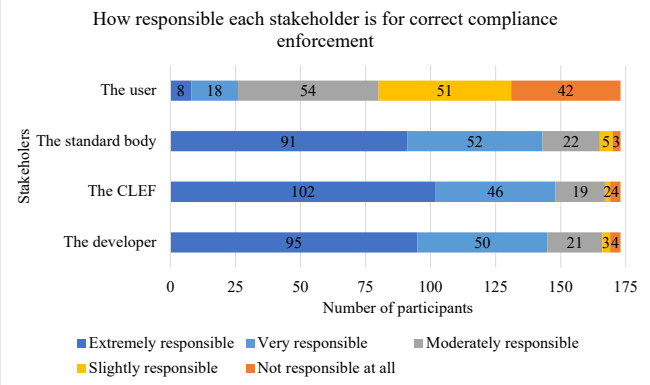

Figure 4: Attribution of responsibility in compliance enforcement

the developer of the non-certified app, 35 (20.23%) assumed that both spent equal effort, while 10 (5.78%) assumed that neither spent any effort. Moreover, when asked to select a certified vs non-certified app in a scenario that required the app to process sensitive information, a majority of participants, i.e., 163 (94.22%) chose the certified app, *nobody chose the non-certified app*, 7 (4.05%) indicated that they would not care about certification, while 3 (1.73%) chose not sure.

Finding 14 ($\mathcal{F}_{14}$) **–** Users overwhelmingly put their trust in certification, assuming that (1) certified apps are more secure (i.e., less prone to vulnerabilities), (2) their developers spend more effort on security, and (3) they can be trusted to handle security/privacy sensitive information.

The rationale for placing this level of trust in certified apps may vary. Several participants desire certified apps for the assurance they offer, e.g., as **P31** states, *"I would prefer the certified IoT apps simply because of the extra peace of mind."* Others attest their preference for the third-party verification and extra focus on security that is expected from the certification process. For instance, **P83** states that *"Certification is an endorsement by a third-party regulating org to assure that the standards are up to compliance. A non-certified has no objective party to endorse its security."*, while **P22** states that *"A certified app shows that the developers of the app placed at least a decent amount of effort on security, and that would bias me toward that app's security features and capabilities."*

4.2.3 All Stakeholders (except users) are Responsible for Proper Enforcement

As shown in Figure 4, participants hold application developers, CLEFs, and standard organizations as somewhat equally responsible for enforcing correct compliance standards.

CLEFs are most responsible (by a small margin): As the CLEFs' entire value proposition is in ensuring that products meet a standard, most participants (102 or 58.96%) hold CLEFs directly responsible for enforcing the compliance standards correctly. As **P23** succinctly states, *"The certification labs should also be on the hook because if they don't enforce their own certifications, what good are they."*

Developers are responsible too: Out of 173 participants, 95 (54.91%) participants considered application developers extremely responsible whereas only 4 (2.31%) participants think developers are not responsible at all for enforcing correct compliance standards. There are two common themes that make developers responsible. The first and most common reason for considering developers responsible is that they develop the app and control it. As **P95** states, *"I'd view any security issues with IoT devices to be primarily a result of the developers' programming..."* The second theme that emerges is that developers should meet certain security standards to ensure the apps' and users' safety and security. That is, as **P13** puts it: *"The developer must take into account the compliance standards...when designing the app and considering how data is used... especially to prevent compromise..."*

Standards organizations are responsible, although less than CLEFs and developers: Participants believe that the standard bodies should ensure that the compliance standards are comprehensive enough, and that CLEFs and developers are accountable. As **P77** states, *"If there are going to be certification authorities, then they need to be responsible and accountable for the products that get their 'seal of approval'"*.

Users also stated that they would consider a certified app secure only if they are able to trust the standards organization itself, regardless of the actual app, i.e., as **P85** states: *"I would trust an FDA-approved vaccine much more than one that wasn't"*. This sentiment is also exhibited in the significant trust that users place in the compliance infrastructure ($\mathcal{F}_{14}$).

Finding 15 ($\mathcal{F}_{15}$) **–** Users hold CLEFs, developers, and standard organizations as almost equally responsible, and are able to clearly define what role each party plays in correctly enforcing security compliance standards.

4.2.4 Developers Are Mostly to Blame

In the survey, we defined liability as the attribution of blame when things break down, i.e., when security failures occur. We presented participants with two such scenarios inspired from Section 2.2. In one scenario, we presented the case of sensitive information leakage from an app (inspired from TUYA, $\mathcal{F}_2$), and in another we discussed the attacker being able to steal authentication tokens and hijack IoT devices (inspired from MSMARTHOME, $\mathcal{F}_3$). A majority of the participants, i.e., 90.17% on average, expressed that these scenarios were critical/high in severity, with only 1.45% on average, assessing them as low/no severity. Similarly, i.e., 71.10% on average, assessed the scenarios as somewhat or extremely likely, with only 13.88% on average, assessing them as unlikely. Tables 11 and 12 in Appendix C provide the per-scenario results for severity and likelihood, respectively.

Finding 16 ($\mathcal{F}_{16}$) **–** Users rate a set of vulnerabilities found in our analysis to be both severe and likely.

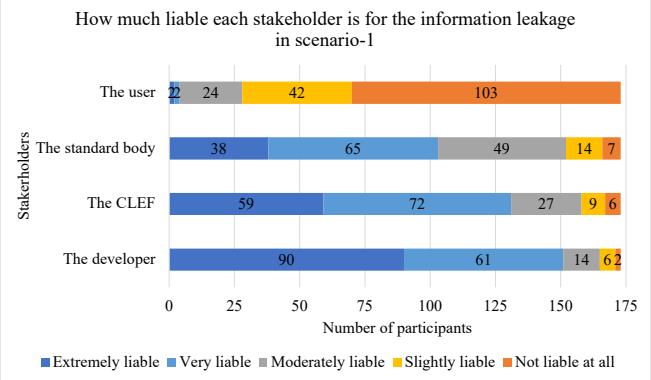

Figure 5: How liable is each stakeholder for information leakage?

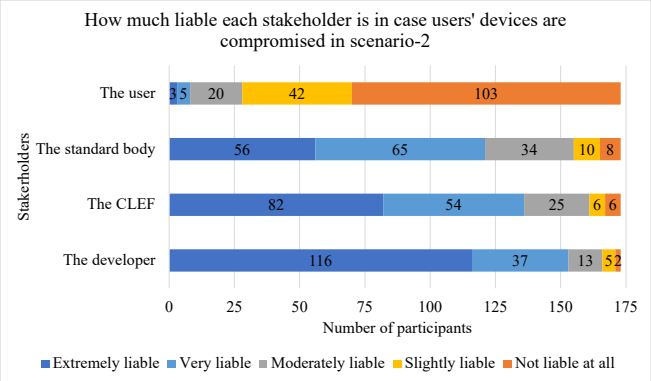

Figure 6: How liable is each stakeholder for compromise?

When we asked participants about who they would hold liable for each scenario, participants overwhelmingly expressed that developers were much more liable relative to CLEFs, standards organizations, or users, as seen in Figure 5 and Figure 6 for the two respective scenarios. **P32** summarizes this sentiment as follows: *"The developer is most responsible for the safety and security of the user, and the certification lab is at fault just as much because their certification should not have dangerous cracks in the infrastructure that allow something like this to happen. The standards body is liable as well by not vetting the certification lab as well as they should. The user is just a pawn and a victim in this scenario."*

Developers are most liable: Application developers are considered liable for handling all vulnerabilities as well as not testing the apps before release, i.e., as **P53** states, *"...definitely the ones liable for not finding the fault beforehand."*

CLEFs are liable because vulnerabilities imply dereliction of duty: Participants hold CLEFs liable as they certified a vulnerable app, i.e., as **P74** states, *"...it is the certification labs job to make sure this does not happen in the first place!"*.

Standard organizations are liable for lack of oversight: Participants consider standards bodies liable, sometimes even more so than developers, because they view vulnerabilities as the result of weak standards or enforcement. As **P69** puts it, *"the developer is liable for a lack of oversight into secure data protocols. but the standards body and certification labs are even more liable because they did not set nor enforce a security standard that was high enough."*

Users are not liable: Participants consider users to be the only stakeholder that can be assigned the least liability for security breaches and vulnerabilities in certified products, as users have no role in the compliance infrastructure, and may lack the technical knowledge to play any.

Finding 17 ($\mathcal{F}_{17}$) **–** Users generally hold developers as the most liable in the event of vulnerabilities and security breaches in certified mobile-IoT apps, since they develop the vulnerable code. CLEFs are considered the second most liable as vulnerable-but-certified apps represent a dereliction of duty, followed by standards bodies who are blamed for weak standards or enforcement.

Finally, even after presenting scenarios where vulnerabilities were found in certified apps, most participants expressed their interest in certified apps over non-certified apps (see Table 10, Appendix C), indicating that users would trust stakeholders to responsibly uphold the standards, and echoing $\mathcal{F}_{14}$.

5 Discussion

We began this study with a simple question: *does the traditional security certification model work for IoT, and as well as consumers expect it to?* Given the evidence and findings from Section 2→4, the answer to this question is a resounding *no*. That is, while consumers have very high expectations and place a significant degree of trust in the certification process ($\mathcal{F}_{14}$), we find that certified mobile-IoT apps are generally vulnerable ($\mathcal{F}_1$, $\mathcal{F}_4$), with vulnerabilities that seriously impact IoT security ($\mathcal{F}_2$, $\mathcal{F}_3$), and no better than non-certified apps ($\mathcal{F}_6$) regardless of what consumers want to believe. While the vulnerabilities map to precise criteria in IOXT, we find that it is hard to establish (non) compliance despite vulnerabilities given the broad criteria ($\mathcal{F}_9$), and the discretion offered to both testers ($\mathcal{F}_{10}$) and developers ($\mathcal{F}_{11}$), indicating a clear failure in compliance enforcement.

To summarize, while a majority of the surveyed users expect security compliance to work for IoT in the form of an additional layer of security assurance, i.e., as **P144** puts it, in a *"belt and suspenders scenario"*, some hold a more dismal belief that certifications are *"just red tape"* (**P11**), and don't provide any added value for security. Given our findings, the skeptics seem to be winning this argument. To avoid a future where IoT compliance enforcement is simply treated as a liability shield, we seek to initiate a timely conversation in the security community, between researchers, practitioners, and policymakers, on how to *transition from the "just red tape" status quo to a practical "belt and suspenders" future.*

1. The Traditional Model is Here to Stay: Given our finding, it might be tempting to conclude that as the traditional model does not seem to work, we should move to a more direct approach that involves *hostile* reviews of products by the

affected party itself (i.e., regulators, on behalf of consumers). Indeed, such models have been tested before, e.g., the US Government's Trusted Computer Systems Evaluation Criteria [60], colloquially known as the *Orange Book*. However, the *Orange Book*'s centralized reviews required significant manual effort, and took months to complete, which limited their scalability and practicality [45], and gave way to the *indirect* compliance enforcement for commodity software, i.e., the traditional model we see today.

As compliance enforcement for millions of IoT products is clearly a matter of scale, reverting to an orange book-style model would be infeasible. That is, we need to delegate enforcement to CLEFs for scalability. Thus, instead of replacing the traditional model, we should explore ways to reform it through effective checks and balances.

2. Reforming the Traditional Model: One way to reform the traditional model is through auditing certified products, similar to the approach used in this paper. However, while necessary to motivate change, auditing products is not a preventative approach, and may not bring long-term benefit. Instead, we must empower the affected party with tools that enable it to radically alter the incentive structure ingrained in the compliance infrastructure, *by empirically validating the claimed security posture of the pivotal participant in the compliance enforcement loop: the CLEF*.

We foresee a future where researchers develop *tools that evaluate the CLEFs' effectiveness at detecting vulnerabilities*, by adapting recent techniques for evaluating vulnerability detection tools [15–17, 22]. Such tools will help regulators evaluate a CLEF 's performance during the license-granting process, thereby incentivizing improvement. Similarly, vendors shopping for CLEFs will leverage the tools to obtain objective measures of the CLEFs' abilities, in contrast with the subjective "reputation" available presently. Finally, practitioners in the compliance industry will be able to empirically evaluate and improve CLEFs, making them robust against rogue vendors that try to evade checks (see $\mathcal{F}_1$).

3. Robust Vulnerability Detectors: As discussed in Section 2, it is likely that most CLEFs use automated tools along with manual analysis in order to analyze products at scale. However, we find that automated static analysis tools are unable to find a significant number of the critical vulnerabilities we discovered using reverse engineering ($\mathcal{F}_5$), which highlights a key gap: the lack of tools suitable for the security-critical task of compliance enforcement. For effective compliance, we need tools that prioritize vulnerability discovery, i.e., seek to avoid false negatives, over other criteria.

4. Holding Developers Accountable through Policy and Analysis: Evading certification checks is not just a matter of cheating CLEFs, but given the high degree of trust put by users in certified apps and the certification process, it is effectively a betrayal of public trust in the developer and their product. As a result, mechanisms must be built into the certification model to deter, prevent, and detect such behavior. Policymakers and regulators have an important role to play here: compliance regulations must clearly describe such behavior and the penalties awarded for it *in addition to* those for non-compliance only. Moreover, regular audits of random samples of products by researchers, similar to the one performed in this paper, will also help identify unwanted behavior. Given how users mostly blame developers when compliance failures happen ($\mathcal{F}_{17}$), such measures would receive consumer support.

5. Bringing the User in the Loop: A key limitation in the existing model is that the user generally remains outside the security certification loop, and often unaware of its very existence ($\mathcal{F}_{12}$). Given the trust users place in the compliance infrastructure, informing users is critical. Several users remarked that while they were unaware of IoT compliance prior to taking the survey, they were now intrigued and would try to learn more about it. As **P109** states, *"Before this, I didn't even know that these things would be certified or have a standard. Now that I know, it makes sense to see what future products meet the standard before I start using them."* If a survey can pique user interest and cause participants to seek out more information, we believe that dedicated consumer education programs (e.g., those recommended in NIST-CSWP-02042022-2 [13]) can change this status quo, by dissuading users from blindly trusting compliance, helping them make informed assessments of certifications, and helping them challenge vendors/CLEFs upon observing inadequacies.

6 Threats to Validity

The value of this work is in obtaining early insights into vulnerabilities in certified IoT apps, and user perceptions of certification. Given this focus, the findings should be considered in view of the following threats:

1. Generalizability: We focus on IOXT-certified mobile-IoT apps, which form the only publicly available set at this time. IoT apps certified by other CLEFs may certainly exist, even if not publicly advertised, and our findings may not generalize to them (or mobile-IoT apps in general). However, our methodology is sufficiently repeatable to be applied to more certified apps, as and when they become available.

2. Exploitability: Although we confirmed vulnerabilities statically (e.g., validating reachability, ignoring non-security crypto-API misuse), we did not confirm the *exploitability* of these vulnerabilities in practice. Further, our analysis only estimates the use case/impact, given the significant obfuscation. Regardless, some of our vulnerability reports have received vendor confirmation (see Sec. 2.5), and we believe that the presence of vulnerabilities in certified mobile-IoT apps is of significant concern, regardless of exploitability.

3. Self-reported survey responses: Generally, survey responses may be biased due to self reporting factors, e.g., over- and/or under-reporting, sample bias, and social desirability bias. While we mitigated this by asking the participants to pro-

vide examples based on their experience, opinions on concrete scenarios, and to elaborate their response through open-ended questions, we acknowledge that these characteristics of our survey responses form a threat to the validity of the findings.

4. Lack of demographic information: Although our survey participants are from the USA, we cannot say that the findings from the survey are generalizable to the population of the USA or outside, as we did not collect several aspects of demographics, such as gender, age and educational background.

7 Related Work

This paper is the first to analyze *certified* mobile-IoT apps, and juxtapose the findings of the vulnerability analysis with qualitative findings from a user study on compliance. Our study is related to prior work in the following areas:

Security Analysis of Mobile-IoT Apps: In terms of the vulnerabilities targeted and techniques used, our work is close to prior security evaluations of mobile-IoT apps [25, 36–38, 61]. A significant distinction is that prior studies simply demonstrate developer mistakes or at most violations of app-store policies [37, 42], we analyze *certified* apps, and hence, the findings of our analysis demonstrate systemic failures involving CLEFs, developers, and standards bodies. Moreover, we posit these failures in the context of user expectations and perceptions, leading to key takeaways that motivate a change in the compliance enforcement infrastructure.

Security Compliance Standards and Processes: Stevens *et al.* recently performed qualitative analyses of the issues in the controls established in digital compliance standards and the challenges faced by organizations in practice [54, 55]. We deviate from this work in terms of the goal: we seek to study *what consumers perceive and expect* from *product compliance certifications*, whereas Stevens *et al.* focus more on organizational compliance with digital security standards, as well as on *how organizations comply*. That said, our general approach towards the analysis of user expectations is inspired by the methodology leveraged in this body of work, although we leverage large-scale surveys of common users as an instrument of collecting data, in contrast to a more focused approach of recruiting experts in prior work.

Building Trust via Certification: Prior work emphasizes the role of third-party certification in building consumer trust [48], indicating that consumers may spend more on certified products [35]. Furthermore, a study conducted by Emami-Naeini *et al.* with IoT consumers showed that consumers are willing to pay more for IoT products with better security and privacy practices [28], which complements our finding ($\mathcal{F}_{14}$) regarding the trust consumers place in certified products. In addition to prior work, our findings demonstrate a blind belief in IoT security compliance ($\mathcal{F}_{12} - \mathcal{F}_{14}$) and provide novel insights into user perceptions regarding perceived responsibility and liability ($\mathcal{F}_{15}$, $\mathcal{F}_{17}$).

Communicating Security Compliance/Certification: Complementary to our work, prototype-based prior studies have explored effective design factors, placement considerations, and delivery mechanisms of *security labels* with the goal of maximizing the informative and educational value of security compliance/certification [27, 29, 39, 44]. For instance, Emami-Naeini *et al.* proposed privacy and security labels for IoT devices that may affect consumers' purchase decisions [27]. Moreover, Kelly *et al.* showed that permissions and privacy information may positively influence consumers' decisions about purchasing apps [39]. While our study demonstrates that users prefer certified products over non-certified ones, we show that certified apps may be over-privileged ($\mathcal{F}_7$) or vulnerable ($\mathcal{F}_1 - \mathcal{F}_4$), and users may hold CLEFs, developers, and labs responsible for the lackluster enforcement of compliance standards ($\mathcal{F}_{15}$). Moreover, several countries, such as Singapore, Germany, and Finland, have introduced independently verifiable cybersecurity certificates with cybersecurity labels [3], which the USA is expected to introduce as well, as per EO 14028 [1, 30]. Given this background, our study demonstrates severe gaps in compliance *enforcement* ($\mathcal{F}_9 - \mathcal{F}_{11}$), a prerequisite for effective labels, and hence, complements prior work towards informed, transparent, compliance.

Non-IoT Compliance Analysis of Mobile Apps: A recent body of work, including but not limited to Cardpliance [46], PolicyLint [18], and PoliCheck [19], analyzes code and text artifacts against security/privacy regulations such as PCI DSS [14], GDPR [51], and CPRA [41]. A key difference between the products/mobile-IoT apps we target versus prior work is that our targets *explicitly claim* to be compliant, i.e., claim to offer a higher degree of assurance, which users also generally believe. Thus, the implications of analysis in the two domains are vastly different, although they yield complementary benefits for end-users: our analysis and findings motivate reform in the compliance ecosystem, whereas prior work motivates better tooling to enable secure apps.

8 Conclusion

This paper presents the first investigation into the initial artifacts from IoT compliance certification, and user expectations and perceptions of it. It presents a vulnerability analysis of 11 certified mobile-IoT apps, with 5 popular standards, and a user study with 173 participants, that together lead to 17 key findings. The findings demonstrate significant failures in compliance enforcement in the form of certified apps with critical vulnerabilities, problems with the standards themselves, as well as the lack of general user awareness of compliance, all of which are contrasted by the overwhelming trust that users place in compliance certifications and certified products. The findings motivate an increased focus on empowering regulators and consumers to reform compliance enforcement through objective checks and balances, starting with techniques to evaluate the performance of CLEFs.

9 Acknowledgements

We would like to thank our shepherd and the anonymous reviewers for their constructive feedback on the paper. The authors have been supported in part by the NSF-2237012 grant, and a COVA CCI Dissertation Fellowship. Any opinions, findings, and conclusions expressed herein are the authors' and do not reflect those of the sponsors.

References

[1] Certification Mark – U.S. Cybersecurity Labeling Program for Smart Devices. https://www.fcc.gov/cybersecurity-certification-mark. Last accessed on July 15, 2023.

[2] Cipher is susceptible to Padding Oracle. https://find-sec-bugs.github.io/bugs.htm#PADDING_ORACLE. Last accessed on February 05, 2023.

[3] Cybersecurity Labelling Scheme (CLS). https://www.csa.gov.sg/our-programmes/certification-and-labelling-schemes/cybersecurity-labelling-scheme.

[4] Encrypted Images-IoT App SDK-Tuya Developer. https://developer.tuya.com/en/docs/app-development/encryptimage?id=Ka6nxw2hetr2y. Last accessed on February 05, 2023.

[5] GSMA IoT Security Guidelines Endpoint Ecosystem Version 2.2 29 February 2020. https://www.gsma.com/iot/wp-content/uploads/2020/05/CLP.13-v2.2-GSMA-IoT-Security-Guidelines-for-Endpoint-Ecosystems.pdf. Last accessed on January 17, 2023.

[6] IoTAA Security Guideline V1.2 November 2017. https://www.iot.org.au/wp/wp-content/uploads/2016/12/IoTAA-Security-Guideline-V1.2.pdf. Last accessed on February 05, 2023.

[7] IoTSF IoT Security Assurance Framework Release 3.0 Nov 2021. https://www.iotsecurityfoundation.org/wp-content/uploads/2021/11/IoTSF-IoT-Security-Assurance-Framework-Release-3.0-Nov-2021-1.pdf. Last accessed on February 05, 2023.

[8] Mobile Application Security Assessment 1.4 January 2022. https://github.com/appdefensealliance/ASA/tree/main/MobileAppSecurityAssessment. Last accessed on February 05, 2023.

[9] NISTIR 8259A IoT Device Cybersecurity Capability Core Baseline. https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8259A.pdf. Last accessed on January 31, 2023.

[10] OWASP MASVS V2: Data Storage and Privacy Requirements. https://mobile-security.gitbook.io/masvs/security-requirements/0x07-v2-data_storage_and_privacy_requirements. Last accessed on July 15, 2023.

[11] OWASP Mobile Application Security Verification Standard v1.4.2 January 2022. https://mas.owasp.org/MASVS/. Last accessed on January 2, 2023.

[12] Prolific. https://www.prolific.co/. Last accessed on February 05, 2023.

[13] Recommended Criteria for Cybersecurity Labeling for Consumer Internet of Things (IoT) Products. https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.02042022-2.pdf. Last accessed on July 15, 2023.

[14] The Payment Card Industry Data Security Standard. https://www.pcisecuritystandards.org/.

[15] Amit Seal Ami, Nathan Cooper, Kaushal Kafle, Kevin Moran, Denys Poshyvanyk, and Adwait Nadkarni. Why Crypto-detectors Fail: A Systematic Evaluation of Cryptographic Misuse Detection Techniques. In *IEEE Symposium on Security and Privacy (S&P)*, April 2022.

[16] Amit Seal Ami, Kaushal Kafle, Kevin Moran, Adwait Nadkarni, and Denys Poshyvanyk. Demo: Mutation-based Evaluation of Security-focused Static Analysis Tools for Android. In *Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering (ICSE'21), Formal Tool Demonstration Track*, May 2021.

[17] Amit Seal Ami, Kaushal Kafle, Kevin Moran, Adwait Nadkarni, and Denys Poshyvanyk. Systematic Mutation-based Evaluation of the Soundness of Security-focused Android Static Analysis Techniques. *ACM Transactions on Privacy and Security (TOPS)*, 24(15), February 2021.

[18] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. PolicyLint: Investigating Internal Privacy Policy Contradictions on Google Play. In *Proceedings of the USENIX Security Symposium*, 2019.

[19] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. Actions Speak Louder than Words: Entity-Sensitive Privacy Policy and Data Flow Analysis with PoliCheck. In *Proceedings of the USENIX Security Symposium*, 2020.

[20] Anonymous. Online appendix for "*Belt and suspenders*" or "*just red tape*"?: Investigating Early Outcomes and Perceptions of IoT Security Compliance Enforcement. https://sites.google.com/view/iotcompliance/home, 2023.

[21] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[22] Richard Bonett, Kaushal Kafle, Kevin Moran, Adwait Nadkarni, and Denys Poshyvanyk. Discovering Flaws in Security-Focused Static Analysis Tools for Android Using Systematic Mutation. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security'18)*, pages 1263–1280, 2018.

[23] V. Braun and V. Clarke. *Thematic Analysis: A Practical Guide*. SAGE Publications, 2021.

[24] California Legislature. Sb-327 information privacy: connected devices. https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201720180SB327, 2020.

[25] Efstratios Chatzoglou, Georgios Kambourakis, and Christos Smiliotopoulos. Let the cat out of the bag: Popular android iot apps under security scrutiny. *Sensors*, 22(2):513, 2022.

[26] Anca E Cretu and Roderick J Brodie. The influence of brand image and company reputation where manufacturers market to small firms: A customer value perspective. *Industrial marketing management*, 36(2):230–240, 2007.

[27] Pardis Emami-Naeini, Yuvraj Agarwal, Lorrie Faith Cranor, and Hanan Hibshi. Ask the Experts: What Should Be on an IoT Privacy and Security Label? In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 447–464, May 2020.

[28] Pardis Emami-Naeini, Janarth Dheenadhayalan, Yuvraj Agarwal, and Lorrie Faith Cranor. Are consumers willing to pay for security and privacy of iot devices?

[29] Pardis Emami-Naeini, Janarth Dheenadhayalan, Yuvraj Agarwal, and Lorrie Faith Cranor. An Informative Security and Privacy "Nutrition" Label for Internet of Things Devices. *IEEE Security & Privacy*, 20(2):31–39, March 2022.

[30] The White House. The President's Executive Order (EO) 14028 on Improving the Nation's Cybersecurity. https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/, May 2021.

[31] International Standards Organization. ISO/IEC 17025 TESTING AND CALIBRATION LABORATORIES. https://www.iso.org/ISO-IEC-17025-testing-and-calibration-laboratories.html, 2021.

[32] ioXt Alliance Members. ioxt: The global standard for iot security. https://www.ioxtalliance.org/, 2021.

[33] J. Developers,. jadx - Dex to Java decompiler. https://github.com/skylot/jadx/, 2022.

[34] Xin Jin, Sunil Manandhar, Kaushal Kafle, Zhiqiang Lin, and Adwait Nadkarni. Understanding iot security from a market-scale perspective. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1615–1629, 2022.

[35] Shane D. Johnson, John M. Blythe, Matthew Manning, and Gabriel T. W. Wong. The impact of IoT security labelling on consumer product choice and willingness to pay. *PLOS ONE*, 15(1):e0227800, January 2020.

[36] Davino Mauro Junior, Luis Melo, Hao Lu, Marcelo d'Amorim, and Atul Prakash. A study of vulnerability analysis of popular smart devices through their companion apps. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 181–186. IEEE, 2019.

[37] Kaushal Kafle, Kevin Moran, Sunil Manandhar, Adwait Nadkarni, and Denys Poshyvanyk. A Study of Data Store-based Home Automation. In *Proceedings of the 9th ACM Conference on Data and Application Security and Privacy (CODASPY)*, March 2019.

[38] Kaushal Kafle, Kevin Moran, Sunil Manandhar, Adwait Nadkarni, and Denys Poshyvanyk. Security in Centralized Data Store-based Home Automation Platforms: A Systematic Analysis of Nest and Hue. *ACM Transactions on Cyber-Physical Systems (TCPS)*, 5(1), December 2020.

[39] Patrick Gage Kelley, Lorrie Faith Cranor, and Norman Sadeh. Privacy as part of the app decision-making process. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 3393–3402, 2013.

[40] Stefan Krüger, Sarah Nadi, Michael Reif, Karim Ali, Mira Mezini, Eric Bodden, Florian Göpfert, Felix Günther, Christian Weinert, Daniel Demmler, and Ram Kamath. CogniCrypt: Supporting Developers in Using Cryptography. In *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, ASE 2017, pages 931–936, Piscataway, NJ, USA, 2017. IEEE Press.

[41] California State Legislature. California Privacy Rights Act of 2020 ("CPRA"). https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5, 2020.

[42] Christopher Lentzsch, Sheel Jayesh Shah, Benjamin Andow, Martin Degeling, Anupam Das, and William Enck. Hey alexa, is this skill safe?: Taking a closer look at the alexa skill ecosystem. *Network and Distributed Systems Security (NDSS) Symposium2021*, 2021.

[43] Josh Lerner and Jean Tirole. A model of forum shopping. *American economic review*, 96(4):1091–1113, 2006.

[44] Tianshi Li, Kayla Reiman, Yuvraj Agarwal, Lorrie Faith Cranor, and Jason I. Hong. Understanding Challenges for Developers to Create Accurate Privacy Nutrition Labels. In *CHI Conference on Human Factors in Computing Systems*, pages 1–24, New Orleans LA USA, April 2022. ACM.

[45] Steven B. Lipner. The birth and death of the orange book. *IEEE Annals of the History of Computing*, 37(2):19–31, 2015.

[46] Samin Yaseer Mahmud, Akhil Acharya, Benjamin Andow, William Enck, and Bradley Reaves. Cardpliance: Pci dss compliance of android applications. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 1517–1533, 2020.

[47] Matt Warman MP. New cyber security laws to protect smart devices amid pandemic sales surge. https://tinyurl.com/ae28rcp, 2021.

[48] Oliver Michler, Reinhold Decker, and Christian Stummer. To trust or not to trust smart consumer products: A literature review of trust-building factors. *Management Review Quarterly*, 70(3):391–420, August 2020.

[49] MobSF. Mobile Security Framework (MobSF). https://github.com/MobSF/Mobile-Security-Framework-MobSF, 2022.

[50] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. {WHYPER}: Towards automating risk assessment of mobile applications. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 527–542, 2013.

[51] European Parliament and Council of the European Union. General Data Protection Regulation (EU) 2016/679 ("GDPR"), 2016.

[52] Prolific. Approvals, rejections and returns; Prolific. https://researcher-help.prolific.co/hc/en-gb/articles/360009092394-Approvals-rejections-returns, 2023.

[53] Sazzadur Rahaman, Ya Xiao, Sharmin Afrose, Fahad Shaon, Ke Tian, Miles Frantz, Murat Kantarcioglu, and Danfeng (Daphne) Yao. CryptoGuard: High Precision Detection of Cryptographic Vulnerabilities in Massive-sized Java Projects. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security - CCS '19*, pages 2455–2472, London, United Kingdom, 2019. ACM Press.

[54] Rock Stevens, Josiah Dykstra, Wendy Knox Everette, James Chapman, Garrett Bladow, Alexander Farmer, Kevin Halliday, and Michelle L Mazurek. Compliance Cautions: Investigating Security Issues Associated with US Digital-Security Standards. In *In the Proceedings of the Network and Distributed Systems Symposium (NDSS)*, 2020.

[55] Rock Stevens, Faris Bugra Kokulu, Adam Doupé, and Michelle L Mazurek. Above and beyond: Organizational efforts to complement us digital security compliance mandates. In *In the Proceedings of the Network and Distributed Systems Symposium (NDSS)*, 2022.

[56] The White House. Executive order on improving the nation's cybersecurity. https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/, May 2021.

[57] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, Xianzheng Guo, and Patrick Tague. {SmartAuth}:{User-Centered} authorization for the internet of things. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 361–378, 2017.

[58] United States Congress. Internet of things cybersecurity improvement act of 2020. https://www.congress.gov/116/plaws/publ207/PLAW-116publ207.pdf, 2020.

[59] United States Senate. S.965 - cyber shield act of 2021. https://www.congress.gov/bill/117th-congress/senate-bill/965, 2021.

[60] U.S. Department of Defense. Department of Defense Trusted Computer System Evaluation Criteria. https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/dod85.pdf, 1985. Accessed on: June 2021.

[61] Xueqiang Wang, Yuqiong Sun, Susanta Nanda, and XiaoFeng Wang. Looking from the mirror: Evaluating iot device security through mobile companion apps. In *Proceedings of the 28th USENIX Security Symposium (USENIX)*, pages 1151–1167, 2019.

Table 3: An overview of certified mobile-IoT apps that are analyzed in this paper.

No.	App	Certified version	Latest version
1.	Tuya	3.28.5	4.4.2
2.	NetHome Plus	V4.3.710	V5.5.0922
3.	Eureka	2.4.0	3.2.2
4.	Midea Air	V5.1.0.625	V5.5.0922
5.	MSmartHome	2.16.1	2.28.1
6.	TSmartLife	1.11.1	1.15.0
7.	GreenMAX DRC	v1.62	01.75.0025
8.	Wyze	2.25.22	2.34.75
9.	Dals Connect	1.2.1	1.3.0
10.	Google Home	2.40.1.10	2.59.33.2
11.	Hubspace	1.1.25	1.7.47

A Appendix - App Analysis

A.1 Preliminary search for mobile-IoT apps

We first analyzed the descriptions/Play Store listings and developer websites of the top 100 mobile-IoT apps from a dataset recently developed by Jin *et al.* [34], to identify any mentions of any of the CLEFs we found, or general mentions of certification, allocating 10 minutes per application. This analysis did not identify any instances of certification mentioned in user-facing app metadata. We also manually examined the websites of CLEFs to search for lists of apps they may have certified. Upon the conclusion of this search, we found two published lists of certified apps: (1) the list of certified mobile-IoT apps published by IOXT [32], a popular CLEF as well as an alliance that administers the IOXT standard, and (2) the list of general certified Android apps from Google's MASA [8]. As most MASA apps were generic Android apps (i.e., not mobile-IoT), with the exception of the NEST app, we decided to focus our investigation of **RQ$_1$** on the list of the mobile-IoT apps certified by IOXT.

A.2 Sampling of non-certified apps

We sampled the set of non-certified apps using both Google Play's search engine as well as a dataset of 37k mobile-IoT apps developed by Jin et al. [34]. To elaborate, to identify a corresponding non-certified app for each certified app, we searched Google Play for the certified app, and short-listed apps that were classified by Google Play as "similar" to the certified app, i.e., of comparable functionality. Further, we eliminated apps that were not a part of the 37k mobile-IoT dataset (as they were not mobile-IoT). As several non-certified apps can be similar to a certified app, we selected the first app with the highest install count. The set of non-certified mobile-IoT apps identified using this approach is shown in Table 4 in Appendix A.

Table 4: An overview of non-certified mobile-IoT apps and their vulnerabilities.

App	Version	Vulnerabilities found
SmartThings	1.7.94.21	Using AES with CBC for encryption Using only AES for encryption
Amazon Alexa	2.2.491118.0	Trusting all certificates Using SSL as context
LG Thinq	4.1.32020	Using AES with CBC for encryption Using AES with ECB for encryption Trusting all certificates Allowing all hostnames Using TLSv1 as context No clearPassword call after using PBEKeySpec
iRobot Home	7.2.0	Using AES with CBC for encryption Using AES with ECB for encryption
Blink Home Monitor	6.21.0	None
Geeni	2.2.3	Using AES with CBC for encryption Using MD5 hashing
Yi Home	6.0.3	Using SSL as context Allowing all hostnames Trusting all certificates Using AES with CBC for encryption Using AES with ECB for encryption Using MD5 hashing
HappyLighting	1.6.21	None
mydlink	2.9.0	None
Gosund	5.1.83	Using AES with ECB for encryption
Amazon Key	2.0.3144.1	Using AES with CBC for encryption

Table 5: Fraction of vulnerable apps per misuse.

No.	Vulnerabilities	certified apps	non-certified apps
1.	Using only AES for encryption	1/11 (0.09)	1/11 (0.09)
2.	AES with ECB for encryption	4/11 (0.36)	4/11 (0.36)
3.	AES with CBC for encryption	6/11 (0.55)	6/11 (0.55)
4 .	No clearpassword() call after using PBEKeySpec	1/11 (0.09)	1/11 (0.09)
5.	Using MD5 hashing	7/11 (0.64)	2/11 (0.18)
6.	Using SHA1 hashing	6/11 (0.55)	0
7.	Trusting all certificates	3/11 (0.27)	3/11 (0.27)
8.	Allowing all hostnames	3/11 (0.27)	2/11 (0.18)
9.	Using SSL as context	2/11 (0.18)	2/11 (0.18)
10.	Using TLSv1 as context	2/11 (0.18)	1/11 (0.09)
	Total Vulnerabilities	**35**	**22**

A.3 Vulnerability Descriptions

B Appendix - Compliance Analysis

C Appendix - User Perceptions and Expectations

We piloted our survey with 8 graduate students and updated it based on their feedback.

Table 6: An overview of the permission analysis results.

No.	App	Requested permissions that the app does not explicitly inform users
1	TUYA	ACCESS BACKGROUND LOCATION
2	NETHOME PLUS	ACCESS BACKGROUND LOCATION
3	EUREKA	READ EXTERNAL STORAGE, WRITE EXTERNAL STORAGE
4	MIDEA AIR	READ EXTERNAL STORAGE, WRITE EXTERNAL STORAGE
5	MSMARTHOME	CALL PHONE, READ CONTACTS, GET ACCOUNTS
6	TSMARTLIFE	CALL PHONE, GET ACCOUNTS
7	GREENMAX DRC	READ EXTERNAL STORAGE, WRITE EXTERNAL STORAGE
8	WYZE	READ SMS, READ CALL LOGS, ANSWER PHONE CALLS, RECEIVE SMS, READ CONTACTS, ACCESS BACKGROUND LOCATION
9	DALS CONNECT	READ EXTERNAL STORAGE, WRITE EXTERNAL STORAGE
10	GOOGLE HOME	ACCESS BACKGROUND LOCATION, GET ACCOUNTS
11	HUBSPACE	CAMERA

s

Table 7: A short description of each type of misuse cases considered in the vulnerability analysis of mobile-IoT apps.

No.	Misuse case	Description
Cryptographic encryption misuse		
1.	Using only AES for encryption	Using AES for encryption defaults to ECB mode in Java, which is vulnerable to dictionary attack on encrypted messages
2.	Using AES with ECB for encryption	Using AES with ECB mode is vulnerable to dictionary attack on encrypted messages
3.	Using AES with CBC for encryption	Using AES with CBC for encryption is susceptible to padding oracle attacks
4 .	No clearpassword() call after PBEKeySpec	Without calling clearPassword, PBEKeySpec keeps an internal copy of the password in the memory, which is unsafe
Cryptographic hash misuse		
5.	Using MD5 hashing	MD5 is considered a weak hashing algorithm that is susceptible to rainbow table attack
6.	Using SHA1 hashing	SHA-1 is considered weak hashing algorithm that is susceptible to rainbow table attack
Client-server secrecy misuse		
7.	Trusting all certificates	Without checking the chain of trust, any certificate is accepted, making man in the middle attack possible
8.	Allowing all hostnames	Without checking hostname, any signed certificate is accepted, making man in the middle attack possible
9.	Using SSL as context	Any version other than TLS <1.2 is less secure and can be used by a malicious actor for a downgrade attack
10.	Using TLSv1 as context	Any version other than TLS <1.2 is less secure and can be used by a malicious actor for a downgrade attack

Table 8: Participants' familiarity with compliance standards

No.	Choice	Answer
Familiarity with Software/General Security Compliance Standards		
1.	Open Web Application Security Project (OWASP) Top 10	6
2.	NIST Cryptographic Standards and Guidelines	9
3.	Mobile Application Security Assessment (MASA) by Google	22
4 .	Mobile Application Security Verification Standard (MASVS) by OWASP	5
5 .	None	140
Familiarity with IoT Specific Security Compliance Standards		
6.	ioXt IoT Security Standard	11
7.	GSMA IoT Security Guidelines and Assessment	6
8.	IoTSF IoT Security Assurance Framework	3
9.	IoTAA Internet of Things Security Guideline	5
10.	JPCERT - IoT Security Checklist	3
11.	None	154

Table 9: Mobile-IoT apps that participants have used or are using (from our given list)

No.	App	Number of participants
1.	Google Home	86
2.	Wyze	25
3.	MSmartHome	6
4.	Eureka	5
5.	Tuya	4
6.	NetHome Plus	2
7.	TSmartLife	2
8.	Midea Air	1

Table 10: Participants' interest in mobile-IoT apps that they consider to be better at protecting their data and devices

Choice	Number of participants
Certified mobile-IoT apps	134
Non-certified mobile-IoT apps	1
Both of them	9
Neither of them	29
Total	**173**

Table 11: The severity level of two scenarios based on participants' rating.

Severity Level	S1: information leakage	S2: stealing authentication
Critical	87	128
High	62	35
Medium	20	9
Low	2	1
Not severe at all	2	0
Total	**173**	**173**

Table 12: The likelihood of two scenarios based on participants' rating.

Likelihood	S1: information leakage	S2: stealing authentication
Extremely likely	43	38
Somewhat likely	86	79
Neither likely nor unlikely	23	29
Somewhat unlikely	19	24
Extremely unlikely	2	3
Total	**173**	**173**