TU WIEN Informatics

“Things” Fall Apart

Adventures in Securing a World of Connected Chaos

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktorin der Technischen Wissenschaften

by

Carlotta Tagliaro, MSc
Registration Number 12130515

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dr.in techn. Martina Lindorfer
Second advisor: Prof. Dr. Kevin Borgolte

The dissertation has been reviewed by:

Stefano Zanero	Nataliia Bielova

Vienna, January 16, 2026

Carlotta Tagliaro

Erklärung zur Verfassung der Arbeit

Carlotta Tagliaro, MSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 16. Jänner 2026

Carlotta Tagliaro

Acknowledgements

First, and foremost I would like to express my deepest gratitude to my supervisor, Martina Lindorfer, for her unwavering support, guidance, and encouragement throughout my research journey. Her expertise and insights have been invaluable in shaping this work. Her supporting words have also helped me when facing rejections, and doubting my abilities. I would also like to thank my co-advisor, Kevin Borgolte, for his constructive feedback and support (and his LaTeX nitpicks which sometimes have caused me nightmares). I would also like to thank all my collaborators that helped me in this journey and supported with their invaluable feedback. I would also like to gratefully acknowledge the support of netidee (Internet Stiftung Austria), whose funding made this research possible. This work was supported by a netidee scholarship, which significantly contributed to the development and completion of this research.

I would also like to thank all my colleagues in the SecPriv group for supporting me and making the office a fun place to be. I will never forget our board games nights, the hikes and the tennis matches. Someone please appoint the next social activities coordinator, I cannot leave the groups die! A special thanks goes to the Seclab group, I am happy to be able to call you not only colleagues but also friends.

A special thanks goes to all my friends outside of the university, who have always been there for me and supported me in every possible way. I am grateful for your friendship and the joy you bring to my life. I would like to thank Jessica, with whom I shared the ups and downs of being a PhD student. And here's to all my undergrad friends, Hubi, Costa, Zanella, Palla and Balle, with whom I spent the last 10 years of adventures. Finally, I also want to thank my friends from Vienna, Claudia, Jil, Riccardo, Sophie and Mel, the MyClubsGang, the Jitsu friends, and all the others, who made my time in Vienna unforgettable. A huge thanks also goes to my boyfriend, Max, for his endless support and patience during these years.

Infine, voglio esprimere la mia più profonda gratitudine a mia mamma, Lorenza, per il suo amore, il suo sostegno e il suo incoraggiamento lungo tutto il mio percorso accademico. Ti sarò per sempre grata per tutto ciò che hai fatto per me. Un ringraziamento va anche al mio cane, Martin, per il suo sostegno inconsapevole.

Kurzfassung

Diese Dissertation befasst sich mit den weitreichenden Sicherheitsherausforderungen im Ökosystem des Internet der Dinge (IoT) durch eine interdisziplinäre Methodik, die Systemanalysen, groß angelegte Internetmessungen und nutzerzentrierte Studien verbindet. Zunächst stellen wir IoTFlow vor, eine statische Analysetechnik, die fast 10,000 Android-Begleit-Apps für IoT-Geräte untersucht, um Kommunikationsendpunkte zu rekonstruieren und unsichere Praktiken wie festkodierte Zugangsdaten und offen zugängliche Testschnittstellen aufzudecken. Anschließend führen wir, aufbauend auf den Erkenntnissen von IoTFlow, eine großflächige Untersuchung von IoT-Backends über die Protokolle MQTT, CoAP und XMPP durch und decken dabei weit verbreitete Schwachstellen wie Datenlecks, schwache Authentifizierungsverfahren und Denial-of-Service-Risiken auf. Daraufhin initiieren wir gemeinsam mit dem Dutch National Cyber Security Centre ein koordiniertes Verfahren zur Schwachstellenoffenlegung, informieren Tausende Betreiber und verfolgen die Behebung der Probleme. Diese Bemühungen decken kritische Lücken in der aktuellen IoT-Sicherheitsaufsicht auf.

Über die technische Infrastruktur hinaus untersuchen wir Datenschutzrisiken des Hybrid Broadcast Broadband TV (HbbTV) in fünf europäischen Ländern und zeigen dabei weit verbreitetes Tracking sowie Nichteinhaltung von Datenschutzrichtlinien bei Smart-TV-Sendern. Abschließend schlagen wir eine Brücke zwischen technischen und menschlichen Perspektiven durch eine zweistufige Nutzerstudie zur Risikowahrnehmung und zu Sicherheitsverhalten im IoT: einer expertengestützten Kategorisierung von IoT-Geräten folgt eine groß angelegte Umfrage mit 213 Teilnehmenden, die erhebliche Diskrepanzen zwischen wahrgenommenem Risiko und tatsächlichen Sicherheitspraktiken in einzelnen Gerätekategorien offenlegt. Unsere Ergebnisse unterstreichen den dringenden Bedarf an Sicherheitslösungen. Durch die Kombination von Systemsicherheit, Internetmessung und nutzerorientierten Ansätzen erweitert diese Arbeit nicht nur das technische Verständnis von IoT-Schwachstellen, sondern liefert auch Empfehlungen für regulatorische Maßnahmen.

Abstract

This thesis addresses the pervasive security challenges in the Internet of Things (IoT) ecosystem through an interdisciplinary methodology that combines system-level analysis, large-scale Internet measurement, and user-centered studies. First, we present IoTFlow, a static analysis technique that examines nearly 10,000 Android IoT companion apps to reconstruct device communication endpoints and reveal insecure practices such as hardcoded credentials and exposed test interfaces. Next, building on IoTFlow's findings, we conduct a large-scale assessment of IoT backends across various communication protocols (MQTT, CoAP, and XMPP), uncovering widespread vulnerabilities that include data leaks, weak authentication schemes, and denial-of-service threats. In response, we initiate a coordinated vulnerability disclosure procedure with the Dutch National Cyber Security Centre, notifying thousands of operators and tracking remediation outcomes. This effort exposes critical gaps in current IoT security oversight.

Beyond technical infrastructure, we examine privacy risks in Hybrid Broadcast Broadband TV (HbbTV) across five European countries, revealing pervasive tracking and policy non-compliance in smart TV channels. Finally, we bridge the technical and human perspectives through a two-part user study on IoT risk perceptions and security behaviors. We perform an expert-driven categorization of IoT devices, followed by a large-scale user survey of 213 participants, which exposes significant mismatches between perceived risk and actual security practices across device categories. Our findings underscore the pressing need for effective security solutions. By combining systems security, Internet measurement, and user-centered approaches, our work not only extends technical understanding of IoT vulnerabilities but also suggests regulatory policy.

List of Publications

This dissertation is based on several peer-reviewed papers, previously published in the following conference and workshop proceedings; the chapters mirror these publications.

[1] David Schmidt, **Carlotta Tagliaro**, Kevin Borgolte, and Martina Lindorfer. *IoTFlow: Inferring IoT Device Behavior at Scale through Static Mobile Companion App Analysis*. In **Proceedings of the 30th ACM SIGSAC Conference on Computer and Communications Security (CCS)**, 2023. https://doi.org/10.1145/3576915.3623211 (Chapter 2)

[2] **Carlotta Tagliaro**, Martina Komsic, Andrea Continella, Kevin Borgolte, and Martina Lindorfer. *Large-Scale Security Analysis of Real-World Backend Deployments Speaking IoT-Focused Protocols*. In **Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)**, 2024. https://doi.org/10.1145/3678890.3678899 (Chapter 3)

[3] Ting-Han Chen, **Carlotta Tagliaro**, Martina Lindorfer, Kevin Borgolte, and Jeroen Van Der Ham-De Vos. *Are You Sure You Want to Do Coordinated Vulnerability Disclosure?* In **Proceedings of the 2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)**, 2024. https://doi.org/10.1109/EuroSPW61312.2024.00039 (Chapter 4)

[4] **Carlotta Tagliaro**, Florian Hahn, Riccardo Sepe, Alessio Aceti, and Martina Lindorfer. *I Still Know What You Watched Last Sunday: Privacy of the HbbTV Protocol in the European Smart TV Landscape*. In **Proceedings of the 2023 Network and Distributed System Security Symposium (NDSS)**, 2023. https://doi.org/10.14722/ndss.2023.24102 (Chapter 5)

[5] **Carlotta Tagliaro**, Rafael Vrečar, Alexander Schmidt, Evropi Stefanidi, Martina Lindorfer, and Paweł W. Woźniak. *Mind over Matter: How Device Functionality Affects Smart Home Security Behaviors.* Under submission. (Chapter 6)

Further, the following publications were authored during the PhD, but are not part of this dissertation:

[1] **Carlotta Tagliaro**, Florian Hahn, Riccardo Sepe, Alessio Aceti, and Martina Lindorfer. *Investigating HbbTV Privacy Invasiveness Across European Countries.* In **Proceedings of the LASER Workshop on Learning from Authoritative Security Experiment Results**, 2023. https://www.ndss-symposium.org/wp-content/uploads/laser2023-102-paper.pdf

[2] **Carlotta Tagliaro**, Andrej Danis, Kevin Borgolte, and Martina Lindorfer. *New Platform, Old Issues: How Web-based TV Broadcasts threaten Users' Security.* Under submission.

[3] **Carlotta Tagliaro**, Martina Komsic, Gianluca Anselmi, Anna Maria Mandalari, and Martina Lindorfer. *Bridging Devices and Apps: A Joint Analysis of IoT Privacy and Communication.* Under submission.

Contents

CHAPTER 1

Introduction

The **number of Internet of Things (IoT) connected devices peaked at 19 billion in 2025**, while forecasts claim that by 2030, there will be more than 29 billion, highlighting the considerable impact of such devices on our world [298]. Their presence extends across various domains: they can assist people with eHealth sensors, enhance living conditions with smart home systems (e.g., Smart TVs and connected thermostats), and enable smart city solutions (e.g., air quality monitors and traffic management). At the same time, industrial IoT deployments are reshaping manufacturing, logistics, and energy infrastructures, embedding IoT into critical national assets. Yet, the same qualities that make IoT devices attractive, pervasiveness, heterogeneity, and connectivity, expose them to security risks. The seemingly harmless devices are vectors for impactful attacks, ranging from exfiltration of sensitive personal or corporate data, to Distributed denial-of-service (DDoS) attacks, and to manipulation and collection of data with real-world consequences for safety and privacy. **Attack scenarios are far from hypothetical and unprecedented in their scale due to the ubiquity of IoT devices.** The (in)famous Mirai botnet is just one example [24]: 600k infected devices launched a major DDoS in 2016, causing severe disruption of Internet services peaking at an unprecedented 1Tbps rate. In another example, the U.S. Food and Drug Administration (FDA) confirmed that St. Jude Medical's implantable cardiac devices are vulnerable, potentially causing battery depletion or the administration of incorrect shocks [160]. Subsequent research has shown that **IoT threats are not limited to legacy devices; new vulnerabilities keep emerging as devices integrate with cloud services, mobile apps, and third-party libraries.** These risks highlight the evolving attack surface, which expands not only in breadth, targeting billions of devices, but also in depth, through the increasing integration of IoT into critical infrastructures and personal spaces.

Despite the urgency of these challenges, **there are currently only a few standardized attempts or best practices for securing the IoT ecosystem, such as Manufacturer Usage Descriptions (MUDs) [83, 91, 162].** Nevertheless, they are only

high-level descriptions and not yet deployed at scale [195]. Other initiatives, such as baseline security recommendations from governments and standards bodies, often remain voluntary or are challenging to enforce in practice due to the diversity of vendors and protocols. Even widely adopted cloud platforms suffer from insecure defaults and flawed documentation. As Jia et al. showed in their study of Amazon Web Services (AWS), the leading IoT cloud platform, their official developer guide provided 38 best practices, 26 of which introduced vulnerabilities [148]. These findings illustrate how insecurity can be embedded in documentation and development practices, propagating into millions of applications. Industry surveys further highlight this paradox. According to a 2020 study of more than 1,600 IoT developers, security is the biggest concern [85]. Despite this, Sombatruang et al. show that **neither Internet Service Providers (ISPs) nor individual consumers are willing to act and take responsibility, due to a lack of technical knowledge or resources [281].** Furthermore, the high number of stakeholders in the IoT ecosystem, including manufacturers, platform providers, ISPs, regulators, and end users, creates a shared sense of responsibility that worsens complexity.

Unlike desktops and mobile phones, which researchers have extensively analyzed for security and privacy issues, **the analysis of smart devices remains an open challenge.** So far, researchers have mainly focused on studying a handful of selected devices [63, 127, 179, 221, 253, 310, 324] or their firmware [58]. While these studies provided valuable insights, they fail to capture the broader landscape. The vast number of available communication protocols and the heterogeneity of IoT architectures, ranging from constrained sensors using CoAP to publish/subscribe models with MQTT and real-time presence protocols like XMPP, complicate analysis and hinder reproducibility. **This heterogeneity, together with the absence of systematic methodologies and scalable datasets, leaves the ecosystem under-scrutinized and exposes end users to risk.**

Devices do not always directly communicate with (cloud) endpoints; for example, companion apps can mediate communication. Mobile phones play a crucial role in controlling IoT devices. Nearly every device comes with a companion app, freely available for Android on the Google Play Store and iOS on the Apple App Store. These apps allow users to control their devices in their local home network or remotely over the Internet. Consequently, companion apps process and store both sensitive information collected from devices and information about the cloud backend, opening up new attack surfaces.

Our first contribution is IoTFlow, a static analysis approach that reconstructs IoT communication parameters from Android companion apps. These apps act as the primary interface between users, devices, and backends [247, 253]. IoTFlow treats these apps as proxies for device behavior, enabling analysis at scale without needing access to thousands of physical devices.

IoTFlow builds on Value Set Analysis (VSA) to extract values that flow into network sinks, such as broker addresses, ports, topics, and authentication material. To specialize the analysis for IoT, we extended VSA with protocol-specific signatures, focusing on the MQTT and CoAP libraries commonly embedded in apps. This design allowed IoTFlow to identify protocol endpoints, reconstruct authentication strings, and extract

topic names and message formats from application code. Applied to a dataset of 9,889 Android companion apps [149, 197, 201], **IoTFlow successfully reconstructed 147 MQTT endpoints across 176 apps and multiple CoAP endpoints and payload formats. The analysis revealed recurring insecure practices.** Several apps embedded hard-coded usernames and passwords, sometimes enabling direct connections to brokers. Others included test or development endpoints that remained accessible in production versions. By uncovering these patterns, we highlighted how poor software engineering choices at the app level directly compromised device and backend security.

We evaluated IoTFlow against dynamic traffic analysis of 13 devices. The results showed partial but meaningful overlap: IoTFlow recovered many of the same endpoints observed dynamically, while uncovering additional ones that were not contacted at runtime. This complementarity reflects the strengths of static analysis in revealing unexplored code paths. For example, IoTFlow reconstructed generic values such as topic structures or payload templates, which were harder to observe in live traffic but revealed device communication patterns. **Beyond these findings, IoTFlow demonstrated that companion apps offer a scalable platform for IoT security research.** Previous work on mobile ecosystems had focused on privacy leaks and ad trackers [246, 254], but IoTFlow extended this perspective by directly investigating IoT protocols. We thus generalized from a handful of devices to thousands of apps, representing diverse vendors and ecosystems.

Using IoTFlow [269], we reconstructed the backends that companion apps contacted and illustrated the risks through concrete case studies. In one example, a health-monitoring wearable transmitted personally identifiable information, medical indicators, and even precise location data over an outdated MQTT broker. The broker accepted unauthenticated subscriptions, enabling any third party to intercept sensitive health information and infer device software versions with known vulnerabilities. In a second case, a smart car dongle exposed real-time telemetry including geolocation traces, fuel consumption, and anti-theft status. Such information compromises user privacy and creates opportunities for targeted theft.

In both cases, weak or absent authentication allowed arbitrary clients to connect and access data that should have remained private. These findings highlight the risks that emerge when poor backend configuration crosses with sensitive applications. They also demonstrate how static analysis results translate directly into exploitable weaknesses. Motivated by these observations, our second contribution is a global measurement of IoT backends. **We broadened our scope and gathered a dataset of 337,464 endpoints across MQTT (251,382), CoAP (85,957), and XMPP (125) to perform a large-scale empirical study of IoT backends.** We collected these endpoints from Shodan queries. We performed non-invasive scans targeting three vulnerability classes: information leakage, weak authentication, and denial-of-service (DoS) attacks. **The results confirmed widespread insecurity.** Of the endpoints we successfully connected to, 9.44% (31,847 servers) leaked sensitive information, including device identifiers, telemetry, or user data. **CoAP deployments appeared particularly concerning, as**

30.18% (25,939 endpoints) suffered from amplification vulnerabilities, enabling trivial denial-of-service attacks against connected clients. For XMPP, 48.36% of endpoints used insecure authentication methods such as PLAIN or ANONYMOUS. Across protocols, transport security adoption remained low: only 0.16% of MQTT and XMPP endpoints used TLS, and nearly 70.93% relied on deprecated versions (<v1.1).

Case studies emphasized the importance of our study. A group of MQTT backends exposed telemetry from energy infrastructures, including fine-grained consumption data that adversaries could use to profile households or facilities, thereby threatening grid resilience. These examples resonated with prior observations of industrial IoT insecurity [97, 261]. Still, our study supported their findings with evidence drawn from nearly a million endpoints, demonstrating that these issues affect the ecosystem as a whole.

Throughout this study, we adhered strictly to ethical guidelines. We designed tests to minimize disruptions to services, refrained from exploitation attempts, and hosted a disclosure website at our scanning IP to inform operators. Our procedures received approval from institutional ethics boards. As security researchers, we had a responsibility not only to identify vulnerabilities but also to disclose them responsibly and contribute to improving the security of IoT backends. Research that uncovered large-scale weaknesses without follow-up would have left users exposed. **Coordinated disclosure was integral to our follow-up work, ensuring stakeholders received the information they needed to mitigate risks.**

Responsible disclosure represented both a necessity and a challenge. For our third contribution, we collaborated with the Dutch National Cyber Security Centre (NCSC) to notify operators and relay endpoint information. **We identified 15,820 vulnerable MQTT IoT backends, and we retrieved contact details for 15,046 (95.11%) via WHOIS.** To scale notifications, we grouped backends by contact and reduced the number of messages to 2,132 emails, each reporting an average of seven vulnerable servers. Between November 10 and 22, 2023, we sent all notifications using a dedicated system at the University of Twente. **The outcomes exposed the limits of disclosure.** We received 1,173 responses, but 428 (36.48%) were automatic replies from ticketing systems, and 700 (59.67%) were delivery failures. Only 32 stakeholders requested additional details, and only 28 continued active conversations to resolve their issues. Beyond these, 311 operators silently patched vulnerabilities without contacting us, while many others remained unresponsive. A follow-up assessment in January 2024 revealed that **only 52 backends (0.35%) fully adopted our recommendations, 282 (1.90%) mitigated CVEs without addressing other issues, 4,780 (32.22%) remained vulnerable, and 6,554 (44.18%) continued to expose unauthenticated resources.**

The NCSC partnership was valuable as an institutional intermediary, as it centralized reporting, relayed information to affected parties across jurisdictions, and provided a trusted communication channel. At the same time, the process underlined weaknesses in IoT control. Manufacturers, ISPs, and users often deflected responsibility [281], and without clear liability frameworks, most disclosures did not initiate remediation. **Our experience confirmed that voluntary reporting was rarely enough;** Effective in-

tervention required regulatory obligations for backend operators, standardized notification pipelines, and mechanisms for accountability.

To date, our research has identified technical vulnerabilities in companion apps and IoT backends. These contributions exposed the systemic weaknesses of the ecosystem and demonstrated how insecure design choices compromised both device functionality and backend infrastructures. However, the ultimate victims of these risks are not the systems themselves but the people who rely on them in their homes, workplaces, and everyday lives. While our technical analyses quantify systemic weaknesses, **a qualitative, user-centered approach is equally vital for understanding real-world consequences when sensitive data, such as health metrics, location traces, or viewing habits, are at stake.** Therefore, we complement our systems work with a user-focused study. Users often remain unaware of the risks created by insecure IoT deployments and lack the means to evaluate or mitigate them. We connect system-level analysis with human-centered inquiry by investigating how people perceive different devices, the risks they recognize, and how they balance privacy against functionality. This interdisciplinary approach enables us to assess not only whether IoT infrastructures are secure, but also whether users understand, accept, or resist the trade-offs imposed on them. In doing so, our research bridges technical security studies with human-computer interaction and regulatory debates, creating a more comprehensive view of IoT risks.

We pursue this broader perspective through **our fourth contribution, a study of Hybrid Broadcast Broadband TV (HbbTV), which extends our technical investigations into consumer experience and regulatory compliance.** Unlike previous work, we focused on the transmission protocol itself. Prior research has shown that HbbTV offers little to no privacy guarantees [109, 110, 111, 112]; consumers cannot easily determine whether connections are secure, what data is transferred, or how it is processed. These risks grew when broadcasters combined HbbTV insecurities with tracking and analytics, particularly in the context of *Addressable TV (ATV)*, where static broadcast advertisements give way to personalized, dynamically inserted ads [163, 274]. This shift transformed television into a powerful vector for large-scale data collection.

To measure real-world implications, we analyzed 36 TV channels across five European countries, Italy, Germany, France, Austria, and Finland, where HbbTV adoption is widespread and awareness of privacy is high [73, 95, 121]. We examined traffic flows in 2021 and 2022 using Smart TVs and protocol-level inspection. Results revealed continued broadcaster oversight: **all 36 channels contacted at least one tracking domain, and 26 engaged trackers before user consent. Seven channels failed to provide a privacy policy.** Even where tracking was partly mitigated, coverage remained incomplete, with common deny-lists blocking at most 44.29% of tracking domains in 2021 and 81.39% in 2022. Many channels also transmitted data over plain HTTP, exposing device IDs, visitor identifiers, ISP details, and even login credentials for services such as **HSE, which permitted account creation and payment without encryption.** These practices violated GDPR principles, illustrating how commercial interests often overrode privacy protection.

We complemented our traffic study with a user survey to assess consumer awareness of these risks. **Among 132 participants, 68% could not name a single security or privacy risk associated with Smart TVs or HbbTV**, and a similar proportion admitted to never reading the privacy policies of digital services. Nevertheless, when confronted with concrete scenarios of data misuse, participants expressed high levels of concern, underscoring a critical gap between awareness and exposure. This dissonance shows that users remain unprotected not only because of insecure protocols but also because they lack the means to recognize and respond to risks.

These findings extend our technical contributions into the societal dimension of IoT security. Integrating backend analysis with user-centered studies demonstrates that IoT insecurity is a systemic issue: companion apps and protocols expose sensitive data at scale, while end users neither understand nor consent to these practices. This interdisciplinary perspective underscores the importance of aligning technical safeguards with regulations and user empowerment. Building on that perspective, **for our fifth and last contribution, we conduct a two-study investigation**: first, expert card sorting to derive IoT device categories, and second, a large-scale user survey to validate them. In the expert study, ten domain experts clustered 29 representative IoT devices into meaningful categories, which we aggregated into five clusters: *Physical Access Devices*, *Surveillance & Assistance Devices*, *Media & Entertainment Devices*, *Household Convenience Devices*, and *Environment Monitoring & Control Devices*. **Experts also rated perceived security severity on a five-point scale, yielding an average score of 3.1 ($SD = 0.6$), which highlighted a clear differentiation between categories, ranging from baby monitors (4.3) to soundbars (1.6).**

Building on these results, we recruited 213 participants via Prolific to complete an online survey that measured perceptions across the five categories. Participants rated *Intention to Use* and *Perceived Risk* on 7-point Likert scales [141], and evaluated *Reliability*, *Functionality*, and *Helpfulness* on 5-point scales following McKnight et al. [185]. To capture individual traits, we also asked them to complete the McKnight Trust Disposition Scale, the CySESH Self-Efficacy Scale [43], and the SeBIS Security Behavior Scale [86]. The results revealed differences across categories. **Participants perceived *Physical Access Devices* as significantly riskier and less trustworthy than other categories, rating them lower on Reliability and Helpfulness compared to *Media & Entertainment* or *Surveillance & Assistance Devices*. Conversely, *Media & Entertainment Devices* scored lower on Perceived Risk and higher on Intention to Use, underscoring how leisure-oriented devices remained underrated in privacy threats.** Despite their popularity, Household Convenience Devices and Physical Access Devices stood out as the riskiest categories.

Participants' trait scales provided contextualization for our results. Average CySESH scores indicated confidence in basic security actions, such as turning off devices or deleting cloud data, but less certainty about complex institutional processes like contacting data protection officers. SEBIS results showed generally strong adherence to basic practices (e.g., PINs, passwords) but inconsistent behaviors such as software updates

or link checking. **These findings highlighted a gap between risk perception and secure practice**: even when participants identified higher-risk categories, they often lacked the confidence or habits to address them. This paradox mirrors broader technology adoption dilemmas: users recognize the sensitivity of health or access-related devices yet continue to prefer convenience and functionality. Conversely, despite well-documented threats, as in our study of Smart TVs [299], they underestimated the risks associated with entertainment devices. Our results suggest the need for countermeasures, including **stricter compliance regulations, protection for health and access-related device categories, and awareness campaigns that highlight the risks associated with entertainment and household devices.**

1.1 Contributions

To ensure transparent attribution, the following sections delineate my specific contributions to papers on which I am not the first author and to those that are partly derived from my master's thesis.

1.1.1 IoTFlow (Chapter 2)

As the second author of IoTFlow, I focused on extending our static analysis approach to reconstruct IoT-specific values. I integrated support for signature-based searches of the main IoT libraries, including MQTT, CoAP, and XMPP, enabling IoTFlow to reconstruct endpoints, credentials, topics, and payloads embedded in companion apps. In addition to extending the static analysis, I filtered and validated the reconstructed values. The results show that IoTFlow successfully reconstructs endpoints at scale. From 9,889 analyzed apps, we recovered 147 MQTT endpoints across 176 apps, as well as multiple CoAP endpoints and payload formats. Notably, 53.62% of MQTT endpoints accepted connections, confirming that static analysis often reveals exposed servers. We also extracted 30 usernames, 34 passwords, 726 topic names, and 330 payload formats, some of which allowed successful authentication to online brokers. Furthermore, I performed a permissions analysis on the app datasets to check whether general-purpose apps requested different permissions than companion apps. We find that companion apps utilize more privileged and potentially dangerous permissions, with two of the most frequently asked dangerous permissions being those related to the user's geographic location.

I further benchmarked IoTFlow accuracy against dynamically obtained results. To carry out this validation, I performed dynamic testing on 13 physical IoT devices, interacting with their companion apps using both automated input generation (via the Application Exerciser Monkey [21]) and systematic manual exploration of app functionality. This allowed me to compare static reconstructions with live traffic. The dynamic experiments reinforced these findings: IoTFlow not only matched endpoints observed at runtime but also uncovered additional ones not triggered during interactions.

1.1.2 Coordinated Vulnerability Disclosure (Chapter 4)

In this study, I collected and analyzed the dataset of vulnerable IoT backends, ensuring its accuracy and completeness before notification. I drafted the initial disclosure template, striking a balance between technical detail and accessibility for diverse stakeholders, including operators with varying levels of security expertise. While the Dutch NCSC and University of Twente handled the large-scale sending of emails, I was actively involved in the remediation process. Specifically, I provided technical guidance to operators who responded to our disclosure, outlining concrete steps for mitigation and explaining protocol-specific recommendations. I also communicated in follow-up emails, ensuring that disclosures translate into actionable improvements.

1.1.3 Extension of the HbbTV Study (Chapter 5)

In my paper, I significantly expand and strengthen the results initially presented in my master's thesis, both in scope and methodological rigor. In the thesis, I analyzed 9 Italian TV channels, as well as a small sample of 5 German and 2 French channels for comparison. In contrast, the paper scales this analysis to 36 channels across five countries (Italy, Germany, France, Austria, and Finland), thereby offering a broader and more representative view of the European HbbTV landscape. In terms of results, my thesis reported tracking before consent in all 9 Italian cases. Still, in the paper, I demonstrate this phenomenon across 26 of 36 channels (72%), with 20 channels (56%) explicitly using tracking pixels, thus quantifying the pervasiveness of invasive practices with greater statistical significance. The user survey also expanded from a limited Italian sample of 100 participants to 174 participants across Europe, with risk perception assessed across eight standardized scenarios, allowing for cross-comparison with prior German studies. Finally, I validate our proposed tool ("HbbTV Blocker") in the paper and compare it against regular analytics and tracking denylists, showing their limitations in the smart TV domain. Together, these enhancements, a larger dataset, longer and more rigorous measurements, more survey participants, and the validation of the HbbTV blocker, clearly mark my paper as a significant advancement over my thesis.

CHAPTER 2

IoTFlow: Inferring IoT Device Behavior at Scale through Static Mobile Companion App Analysis

The number of "smart" devices, that is, devices making up the Internet of Things (IoT), is steadily growing. They suffer from vulnerabilities just as other software and hardware. Automated analysis techniques can detect and address weaknesses before attackers can misuse them. Applying existing techniques or developing new approaches that are sufficiently general is challenging though. Contrary to other platforms, the IoT ecosystem features various software and hardware architectures.

We introduce IoTFlow, a new static analysis approach for IoT devices that leverages their mobile companion apps to address the diversity and scalability challenges. IoTFlow combines Value Set Analysis (VSA) with more general data-flow analysis to automatically reconstruct and derive how companion apps communicate with IoT devices and remote cloud-based backends, what data they receive or send, and with whom they share it. To foster future work and reproducibility, our IoTFlow implementation is open source.

We analyze 9,889 manually verified companion apps with IoTFlow to understand and characterize the current state of security and privacy in the IoT ecosystem, demonstrating the utility of IoTFlow. We compare how these IoT apps differ from 947 popular general-purpose apps in their local network communication, protocols, and who they communicate with. Moreover, we investigate how the results of IoTFlow compare to dynamic analysis, with manual and automated interaction, of 13 IoT devices when paired and used with their companion apps. Overall, utilizing IoTFlow, we discover various IoT security and privacy issues, such as abandoned domains, hard-coded credentials, expired certificates, and sensitive personal information being shared.

2.1 Introduction

The number of Internet of Things (IoT) devices, that is, smart devices, is rising rapidly: Forecasts expect the number of IoT devices to grow to 29 billion in 2030 [130]. These devices collect data about their users and environment to make smart decisions. For example, to call for help in an emergency, a smartwatch may collect health indicators. This means that users need to trust them to handle their data with care. Unfortunately, smart devices have gained notoriety for their security and privacy issues, leading to the catchphrase *"the S in IoT stands for security."* Notably, employees of Ring had unauthorized access to users' security camera footages uploaded to their cloud backend [159]. Similarly, the European Union (EU) recalled kids' smartwatches because they exposed sensitive information and could be easily compromised by attackers [42].

Prior work extensively analyzed open and closed source desktop and mobile applications (apps) for security and privacy issues, but analyzing smart devices remains an open challenge. Related work in this domain mainly focused on firmware vulnerabilities [58, 70] or on analyzing a handful of selected devices [63, 127, 179, 253, 310, 324]. This does, however, not scale to the wide variety of smart devices with diverse software and hardware architectures. Intuitively, buying thousands of devices to analyze them in a lab setting is financially and practically infeasible.

Therefore, to enable the large-scale discovery and analysis of security and privacy issues in the IoT ecosystem, we propose IoTFlow, a novel static analysis approach for IoT devices via their mobile companion apps. These apps are important in directly controlling IoT devices and can serve as intermediaries to cloud backends. Practically all IoT devices have such apps available for Android and iOS [58, 197, 201, 247]. They allow users to setup and control their devices locally, via the local network or Bluetooth, or remotely, via the Internet. For some devices, their apps are the only gateway to the Internet. Overall, the apps store and process information collected by the IoT devices and about the remote infrastructure. The nature of the data the devices collect and use may also be highly sensitive. Further, attackers could misuse apps with hard-coded information (e.g., endpoints, credentials) to eavesdrop on others' private information, or distribute malicious content via misconfigured IoT backends or abandoned domains. Using a misconfigured backend, they could exploit vulnerabilities to create a new botnet of hundreds of thousands of devices, even if they are not directly reachable on the Internet.

The basic idea of evaluating the security and privacy of IoT devices indirectly by studying their companion apps has been explored by prior work. For example, Wang et al. [315] leveraged it to identify rebranded devices by searching for similar apps. They find vulnerabilities in other devices because of "private labeling" and component re-use. Chen et al. [58] and Redini et al. [247] used companion apps to inform fuzzing IoT devices, while Zuo et al. [333], Sivakumaran et al. [277, 278], and Zhao et al. [330] leveraged companion apps to identify Bluetooth (BLE) issues. Wang et al. [314] statically analyzed Samsung SmartThings apps, which are part of the SmartThings smart hub IoT ecosystem.

Existing approaches focus on re-identifying known issues shared among devices (previously

discovered through traditional techniques), still require physical devices (fuzzing), focus on a subset of companion apps (BLE), or analyze conceptually simple apps that are less widespread than general companion apps [202] (e.g., Samsung SmartThings apps, which are event flow graphs, rather than full apps; similar to "If This Then That" [140]).

In this paper, we introduce a new static analysis approach, IoTFlow, that substantially advances this basic idea. Our new approach enables us to gain new fundamental knowledge about the companion apps and corresponding smart devices at scale without actually requiring the physical device. We focus on addressing two crucial limitations of state-of-the-art techniques: First, we discover new issues automatically instead of re-identifying existing issues, which would require a priori knowledge that they exist. Second, we investigate individual devices instead of assuming that groups of devices share or re-use components. Specifically, our approach enables us to infer and gain new insights into the security and privacy of companion apps and their corresponding smart devices by reconstructing information about the used network protocols, endpoints, and the data they receive. With our approach, we can answer the following important but open questions concerning security and privacy in the IoT ecosystem:

RQ1: *How do companion apps and devices communicate?*

RQ2: *Who are companion apps communicating with?*

RQ3: *Which data are companion apps sharing (and how)?*

Specifically, our approach (1) identifies communication trigger points, (2) uses Value Set Analysis (VSA) to reconstruct network-related information on where data is coming from or transferred to, such as the URLs that are being contacted, (3) utilizes Data-flow Analysis (DFA) to determine what data is being accessed, shared, and with whom, and (4) assesses the corresponding impact.

We evaluate our approach on 9,889 unique and manually verified companion apps [149, 197, 201] to show that we can analyze IoT devices accurately and at scale. Additionally, we study the differences in network behavior between the companion apps and 947 popular general-purpose apps. Finally, we verify IoTFlow's accuracy and compare it with dynamic analysis, interacting with 13 IoT devices via their companion apps.

In this paper, we make the following contributions:

- We introduce IoTFlow, a new static program analysis approach utilizing VSA and DFA to analyze the behavior of IoT devices based on their companion apps' interactions with them and their remote backend.
- We show that IoTFlow can accurately infer the network behavior of companion apps at scale by analyzing 9,889 IoT apps.
- We analyze how and with whom companion apps communicate, what data they share locally with devices and remotely, and we highlight their differences to general apps.

- Using IoTFlow, we automatically discover rampant security and privacy issues in the IoT ecosystem, such as abandoned control domains, hard-coded credentials, expired certificates, or shared Personally Identifiable Information (PII).

Artifacts. To foster reproducibility and future research, we make our implementation and analysis artifacts available at https://github.com/SecPriv/iotflow.

2.2 Motivation

Following, we motivate IoTFlow with the need for at-scale IoT device behavior analysis, the interdependence of companion apps and IoT infrastructure, and the unique features of companion apps compared to general-purpose apps.

Large-scale IoT Device Behavior Analysis. The plethora of security and privacy issues that supposedly plague smart devices are a well-hypothesized problem in the security community and often anecdotally confirmed when yet another real-world issue is found and the press is reporting on it. Unfortunately, we currently lack techniques to discover such issues and also other vulnerabilities in smart devices automatically and at scale. State-of-the-art approaches focus on analyzing the devices' firmware [70], requiring tedious and substantial manual effort to tailor it to each individual device, possibly even each hardware revision of a device. It also suffers from the many challenges of analyzing firmware, such as having to deduce and infer what sensors and actuators exist, model them, and understand how the firmware is communicating with it. Even if it would be feasible to scale such approaches to the many devices, it is also challenging to automatically gather thousands of firmware images, as devices use different processes to retrieve and update their firmware. At the same time, more and more IoT devices are being manufactured and used. Thus, it remains an open problem how to analyze the increasing number of diverse devices.

For large open source projects, the average lifetime of vulnerabilities is multiple years [6, 166]. Considering the profit-driven nature of the IoT ecosystem, it appears likely that security is indeed an afterthought in the IoT ecosystem and vulnerabilities might remain unpatched similarly long or even longer. Automated large-scale analysis allows us to promptly identify vulnerabilities and mitigate them. Moreover, even when automated analysis cannot replace in-depth analysis, it still helps developers to identify issues and address them. Being able to accurately analyze how IoT devices truly behave also informs privacy policy and behavior of (privacy-conscious) consumers. Practical large-scale automated analysis provides the much-needed foundation and knowledge to better understand IoT devices and improve their security and privacy.

IoT Control Infrastructure. The fundamental idea of smart devices is that they coordinate and cooperate with other devices, that is, they do not work in isolation.

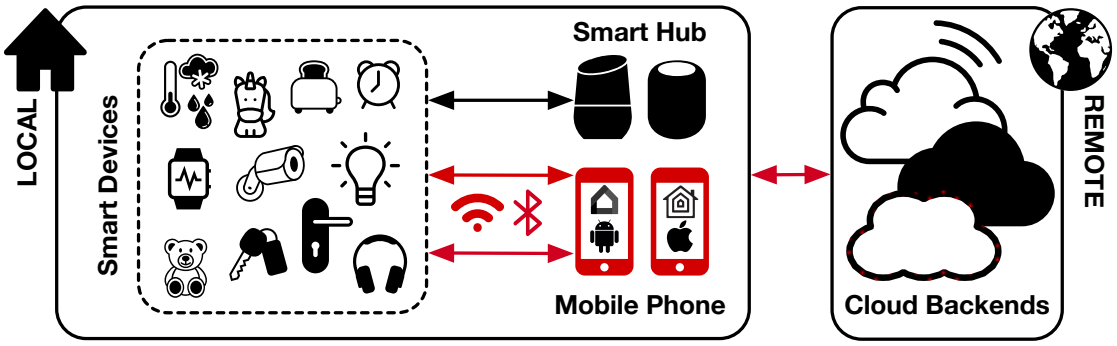


Figure 2.1: Overview of the IoT ecosystem and its command and control scenarios, including apps as intermediaries.

Typically, the devices communicate with companion apps, smart hubs, or remote cloud-based backends (see Figure 2.1), the latter of which may distributed over different regions world-wide [261]. Users interact with the devices almost exclusively via their companion apps. If a device supports Wi-Fi, then the app may communicate with the device over the local network or the Internet. If a device does not support Wi-Fi but only uses Bluetooth, then all device-to-cloud communication needs to pass through the app or a hub. Moreover, due to missing user interfaces, updating a device's firmware frequently happens via the app [315]. That is, the apps play a central role during the *setup, operation,* and *update* of the devices. In fact, many devices cannot be set up without using a device that can run the app. Thus, apps must contain some information about the devices and their behavior, and they provide a unique analysis opportunity.

General-purpose Apps vs. Companion Apps. Compared to general-purpose apps, companion apps face different challenges and introduce new threats. Generally, mobile operating systems restrict access to sensitive data and sensors (e.g., through Android or iOS permissions). However, this does not apply to data collected through smart devices. Users also lack visibility and control over the data the devices collect and share. It is crucial to investigate the threat of collusion between device and app, especially because it circumvents existing defenses and allows to build more accurate user profiles by combining PII and data collected by both [254].

Ads and trackers to collect user data for behavioral targeting appear widely in general-purpose apps [246, 254, 306]. These services are attractive for developers to monetize their apps [118]. For companion apps, one might assume that the business model centers around selling the devices. However, related studies showed that these apps and even devices themselves include ads and tracking [179, 253, 310]. In hindsight, considering the IoT environment and collusion potential, this makes sense: it is additional income. For example, companion apps can interact with the local network to discover and manage devices (a permission often required to set up the device), which is data general-purpose apps have difficulty to collect, and which is also useful for advertisement or tracking [156].

Prior work on network behavior and PII leakage of apps mainly considers traffic sent to remote servers. For IoT devices that use local communication, via Bluetooth or Wi-Fi, app-to-device or device-to-app communication has additional significance [280]. A smart device only using Bluetooth can collude with a companion app to "clean" sensitive data: receive it, encode it in some way, and send it back to the app, which sends it to the tracker. Existing ways to identify and block such behavior in general-purpose apps cannot address the challenges of the IoT environment, like collusion.

2.3 IoTFlow

We introduce IoTFlow, a new static analysis approach for companion apps. We analyze the behavior of Internet of Things (IoT) devices without requiring the physical device.

IoTFlow itself has two main phases (see Figure 2.2): Value Set Analysis (VSA) and Data-flow Analysis (DFA). With VSA, we identify *trigger points*, that is, sources and sinks of interesting (network) activities. This appears trivial at first, but it is important to realize that (1) we expect a substantial amount of communication, as smart devices are meant to communicate and coordinate extensively, and (2) we need to be able to determine the communication endpoint. For example, a user might expect and accept that the companion app shares their location to turn on their heating when they are on their way home. But, most users would likely object if it is sent to an advertisement company. Enumerating all potential sources and sinks will lead to inaccurate results and render the analysis impractical. Instead, we need to distinguish where apps send data, to the device or a remote service, to which services, and utilizing which network protocols. We accurately reconstruct this information leveraging VSA (Section 2.3.1) and use it to identify precise sources and sinks for our DFA (Section 2.3.2).

For reconstructed endpoints, which may be third-party services, we (1) categorize them based on their purpose, (2) analyze their geographic locations, and (3) test for abandoned domains, allowing us to evaluate if communication would be expected and assess their security and privacy impact. A privacy-conscious user within the European Union (EU) may not expect that their device sends data to a non-GDPR compliant country. Similarly, abandoned domains can lead to devices being taken over by attackers [45, 64, 237].

With DFA, we can assess precisely which data companion apps share, with whom they communicate, and how. Specifically, we analyze the data-flow for data from the identified and categorized trigger points and from sensitive data sources (e.g., GPS location).

Motivating Example. Considering the examples in Figure 2.3 and Figure 2.4, we (1) need to reconstruct the destination of the Message Queuing Telemetry Transport (MQTT) broker (Figure 2.4, line 15), and (2) trace the data flow from the Bluetooth source (Figure 2.3, line 3) to where the message is published (Figure 2.4, line 18). An additional challenge is that the data is passed from Figure 2.3 line 6 to Figure 2.4 line 7 via Inter Component Communication (ICC). Traditional approaches would miss this example. However, we can reconstruct the keys of the ICC during VSA and then bridge

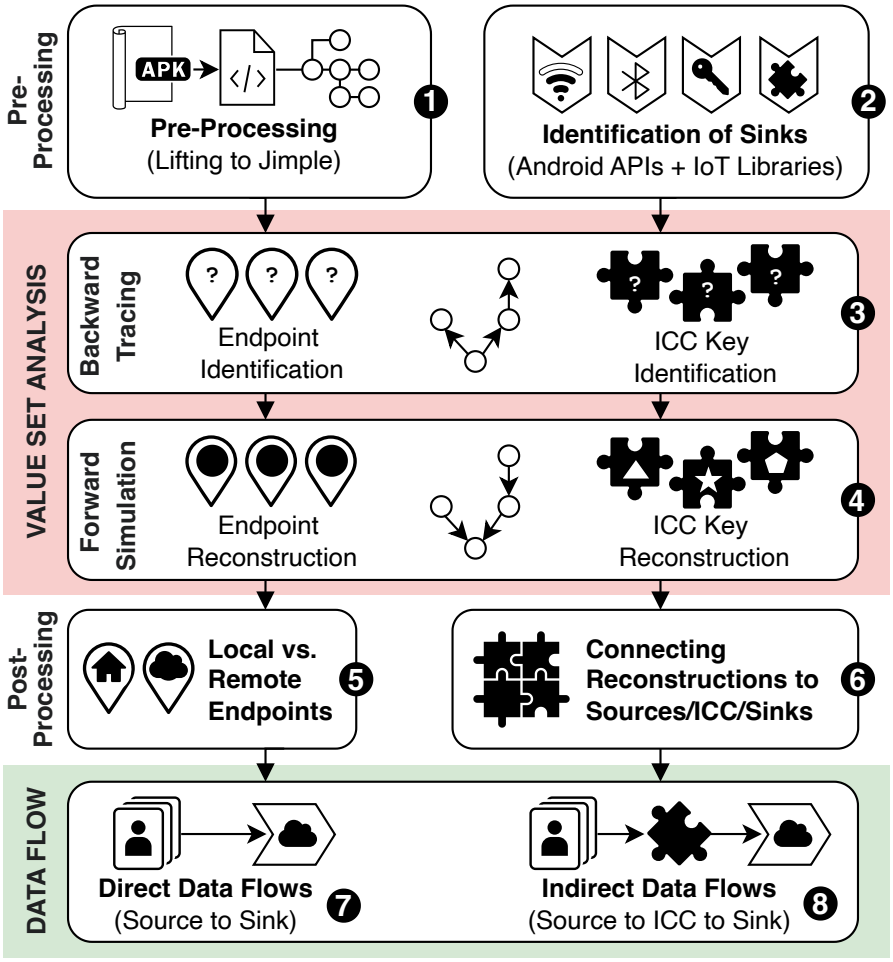


Figure 2.2: Overview of IoTFlow. We use VSA to reconstruct endpoints, cryptographic data, and ICC keys for the flow analysis. We use flow analysis to find data leaks, and connect request/response data with endpoints. With the ICC information of the VSA, we support data flows involving ICC.

the connection via the reconstructed keys, enabling us to perform more precise DFA across the ICC boundary.

2.3.1 Value Set Analysis

Value Set Analysis (VSA) is a program analysis technique to reconstruct values at specific program points. We utilize it to gain insights about the communication of IoT apps and to accurately handle ICC for our DFA. VSA has been used by related work before [332, 333], however, with the focus on reconstructing Application Programming Interface (API) keys or Universally Unique Identifiers (UUIDs) of Bluetooth Low Energy (BLE) to identify vulnerable implementations of the BLE pairing process. That is, related work

reconstructed primarily strings using manually derived rules, while IoTFlow supports arbitrary objects (as is required to precisely reconstruct endpoints, like in Figure 2.4).

Pre-Processing ❶. We implemented our IoTFlow prototype in Java and target Android. We use Soot [305] to parse Dalvik byte code from Android apps. It translates the byte code into the Jimple Intermediate Representation (IR), which simplifies our analysis (e.g., by splitting nested instructions). Notably, both Kotlin and Java Android apps are compiled into Dalvik code, and, in turn, IoTFlow can readily analyze both types of apps. In preparation for the forward computation step, we also translate the Dalvik byte code into Java byte code with `dex2jar` [229] because Java cannot load classes directly from the Dalvik byte code.

Identification of Sinks ❷. IoTFlow starts at interesting sinks tracing backward their values. We analyze network-related sinks from Android, Java, and 19 manually selected popular network communication libraries, focusing on IoT application layer messaging protocols (e.g., MQTT, Constrained Application Protocol (CoAP), Advanced Message Queuing Protocol (AMQP), and Extensible Messaging and Presence Protocol (XMPP)) [25, 224]. Additionally, we consider ICC and cryptographic methods as sinks. We later use the reconstructed ICC information to bridge the ICC boundary during DFA. As apps might encrypt data before sending it, we also examine cryptographic methods.

Backward Tracing ❸. We trace back through the program, starting at the identified sinks to all program points where the app modifies the values we are interested in. Naturally, this yields an over-approximate trace set. For example, if we want to reconstruct the parameter passed to `MqttManager` in Figure 2.4, then our reconstruction starts at line 15 (following ←→). We trace back the value of `config.endpoint` to line 14, to line 8, to lines 1–3, until we have traced all variables on which `config` depends.

Forward Simulation ❹. In the next step, we reconstruct the actual value set. Here, we must reconstruct arbitrary objects passed to the sinks, or we would miss the value of `config` in our example. That is, only reconstructing string operations is insufficient. Instead, we adapt our value reconstruction to handle arbitrary objects from any classes defined by the app, such as the `MqttConfig` class. We utilize reflection and forward simulate the backward trace, using the classes and methods as the app would do while normally executing it. Using reflection for simulating execution paths has a further advantage: We can handle code where the app itself uses reflection, which prior work cannot. However, reflection also introduces new challenges that we need to address:

(1) **Android Methods.** Some data might not be available statically, like user input. Additionally, we cannot simulate Android methods with reflection because only stub implementations are available and we use placeholders instead (e.g., intents, shared preferences, and database).

```
1  String BLE_DATA = "device";
2  @Override
3  void onCharacteristicRead(BluetoothGattCharacteristic bgc, /*...*/) {
4      Intent intent = new Intent(DeviceActivity.class)
5      byte[] value = bgc.getValue();
6      intent.putExtra(BLE_DATA, parseData(value));
7      this.startActivity(intent);
8  }
```

Figure 2.3: Simplified example code that reads device data via BLE and sends it via an Intent (ICC). Arrows on the left show VSA, and arrows on the right show DFA. We reconstruct the ICC key "device," marked yellow, via VSA ⇠⇢ (line 1). The ⟶ arrows show the data flow from source to the ICC sink, via purple statements. The flow continues green in Figure 2.4.

(2) **Non-Terminating Methods.** Simulating arbitrary methods with reflection can also lead to non-termination, such as when it waits for an IoT device to connect. We mitigate this issue by terminating it after two seconds. We determined this threshold empirically as a trade-off between precision and time. In practice, most instructions finish within a fraction of a second.

(3) **Partially Reconstructed Values.** Partially reconstructed values can cause us to miss values. We may simulate a substring operation, but the analysis does not reconstruct the whole base string because parts depend on dynamic values that we cannot determine statically. This can then result in an out-of-bounds exception, which would cause us to miss more values. For example, if a URL obtained dynamically would contain a 32 character device serial number, but our placeholder is `from_pref`, then the analysis may cause an out-of-bounds exception if it accesses index 9. We mitigate this issue by preempting the calls that can cause such issues and expand the value on demand. Notably, this is not limited to string operations, but also extends to accessing arbitrary member fields of objects. For missing parameters or base objects, we attempt to create them with their default constructors. For primitive data types (e.g., boolean, int, or float), we assign default values.

Local vs. Remote Endpoints ❺. A unique aspect of companion apps is that their communication can be local, to connect to IoT devices or hubs, or remote, to connect to remote backends. We need to distinguish these classes to answer what data they share, how, with whom, and what the security and privacy impact is. Therefore, we categorize endpoints as certainly local or possibly remote. That is, we identify local connections by checking whether a reconstructed endpoint points to a local IP address, a broadcast address, a multicast address, or the domain originates from user input (`fromUI.local`). We consider all other endpoints as remote.

2.3.2 Data-Flow Analysis

In the second phase of IoTFlow, we use DFA to trace data flows from IoT devices and sensitive Android methods. IoTFlow builds on FlowDroid [26], which is a data-flow

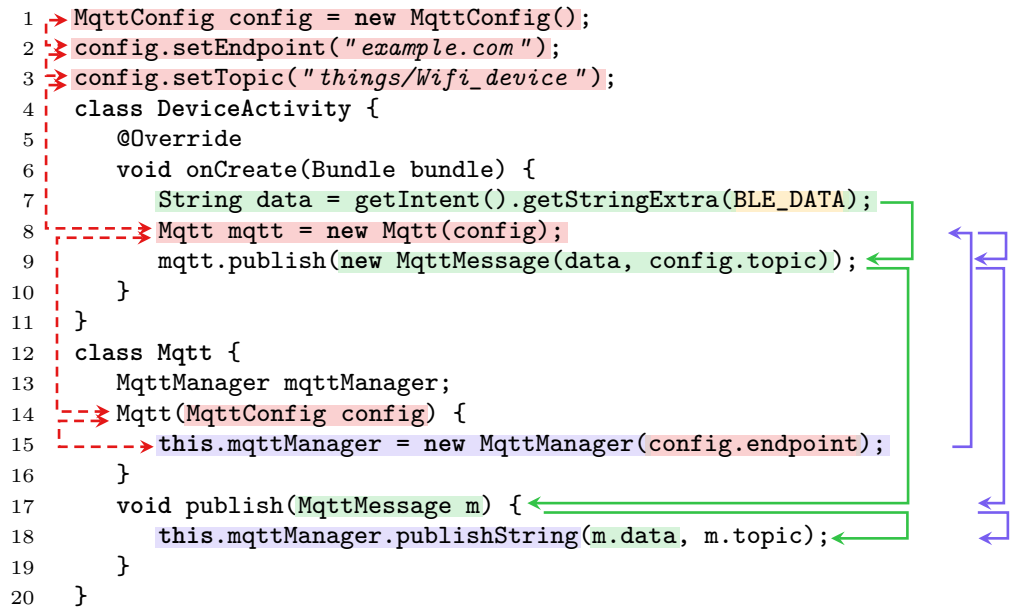


Figure 2.4: Simplified example code of an activity that receives BLE data and publishes it via MQTT. Arrows on the left show identification and reconstruction via VSA, marked ←→. Arrows on the right show DFA. Connecting reconstructions are marked ⟶, via blue statements. The data flow from ICC source to sink, completing the flow from source to ICC sink of Figure 2.3, is marked ⟶, through green statements. We highlight the reconstructed ICC key "device" yellow again.

framework for Android. We extended it to address the unique challenges of the IoT ecosystem. We (1) connect reconstructed endpoint information to data sent or received, similar to pointer analysis, and (2) trace flows across ICC.

Considering how modern apps work internally, we must pay particular attention to ICC. It is now the recommended way for app components to communicate with each other and often used, which is why tracing data flow across it is crucial. Theoretically, FlowDroid supports ICC via ICCTA [167]. However, ICCTA cannot generate ICC models for current Android apps [204, 328], which prevents FlowDroid from tracing flows through ICC. IoTFlow addresses this blind spot by treating ICC as sources and sinks, and connecting an ICC sink (writing to a key) to the corresponding ICC sources (reading from the key) by reconstructing the key used in ICC through VSA.

Connecting Reconstructions ❻. After reconstructing network endpoint information with VSA, we must connect them to the points where the app adds data to the request objects or receives a response, as these might be different from where the endpoint is set. For example the endpoint might be set during initialization of a connection object

that is later used (repeatedly) to send or receive data. We identify the points where the app receives data and use the receiving statement as communication trigger points. Similarly, we need to connect a request's destination with the request's data when the request is executed. We do so using multiple data-flow analysis runs, which we split by method type for easier parallelization (e.g., MQTT, UDP, or CoAP). Returning to Figure 2.4, we previously reconstructed the MQTT broker endpoint via VSA (line 15). For our DFA in the next step, we now associate the MQTT broker endpoint (line 15) to the sink `publishString` (line 18) (marked →).

Direct Data Flows (Source to Sink) ❼. We are interested in data flows from sources that are (1) Bluetooth, (2) responses from the local network, or (3) sensitive Android methods. We trace them to (1) ICC sinks and (2) remote sinks, that is, data leaks.[1] Bluetooth data is interesting as it may contain data from smart devices and local network communication is likely data from smart devices.

Crucially, we need to treat flows to and from the same method differently depending on the context and how the app uses the method (e.g., we want to analyze local network responses but ignore responses from remote endpoints). Thus, we extended FlowDroid to support context-sensitive flow analysis. We precisely identified the methods and the context that we need to consider as trigger points with the help of our VSA and by Connecting Reconstructions ❻, which we can utilize to understand potential data leaks.

We focus first on three types of straight-forward immediate flows: (1) Bluetooth to network, (2) local network to network, and (3) sensitive data to network. Additionally, we trace sources to ICC sinks, to analyze flows across ICC, giving us three more flow types: (4) Bluetooth to ICC, (5) local network to ICC, and (6) sensitive data to ICC. Considering our example Figure 2.3, here, IoTFlow identifies the flow (marked →) from the Bluetooth source `bcg` in line 3 via line 5 to line 6, where the data `value` is passed to the intent using the key `BLE_DATA` (reconstructed via VSA, marked ↔).

Indirect Data Flows (Source to ICC to Sink) ❽. Finally, we need to follow up on the flows we identified that have an ICC sink, to properly bridge the ICC boundary. We trace the additional flow type (7) ICC source to network sink, and then precisely connect the new flows with previously identified flows of types (4)–(6). This allows us to discover and analyze data leaks involving ICC. For our examples Figure 2.3 and Figure 2.4, based on Direct Data Flows ❼, we identified a flow from Bluetooth to ICC using the key `BLE_DATA`. In Figure 2.4, using our indirect flow analysis, we now identify the flow (marked →) from the ICC source `getStringExtra()` in line 7 to line 9 to line 17 to line 18, where the app sends the Bluetooth data to the MQTT broker. Last, we connect the new ICC to network flow to the previously identified Bluetooth to ICC flow leveraging the VSA reconstructed ICC keys, giving us the indirect data flow that crosses the ICC boundary from Bluetooth to ICC to network.

[1] Full list of sources and sinks: https://github.com/SecPriv/iotflow/tree/main/config

2.4 Insights into the IoT Ecosystem

We evaluate IoTFlow on 10,836 apps on an Ubuntu 20.04 machine with 48 physical CPU cores (96 cores with hyper-threading, 2x Intel(R) Xeon(R) Gold 6342 CPU) and 1,024 GiB RAM. We limit the memory for the analysis of each app to 150 GiB (`-Xmx150g`).

2.4.1 Dataset

Verified Companion Apps. We analyze IoTFlow on 9,889 unique IoT companion apps that were verified manually by prior work as part of three individual datasets [149, 197, 201]. We refer to our consolidated dataset as IoT-VER. It contains 455 apps collected by Neupane et al. [201] for studying if apps follow best practices, 5,100 apps that Jin et al. [149] used for the training, validation, and testing of IoTSpotter, and 6,208 apps that Nan et al. [197] collected and manually verified for IoTProfiler. Three quarters of the IoTProfiler apps are from the Google Play Store (74.6%), the remaining apps are from third-party stores. We did not augment these datasets with additional apps to not fragment the IoT companion app dataset space, which we deem important for reproducibility. Unfortunately, the public IoTSpotter dataset is incomplete and it misses 128 apps. Neupane et al.'s dataset misses two apps for which only the package name is available. We excluded these apps from our dataset.

All three datasets have 118 apps in common. IoTSpotter and IoTProfiler share 1,430 apps. The dataset of Neupane et al. shares 57 apps with IoTSpotter and 21 apps with IoTProfiler. If multiple datasets contain the same app, we only analyze the most recent version, the app with the highest version code, since it is monotonically increasing [18]. Our consolidated dataset IoT-VER contains 9,889 apps, unique by their package names.

Popular General-purpose Apps. We also downloaded 1,000 popular apps and games from the *top selling free* category of the Google Play Store in January 2022, which we use to show the differences between IoT companion apps and other apps. We manually removed companion apps from the dataset and refer to the remaining 947 apps as GP-2022. Two researchers independently classified each app based on its metadata in the Play Store. If they disagreed, they studied it in-depth until they reached an agreement.

2.4.2 Performance

We first discuss the performance of IoTFlow on our datasets (see Table 2.1). In addition to the total run time, we investigate the required time separately for VSA and DFA. On average, general apps take almost five times as long to analyze as companion apps (125m31s vs. 26m23s). This difference is even more pronounced when considering the median (129m36s vs. 6m51s): the processing time for companion apps is almost 20x faster than for general apps. Reasons may be the larger code base of general-purpose apps or that they tend to have more sources and sinks that we need to consider. Overall, we consider a median analysis time of less than 7 minutes and an average analysis time of approximately 26 minutes practical.

VSA Performance. We allow up to 600 backward traces for each identified statement to prevent long-running analyses. Increasing the number of backward traces typically leads to more combinations of the same data, like request parameters. Each backward trace has up to 300 steps. We determined these thresholds empirically, observing a reasonable trade-off between resources and precision. Additionally, we configure timeouts for backward tracing (15 minutes) and forward computation (20 minutes). Our analysis only triggered the backward timeout when analyzing 11 (0.1%, all from GP-2022) apps and the forward timeout for 304 (2.81%; 155, 1.57% IoT-VER and 149, 15.73% GP-2022) apps, which we consider reasonable. Higher thresholds could lead to more flows.

Data-Flow Performance. For DFA, we increased the timeout suggestions by the FlowDroid authors [33] by 50%. We set the FlowDroid callback collection timeout to 7m30s and the timeout for flow analysis to 15m. Our analysis triggered the callback timeout for 2,432 apps (22.44%) and the flow analysis timeout for 3,004 apps (27.72%). Separating the two datasets, 1,847 companion apps (18.68%) and 585 general-purpose apps (61.77%) triggered the callback timeout, while 2,484 companion apps (25.12%) and 520 general-purpose apps (54.91%) triggered the flow analysis timeout.

2.4.3 How Companion Apps Communicate

To answer *RQ1: How do companion apps and devices communicate?*, we identify device-to-app communication and the network protocols, and we study certificate pinning.

Direct Device Communication

First, we analyze the reconstructed values for indicators of direct communication with the devices, such as local IP addresses, broadcast, and multicast addresses, user-configurable addresses (i.e., endpoints from user input; marked as fromUI.local), and Bluetooth permissions. The latter indicates that the devices themselves might not have Wi-Fi capabilities, but that they use the companion app as a gateway to access the Internet. Some devices may also spawn their own Wi-Fi network that the phone needs to join for pairing. Within the network, the device has a fixed address known by the companion app. The apps can also use broadcasts to discover devices in local networks, for example apps use Universal Plug and Play (UPnP) to find devices that support screen mirroring. A fourth method is asking the user directly. Table 2.2 summarizes our findings.

IoT-Verified. 6,355 (64.26%) apps declare at least one Bluetooth permission. We find a local IP address in 1,483 (14.99%) apps, a broadcast or multicast addresses in 452 (4.57%) apps, and addresses from user input in 123 (1.24%) apps. Among broadcast and multicast addresses, we found the broadcast address 255.255.255.255 (2.44%) most often, followed by the multicast DNS (mDNS) 224.0.0.251 (1.28%), and UPnP's 239.255.255.250 (0.75%). Besides the IPv4 addresses, we found three (0.03%) IPv6 multicast addresses.

Table 2.1: *Dataset and Performance Overview.* We show for the VSA, Flow Analysis, and the total time (VSA+Flow Analysis), the average time (Avg.), median time (Med.), and standard deviation (Std.) per app in minutes [minutes:seconds].

		VSA			Flow Analysis			Total		
Dataset	**# Apps**	**Med.**	**Avg.**	**Std.**	**Med.**	**Avg.**	**Std.**	**Med.**	**Avg.**	**Std.**
IoT-VER	9,889	1:52	5:40	9:46	3:59	21:19	31:29	6:51	26:23	37:20
GP-2022	947	75:53	70:44	36:40	55:57	54:47	40:29	129:36	125:31	65:56

Table 2.2: *Number of Apps using Direct Device Communication.* Indicators are hard-coded local network IP addresses (grouped if found in 30 or more apps), user-configurable addresses (fromUI.local), broadcast and multicast, or Bluetooth.

Address	**IoT-VER**	**GP-2022**
10.*.*.*	716 (7.24%)	12 (1.27%)
10.0.0.172	516 (5.22%)	1 (0.11%)
10.0.0.200	438 (4.43%)	
10.10.2.2	48 (0.49%)	7 (0.74%)
other	242 (2.45%)	12 (1.27%)
172.16-31.*	103 (1.04%)	4 (0.42%)
172.17.0.1	49 (0.50%)	1 (0.11%)
other	56 (0.57%)	3 (0.32%)
192.168.*.*	746 (7.54%)	4 (0.42%)
192.168.0.1	115 (1.16%)	
192.168.1.1	180 (1.82%)	2 (0.21%)
192.168.1.3	36 (0.36%)	
192.168.4.1	77 (0.78%)	
other	518 (5.24%)	2 (0.21%)
fe80	3 (0.03%)	
Multicast and Broadcast	452 (4.57%)	4 (0.42%)
224.0.0.251	127 (1.28%)	1 (0.11%)
239.255.255.250	74 (0.75%)	
255.255.255.255	241 (2.44%)	4 (0.42%)
IPv4 other	93 (0.94%)	
IPv6 other	3 (0.03%)	
fromUI.local	123 (1.24%)	1 (0.11%)
Bluetooth	6,355 (64.26%)	180 (19.01%)

General-purpose Apps. We observe a significant lower number for all four direct device communication indicators for general-purpose apps in GP-2022. We find local IP addresses in only 2.21% of apps, compared to 14.99% in IoT-VER. Similarly, broadcast and multicast addresses drop from 4.57% in IoT-VER to 0.42% in GP-2022. Only one (0.11%) address depends on user input in GP-2022, compared to 123 (1.24%) addresses in IoT-VER. The number of apps requesting Bluetooth permissions also decreased from 64.26% in IoT-VER to 19.01% in GP-2022. These findings strengthen our assumption that our direct device communication indicators are indeed meaningful.

Takeaways. We identified four strategies apps use to communicate locally with smart devices, and we show by comparing them to general-purpose apps that they are indeed specific to companion apps. Identifying this kind of communication helps security and privacy analyses (see Section 2.4.5). Prompting the user for the device location and using multicast can be dangerous and prone to misconfigurations. Users might make devices unwittingly accessible over the Internet [46]. A Shodan [276] query for open port 554 returns 78,858 results of exposed cameras, suggesting that misconfigured devices accessible remotely are a common issue. Attackers can also sniff broadcast packages or mimic the legitimate device to act as a Monkey-in-the-Middle (MITM) [79]. Finally, we note that any information about local network devices is sensitive and can be abused for advertising and tracking purposes [156].

We recommend to use device discovery and avoid requiring user configurable addresses, to reduce the risk of accidental misconfigurations [80]. Apps should also respect users' privacy and not send local network information to remote servers. In fact, they should prefer local communication over cloud communication whenever possible, as remote requests can reveal usage patterns to others.

URL Protocol Schemes

We identify network protocols based on the values we reconstructed through VSA. First, we analyze the URL schemes of the endpoints that apps communicate. Second, as we reconstruct endpoint information for libraries for *AMQP*, *MQTT* and *XMPP* communication, we can draw conclusions about them, even if they do not use specific schemes. Table 2.3 summarizes our results. The row *IoT-related* summarizes the schemes and protocols that are tailored to IoT devices. We group IPP, IPPs, RMTP, and VNC as *IoT-other* as we found them only in one or two apps, and we group protocols from IANA's list of URI schemes [142] that are less interesting for our use case (e.g., *service, about, info*) as *Other*. Overall, for IoT-VER, we reconstructed schemes in 7,113 unique apps for remote endpoints and in 871 apps for local communication.

HTTP(S). We find that apps still widely use plain HTTP. Our numbers represent an upper bound as we do not know how many actual connections occur over HTTP since we base our results on statically reconstructed endpoints. In practice, HTTP might be upgraded by default, but even if used as a fallback, HTTP can lead to security and privacy issues through protocol downgrade attacks.

Our results show that a high proportion of HTTP traffic, compared to HTTPS traffic, is for local communication. This is not surprising: local communication might appear safe, and deploying TLS properly for IoT devices remains challenging [230]. Nevertheless, even if communication is local, TLS protects against eavesdroppers, which is important as devices use broadcast media like Wi-Fi.

MQTT Endpoints. Smart devices have unique usage scenarios and requirements, such as device-to-device communication and energy efficiency. Traditional communication

Table 2.3: *Number of Apps with Reconstructed URL Protocol Schemes.* Percentages are relative to the numbers of total apps with at least one scheme. For IoT-VER, we identified schemes for 871 local endpoints and 7,113 remote endpoints. For GP-2022, we identified schemes for 14 local endpoints and 898 remote endpoints. Protocols marked with a star (*) are based on IoTFlow identifying the corresponding libraries.

	Local		**Possibly Remote**	
Protocol	**IoT-VER**	**GP-2022**	**IoT-VER**	**GP-2022**
Android			29 (0.41%)	184 (20.49%)
File	4 (0.46%)		2,180 (30.65%)	578 (64.37%)
FTP	1 (0.11%)		8 (0.11%)	
HTTP	788 (90.47%)	13 (92.86%)	4,901 (68.90%)	639 (71.16%)
HTTPS	81 (9.30%)	2 (14.29%)	5,445 (76.55%)	885 (98.55%)
IoT-related	49 (5.63%)		315 (4.43%)	1 (0.11%)
AMQP*			6 (0.08%)	
Cast			4 (0.06%)	
CoAP*	2 (0.23%)		9 (0.13%)	
CoAPs*			2 (0.03%)	
MQTT*	27 (3.10%)		158 (2.22%)	1 (0.11%)
Palm			58 (0.82%)	
RTSP	1 (0.11%)		15 (0.21%)	
RTSPs			2 (0.03%)	
TV			9 (0.13%)	
URN	8 (0.92%)		18 (0.25%)	
XMPP*	11 (1.26%)		29 (0.41%)	1 (0.11%)
IoT-other			4 (0.06%)	
JAR			65 (0.91%)	1 (0.11%)
SMB	4 (0.46%)		62 (0.87%)	2 (0.22%)
WS	14 (1.61%)	7 (50.00%)	130 (1.83%)	10 (1.11%)
WSS	2 (0.23%)		138 (1.94%)	14 (1.56%)
Other	7 (0.80%)		1,604 (22.55%)	830 (92.43%)

protocols do not satisfy these requirements. New protocols can fit these demands, but they can also threaten security and privacy, especially if they were designed without considering an adversarial environment or if developers make wrong assumptions about them. One protocol used often by IoT devices is the *Message Queuing Telemetry Transport (MQTT)* protocol. In practice, it often lacks authentication and authorization, allowing attackers to access user data or take over devices [148, 325]

MQTT is the most widespread IoT-specific communication protocol for IoT-VER. We reconstructed 147 MQTT endpoints in 176 apps, of which nine represent local IP addresses. We verify that the remaining 138 remote endpoints are valid by opening a connection to them. To not raise any ethics concerns, we open and immediately close the connection, and we do *not* perform any action (e.g., subscribing to a topic). We use the Python Paho library [168] for our test and base our results on the return code: If the connection is successfully established (return code 0) or an error related to connection parameters is returned (return codes 1 to 5), we consider the endpoint as reachable and valid.

We connected successfully (return code 0) to 74 MQTT endpoints (53.62%). To further investigate the remaining 64 endpoints, we probed for other ports typically used for

MQTT (1883 and 8883) with `nmap` [174]. Seven endpoints were *closed* and 37 were *filtered*, meaning our connection attempts were prevented at the network level. One reason may be geographical restrictions. The remaining 20 endpoints were unresponsive to ICMP echo requests and we consider them unreachable.

MQTT Credentials. IoTFlow can also reconstruct authentication credentials. Hard-coding credentials into the app can lead to attacks on the integrity and confidentiality of data by allowing an attacker to connect and publish or subscribe to topics (e.g., modifying a parameter of a physical actuator). We reconstructed 30 unique usernames and 34 unique passwords in IoT companion apps.

MQTT Topics and Payloads. Our analysis can reconstruct the topics (i.e., topics for which the phone or IoT device should receive messages) and message payload formats (i.e., the format of messages shared between phone, IoT device, and the cloud). We found 726 topic names and 330 payload formats. While we may miss dynamic values from communication with the device, the information we gain is valuable to understand the behavior of IoT apps and devices.

Other IoT Protocols. We also identified other IoT protocols in IoT-VER apps, namely XMPP, AMQP, and CoAP. Among the 36 XMPP endpoints we identified, we could connect to five, the port was filtered for six, and the remaining 25 endpoints were unresponsive to ICMP echo requests. For the six identified AMQP endpoints, we could connect to two, we received an authentication error for one, and the AMQP-specific port was closed or filtered for the remaining endpoints. For the two CoAP endpoints, one was a local IP address, but we successfully reconstructed 55 unique URL paths used to specify the location of resources on the server.

General-purpose Apps. For GP-2022, we reconstructed local addresses only in combination with HTTP, HTTPS, and WS (WebSocket). Like for companion apps, only a minority uses HTTPS locally (14.29%). However, unlike IoT apps, nearly all apps (98.55%) use it for remote communication. Unsurprisingly, general apps do not use IoT protocols. Only a card game app uses MQTT and XMPP.

Takeaways. Despite its insecurity, we found widespread HTTP adoption across IoT-VER apps. The situation improves for GP-2022 apps as almost all apps communicate over HTTPS. However, most local communication does not adopt TLS to secure the connection in both datasets. We also identified how IoT-specific protocols (MQTT, AMQP, XMPP, CoAP) are used, and we reconstructed crucial information, like credentials and topics.

Generally, apps should not use hard-coded credentials but generate them individually during initialization, use limited and narrow authorization scopes, follow best practices (e.g., encrypting Android shared preferences [19]), and encrypt all communication (e.g., via TLS, but preferably end-to-end).

Table 2.4: *Certificates and Pinning.* The first rows show the number of apps in which we found pinning, certificates, and apps containing expired or self-signed certificates. The remaining rows show the corresponding certificates. The number of expired certificates at the time of download is a lower-bound for IoT-VER because it is not always known.

		IoT-VER	**GP-2022**
Apps	Pinning	385 (3.89%)	111 (11.72%)
	Certificates	1,207 (12.21%)	119 (12.57%)
	Expired (at download)	474 (4.79%)	49 (5.17%)
	Expired (May 2023)	822 (8.31%)	59 (6.23%)
	Self-Signed	1,042 (10.54%)	91 (9.61%)
Certificates	Total Number	31,285	1,837
	Expired (at download)	3,976 (12.71%)	268 (14.59%)
	Expired (May 2023)	9,129 (29.18%)	321 (17.47%)
	Self-Signed	18,018 (57.59%)	684 (37.23%)
	Avg per App (Std)	3.21 (20.97)	1.94 (14.59)

Pinning and Certificates

An additional aspect of how companion apps secure their communication is certificate pinning. It is a contentious topic: while OWASP [313] suggests it to verify the host's identity, Google [14] advises not to adopt it because of issues deriving from certificate changes. However, determining whether it is good or bad is out of scope of our work.

We use Pradeep et al. [239]'s approach to identify pinning and the corresponding certificates by analyzing the Network Security Configuration (NSC) specified in the Android Manifest and the certificates included in the app. Table 2.4 summarizes our results.

IoT-Verified. More companion apps include certificates (12.21%) than use pinning (3.89%). On average, each app includes 3.21 certificates. More than half of the certificates in IoT-VER were self-signed, possibly to communicate with IoT devices. We also investigate if certificates were expired when the apps were downloaded. If the download date is unknown, we infer it based on the app versions. Our numbers are lower bounds for the apps from IoTProfiler and Neupane et al. because we assume apps were downloaded on the first day of the year when they could have been downloaded. We treat certificates as expired if their expiration date is before 2018 for IoTProfiler and before 2021 for apps by Neupane et al. Apps might be downloaded later, but this does not threaten validity as our numbers are a lower bound. For IoTSpotter apps, the download date is available as they were downloaded via AndroZoo [8]. Overall, 12.71% certificates were expired when the apps were downloaded (in 4.79% apps). In May 2023, 822 (8.31%) apps contain 9,129 (29.18%) expired certificates. Intuitively, expired certificates point to poor security practices and can even prevent communication.

General-purpose Apps. Compared to IoT-VER, more apps adopt pinning (11.72% vs. 3.89%), but the same proportion of apps include certificates (12.57% vs. 12.21%). On average, however, they include less certificates (1.94 vs. 3.21). One reason may be the lower number of self-signed certificates. While more than half of all (57.59%) certificates are self-signed for companion apps, only slightly more than one third (37.23%) of certificates are self-signed for general-purpose apps. At the download date, 14.59% certificates were already expired. IoT-VER's older download date could be a reason for the increase in expired certificates in May 2023 (29.18% vs. 17.47%).

Takeaways. Comparing included, expired, and self-signed certificates, we can conclude that more certificates do not lead to better security. Companion apps include substantially more certificates than general-purpose apps, and proportionally significantly more of them are expired or self-signed. Interestingly, fewer companion apps adopt the controversial practice of certificate pinning.

Generally, developers should renew certificates well before expiration, as users may not install updates immediately. Further care is needed for self-signed certificates as apps must add code to explicitly trust them, or Android will prevent the communication that attempts to use them. Worse, doing so incorrectly, like instructing TrustManager to trust every certificate, enables MITM attacks [14].

2.4.4 With Whom IoT Apps Communicate

After analyzing how apps communicate, we investigate *RQ2: Who are companion apps communicating with?* We categorize the reconstructed fully-qualified domain names (FQDNs) and effective top-level domains+1 (eTLD+1) to spot potentially problematic endpoints, like trackers, investigate where data is sent geographically, and analyze if endpoints are vulnerable to domain takeovers.

Advertisers and Trackers

We classify the FQDNs to learn who receives data from the app and, via the app, from the devices. We use the domain lists by Ren et al. [254], which they compiled from various ad-blocking lists. Additionally, we use the Exodus tracker list [96]. Table 2.5 summarizes our results.

IoT-Verified. Overall, 2,959 (29.92%) apps include 487 unique advertisement FQDNs and 1,647 (16.65%) apps use 114 analytic-related FQDNs. Although they belong to different categories, both domains behave similarly by collecting user information. We also reconstructed 410 FQDNs pointing to Content Distribution Networks (CDNs) in 1,165 (11.78%) apps. Additionally, we identified 84 social network FQDNs shared across 1,046 (10.58%) apps, with the respective standard deviation indicating that if apps use one, they often use more. The remaining 7,248 FQDNs in 4,917 (49.72%) apps do not fall in our categories and we label them *Other*.

Table 2.5: *Categorized Endpoints by IoTFlow for IoT-VER (IoT), GP-2022 (GP), and Comparison between IoTFlow (IF) and Dynamic Analysis (DA).* We report with the number of unique FQDN per dataset and shared between them, prefixed with # the number of apps with at least one domain per category, the average number of domains per app, and the standard deviation.

	Large-scale IoTFlow Analysis							**IoTFlow vs. Dynamic Analysis**				
	IoT	**GP**	∩	**# IoT**	**# GP**	$\bar{IoT}$ **(SD)**	$\bar{GP}$ **(SD)**	**IF**	**DA**	∩	∩ **TLDs**	**# Apps**
Ads.	487	279	141	2,959 (29.92%)	848 (89.55%)	0.76 (1.73)	6.33 (4.92)	34	56	10 (12.50%)	9 (39.03%)	13 (100%)
Analytics	114	96	78	1,647 (16.65%)	679 (71.70%)	0.38 (1.03)	2.89 (3.68)	12	16	6 (27.27%)	5 (45.45%)	9 (69.23%)
CDNs	410	97	26	1,165 (11.78%)	733 (77.40%)	0.17 (0.64)	1.02 (0.97)	9	16	- (-)	- (-)	9 (69.23%)
Crash Report	4	3	3	195 (1.97%)	192 (20.27%)	0.02 (0.15)	0.23 (0.49)	6	1	1 (16.67%)	1 (50.0%)	3 (23.08%)
Social Net.	84	49	24	1,046 (10.58%)	137 (14.47%)	0.37 (1.45)	0.23 (0.79)	11	6	1 (6.25%)	1 (14.29%)	2 (15.38%)
Other	7,248	1,420	271	4,917 (49.72%)	685 (72.33%)	2.08 (4.57)	3.52 (5.29)	80	84	17 (11.56%)	19 (25.33%)	12 (92.31%)

Table 2.6: *Geographic Location of Network Endpoints.* The numbers show the amount of endpoints from each location and the ratio to the overall number of endpoints.

	US	**CN**	**EU**	**Asia**	**UK**	**RU**	**Other**
IoT-VER	17,283 (46.09%)	10,221 (27.25%)	5,380 (14.35%)	3,166 (8.44%)	438 (1.17%)	113 (0.30%)	901 (2.40%)
GP-2022	10,606 (79.69%)	181 (1.36%)	1,786 (13.42%)	207 (1.56%)	47 (0.35%)	299 (2.25%)	183 (1.38%)

General-purpose Apps. The average number of advertisement and tracker FQDNs per general-purpose app is 6.33, eight times higher than per companion app (0.76). Additionally, they occur in almost all apps (89.55%), while they only occur in less than one third (29.92%) of companion apps. The situation for analytics FQDNs is similar (71.70% vs. 16.65%). Most analytics and crash reporting FQDNs are shared between the two datasets, while FQDNs from other categories are mainly limited to one dataset.

Takeaways. The large number of 7,248 *Other* FQDNs in companion apps combined with the low number of 271 FQDNs shared with general-purpose apps (3.25% of all IoT FQDNs) suggests that many are IoT-specific. Prior work observed a low coverage of existing filter lists for IoT domains [179, 299], highlighting the need for more scrutiny by future work into who receives data by these apps. Prior work showed that users value IoT security and privacy [87] and are willing to pay a premium for devices that respect their security and privacy [88]. Indeed, not using ad services or trackers could be a unique and convincing differentiating value proposition for IoT devices, especially because users already pay for the device.

Geographic Location

Next, we determined the location of the reconstructed FQDNs to study where data is sent and which countries receive IoT data. We first resolved the FQDNs to determine the location of the IPv4 addresses against the allocated blocks [90]. We resolved them from Vienna, Austria, which is in a jurisdiction that has implemented the EU's General Data Protection Regulation (GDPR) [93]. Notably, due to geographic split horizon DNS (GeoDNS), the resolved IP addresses may differ for other vantage points. Table 2.6 shows aggregated geographic regions. We perform our analysis at the FQDN level because the FQDN endpoint receives the data. This granularity is also important because FQDN and eTLD+1 locations can differ. For example, `xiaomi.com` is hosted in China, but `ru.register.xmpush.xiaomi.com` is hosted in Russia.

We make multiple observations comparing endpoint locations between companion and general-purpose apps. First, there are substantially fewer endpoints for companion apps in the US (46.09%) than for general-purpose apps (79.69%). The difference (33.6 percentage points) stems almost exclusively from more Chinese endpoints (27.25% to 1.36%, 25.89

pp), with the remainder (7.71 pp) being nearly covered by other Asian countries for IoT-VER (8.44% compared to 1.56%, 6.88 pp). Other regions remain mainly stable.

Takeaways. The scattered geographic location of endpoints might raise privacy concerns. Countries have implemented various data protection regulations with stricter or more relaxed requirements. For example, the EU's GDPR [93] is considered the world's strongest privacy law. If a European user downloads an app that contacts endpoints outside the EU, their data is subject to GDPR, but the app may transfer it to foreign countries and process it there. This clearly raises privacy concerns and may even be illegal. Moreover, even if no sensitive data is sent directly, metadata can suffice to infer usage patterns, which can be sensitive (e.g., for smart locks).

Abandoned Domains

Domains that could be re-registered but are still used pose severe security and privacy risks for users, as attackers could take them over. A similar argument applies to registered domains, but for which DNS information is stale and where the corresponding IP address could be taken over [45]. We focus on expired domains as they provide attackers with longer-term capabilities. We extract the eTLD+1 from the reconstructed FQDNs to identify abandoned domains. We then resolve the eTLD+1 to test whether they are in use. For domains we cannot resolve, we use `WHOIS` to check if it is registered or free.

IoT-Verified. We identified 136 potentially abandoned domains in companion apps. After manually investigating and removing artifacts, we verified that 67 domains from 73 apps are indeed available for registration. They are in apps for watches, TVs, cars, health equipment, security and baby cameras, lights, and locks. An attacker could take over these devices by registering the domains.

We also investigated if the 73 apps can still be downloaded from the Google Play Store. Unavailable apps remain critical, but differently so. They can still impact users as the devices might not have been replaced and they might still connect to those domains. We found that 27 apps (37.0%) are available. Remarkably, one app has over one million downloads, a second app has over 500,000, and three others have more than 100,000. For ten apps, based on the reconstruction information, it is likely the domains receive IoT data. They use IoT information in URLs, such as `ipcDeviceIdList` as a request parameter, or `petinfoDatas/addpet` as a path. Eight apps use abandoned domains to download files, which may be executed or could be device updates. Sixteen domains are API endpoints and also likely receive sensitive data. We responsibly disclosed our findings to developers and the Google Play Store.

Takeaways. Pariwono et al. [231] investigated abandoned domains for general apps, but the dangers can be more serious for IoT devices. Attackers could not only take over the apps and receive Personally Identifiable Information (PII), but they might also be able to control hundreds of thousands of devices, enabling large-scale distributed

denial-of-service attacks and allowing them to create botnets. Our analysis shows (1) that abandoned domains are a real danger in the IoT ecosystem, (2) that they affect a varied range of devices, and (3) what data they receive.

Developers should actively monitor the domains that their apps may contact, including those of third-party libraries. Additionally, old or deprecated domains that may still be contacted should remain registered, as users may depend on outdated app versions, and made inoperable instead of allowing others to register it.

2.4.5 What Data Companion Apps Share

To answer *RQ3: Which data are companion apps sharing (and how)?*, we first report what data apps can access, based on the requested permissions, to understand what data they could share. We then analyze the data flows we extracted to identify leaked data and whether encryption is used to protect data.

Permissions

We extract permissions and *protectionLevel* with Androguard [78]. We focus on permissions with a *protectionLevel* of *dangerous* (permissions protecting sensitive resources) and *privileged* (permissions that third-party apps should not adopt).

On average, GP-2022 apps request more permissions than companion apps (17.54, SD 14.10 vs. 14.26, SD 10.23). For IoT-VER, 8,769 (88.67%) apps request at least one dangerous permission. `WRITE_EXTERNAL_STORAGE` occurs most often (7,209 IoT apps, 72.9% and 559 general-purpose apps, 59.03%). The second and third most frequent permissions are `ACCESS_COARSE_LOCATION` (5,735 IoT apps, 57.99%) and `ACCESS_FINE_LOCATION` (5,529 IoT apps, 55.91%) as in most cases IoT devices also rely on location to operate (e.g., smart watches recording physical activity).

Additionally, 2,604 (26.33%) IoT apps request one or more privileged permissions, while only 73 (7.71%) GP-2022 apps do. Even if system permissions are requested, they will only be granted if the phone is rooted or if the app has a special entitlement (e.g., the phone vendor may grant such an entitlement to their own apps, and they might also produce IoT devices). They can also occur for backward compatibility reasons or be remnants from development that were never removed (e.g., the second most common privileged permission is `READ_LOGS`, which appears in 998 IoT apps).

Finally, 5,660 (57.24%) IoT apps use "non-standard" permissions. The permission occurring the most belongs to Google Cloud Messaging (GCM) (3,656 apps, 36.97%) and is used when receiving a broadcast from GCM. We also find permissions of specific brands, for example, Huawei (60 permissions occur 1,207 times in 424 apps) or Sony (31 permissions 787 times in 424 apps).

Takeaways. General-purpose apps request more permissions than companion apps on average. However, IoT apps use more *privileged* and *dangerous* permissions, with two of

Table 2.7: *Flow Analysis.* We separated the flows by their categories. The ICC columns represent the flows involving any ICC, and the endpoint columns (Endp.) the flows with additional endpoint information. The ratio concerns the number of flows from the category. The app columns show the number of apps with the respective flows and the relation to the apps in the dataset.

	Bluetooth				Local Network				Android			
Dataset	**ICC**	**Endp.**	**Flows**	**Apps**	**ICC**	**Endp.**	**Flows**	**Apps**	**ICC**	**Endp.**	**Flows**	**Apps**
IoT-VER	497 (85.84%)	50 (8.64%)	579	90 (0.91%)	4 (5.33%)	49 (65.33%)	75	53 (0.54%)	2,340 (34.89%)	1,952 (29.11%)	6,706	1,682 (17.01%)
GP-2022									420 (30.75%)	619 (45.31%)	1,366	318 (33.58%)

the most requested *dangerous* permissions being for the user's geographic location.

We recommend to regularly review if permissions are still current, to request the least necessary set of permissions, and to only temporarily acquire them when needed. With new Android updates, permissions might also change, for example, scanning for Bluetooth devices required location permissions only up to Android 12 [17, 119].

Data Flows

To learn more about what data is sent, we analyze the flows we discovered via DFA. Table 2.7 summarize our findings. We distinguish between three flow types based on the destination: Bluetooth, local network, or a sensitive Android API. We determine where the data is sent by connecting the VSA results with the individual flows. Unfortunately, we may not have precise information for all flows for two reasons: (1) VSA might not reconstruct an endpoint precisely, for example, because it depends on dynamic values, (2) we could not connect reconstruction and flow.

We identified data flows from Bluetooth and local network sources only for IoT apps, unsurprisingly, as we have shown that such communication is companion app specific (see Section 2.4.3). Overall, we found 579 flows from Bluetooth sources in 90 apps. Remarkably, 497 (85.84%) of these flows involve ICC, which highlights the need for DFA that is ICC aware, like our approach. We identified endpoint information (i.e., where the data is sent to) for 50 (8.64%) flows. For local network sources, we discovered 75 flows, of which four (5.33%) involve ICC. IoTFlow reconstructed precise endpoint information for 49 (65.33%) of them. Finally, we identified 6,706 flows from sensitive Android API in 1,682 (17.01%) IoT apps, and 1,366 such flows in 318 (33.58%) GP-2022 apps.

Case Study: Smart Grill. Our analysis finds a flow in a companion app for smart grills. The app reads data from the device via BLE, parses it, processes it via an intent, and later sends it to an Amazon AWS endpoint via MQTT. We successfully connected to the endpoint without requiring credentials (anonymously) (see Section 2.4.3). This means that we could potentially receive data from others (for ethical reasons, we did not explore this further).

Case Study: Smart Camera. In a smart camera companion app, we found a flow from `getDeviceId` to a remote endpoint. The app uses the IMEI together with a username and password for authentication. Worth mentioning is also that the app hashes the password using MD5, which is insecure and cryptographically broken. Afterward, the app encrypts the username and password with 3DES, which is also insecure and cryptographically broken. IoTFlow's VSA reconstructed the key, even though the app developers put one byte of the key into a different class file, potentially trying to obfuscate it and avoid regex-based key recovery.

Sharing the IMEI is problematic because users can only change the IMEI by physically replacing the device as it is a non-resettable hardware identifier. Google strongly discourages developers from using any hardware identifiers, including the IMEI [16], and it is also prohibited by Android's user data policy [115]. With Android 10 (API level 29, released in 2019), Google added additional restrictions on the IMEI [16, 20], but around 14.4% of users are still using older versions, allowing apps to access these identifiers [39].

Geographic Location. We also analyze the geographic location of data flows with endpoint information. For IoT-VER, we find 917 (15.04%) flows sending data to Chinese and 604 (9.90%) to US endpoints. The share of flows with US endpoints increases for general-purpose apps (75, 27.88%), while the share for Chinese drops (12, 4.46%), aligned with their distribution (see Section 2.4.4). Positively, as we conducted our experiments from the EU, most destinations are within the EU: 73.80% for IoT-VER and 67.66% for GP. Unfortunately, it also means that more than 25% of destinations are outside the EU, potentially violating GDPR. The situation is worse for Bluetooth-based sources than it is for local network flows. For Bluetooth, 27 endpoints (45.76%) are US endpoints, and for local network flows, 37 endpoints (52.86%) are Chinese endpoints. Our artifact provides more details.[2]

Takeaways. With the help of IoTFlow's combination of VSA and DFA, we identified real-world security and privacy issues in the IoT ecosystem and we discussed what data companion apps leak and where they send it, which we illustrated with two examples.

Following best practices and for privacy reasons, developers should minimize data they collect and use, and only send data if it is truly necessary. Generally, we recommend to process as much data as possible locally, and to encrypt any data leaving the devices.

Encryption Analysis

Finally, we analyze the encryption algorithms apps use and we investigate the reconstructed data passed to cryptographic methods. We reconstructed the algorithms in 812 (85.74%) GP-2022 apps and in 4,069 (41.15%) IoT-VER apps. Table 2.8 summarizes our results. AES is the most widely used encryption algorithm for IoT apps (92.97%) and GP-2022 apps (99.38%). Algorithms that are considered insecure or cryptographically

[2] https://github.com/SecPriv/iotflow/tree/main/scripts/evaluation/dfa

Table 2.8: *Encryption Algorithms.* The number of apps that use the respective encryption algorithm and its relation to the number of apps with encryption algorithms (4,069 IoT-VER, and 812 in GP-2022). Recommended algorithms are marked 🔒. Algorithms considered insecure or broken are marked ⊘.

	Algorithm	IoT-VER	GP-2022
🔒	AES	3,794 (92.97%)	807 (99.38%)
🔒	ChaCha	4 (0.10%)	3 (0.37%)
🔒	Diffie-Hellman	14 (0.34%)	44 (5.42%)
🔒	RSA	16 (0.39%)	
	Serpent	136 (3.33%)	31 (3.82%)
⊘	Blowfish	79 (1.94%)	10 (1.23%)
⊘	DES	1,366 (33.47%)	120 (14.78%)
⊘	3DES	351 (8.60%)	66 (8.13%)
⊘	GOST		5 (0.62%)
⊘	RC4	288 (7.06%)	21 (2.59%)

broken are much more prominent in IoT apps (1,461 apps, 35.80%) using encryption than they are in GP-2022 apps (135, 16.63%).

We also evaluate reconstructed encryption keys. Unfortunately, removing false positive artifacts is highly challenging because it is difficult to determine whether a key is truly used as is. For example, a 16-byte array with all 0 values could be an insecure key, or not initialized. Therefore, our numbers are an upper bound. Overall, we reconstructed hard-coded keys in 2,321 (57.04%) IoT apps and 408 (50.24%) general-purpose apps.

Takeaways. The main differences between IoT apps and GP-2022 apps are in what encryption they use and how they use them. Using hard-coded keys and broken encryption algorithms gives a false sense of security and does not provide security or privacy. Unfortunately, both issues are worse for companion apps.

Beyond using strong encryption algorithms, we recommend initializing encryption keys on demand and storing them securely, for example, with the help of Android KeyStore [15].

2.5 IoTFlow vs. Dynamic Analysis

Our static analysis approach has some limitations, especially because we do not require access to the device. It is crucial to understand what and how much information we can truly reconstruct from companion apps without the device. We verify the accuracy and completeness of the reconstructed values by analyzing and comparing the results we obtained through IoTFlow with our in-depth manual analysis when interacting with apps and devices. To this end, we recorded traffic when using 13 different devices and their companion apps in our lab environment (see Table 2.9).

Our test environment uses a machine running Ubuntu 20.04 with `frida-tools` [245] and `mitmproxy` [69], and a rooted Google Pixel 4 running `frida-server` [245] on Android 12. The machine hosts a Wi-Fi network to which the phone and the devices connect, providing Internet connectivity through an Ethernet connection. We bypass certificate pinning via Frida's built-in scripting. We test companion apps with two strategies: First, *automatic inputs*, we test apps with the Application Exerciser Monkey (AME) [21] for 10 minutes or until they crash, whichever occurs first. We do not expect to trigger complex behavior of IoT devices (e.g., because it would require us to set up the device), but AME remains a common testing technique [62] and we include it for completeness. Second, *manual inputs*, we manually interact with each app for 30 minutes and trigger all functionalities, including pairing and interacting with IoT devices, changing their settings, etc. Indeed, we observed significantly less traffic with AME than with manual interaction, demonstrating the scalability issues of dynamic analysis.

From the observed traffic, we extract requests' domain names, which correspond to who receive data, and resource paths, which correspond to functionality (e.g., API endpoints). We match them based on exact string equivalence exactly between IoTFlow and dynamic analysis. Considering the configuration of our dynamic environment, we also manually match domains and paths by (a) identifying over-approximate placeholders, such as the device product code, serial number, etc. and matching them to concrete dynamic information, (b) generalizing the dynamic system configuration, like language and locale, (c) grouping repeated dynamic data (as they also do not provide new information in the dynamic analysis setting, but provide a false sense of accuracy), and (d) resolving network-level redirects (e.g., DNS-based or IP anycast). We remove analysis artifacts that are *clearly not* related: (a) domains and resources that were requested from outside of the IoT app, such as by the Android operating system (e.g., background update checks), (b) domains and resources that were requested by Android WebView components unrelated to device behavior (e.g., opening a vendor's online shop website), and (c) invalid domain names. We retain all data that cannot be clearly attributed to dynamic analysis artifacts, making our results a lower-bound. Last, as we focus on IoT-related behavior, we manually label data as related if it relates to IoT device behavior, security, privacy, or data exchange. Our artifact provides further details on the identified domains and paths, and their matching.[3]

IoTFlow extracted 214 domains from the 13 companion apps, with a minimum of 3 domains, an average of 16.46 domains, and a maximum of 42 domains per app. With dynamic analysis, we observed 218 domains, with a minimum of 1 domain, an average of 16.77 domains, and a maximum of 48 domains per app. Between static and dynamic analysis, 36 domains match exactly, and we matched 7 additional domains manually.

We categorize all domains using our previous approach (see Section 2.4.4). Table 2.5 summarizes our results and our artifact provides further details.[3] Notably, we find substantially more advertisement domains via dynamic analysis than through IoTFlow.

[3] https://github.com/SecPriv/iotflow/tree/main/dynamic_analysis

Table 2.9: *Tested Devices.* The IoT devices that we tested dynamically together with their device type and package name.

Device	Type	Package Name
Bose QC35	Headphones	com.bose.monet
Divoom Timebox	Alarm Clock	com.divoom.Divoom
Fitbit Inspire 1	Smart Watch	com.fitbit.FitbitMobile
Blaupunkt	Smart Watch	cn.xiaofengkj.fitpro
HHCC FlowerCare	Plant Sensor	com.huahuacaocao.flowercare
Hama WiFi	Light Bulb	com.hama.smart
Philips Hue	Light Bulb	com.signify.hue.blue
Ikea DIRIGERA	Smart Hub	com.ikea.tradfri.lighting
Anti-Lost	Smart Tracker	com.lenzetech.kindelf
LIFX A60	Light Bulb	com.lifx.lifx
Nut Find3	Smart Tracker	com.nut.blehunter
Soundcore Life Q35	Headphones	com.oceanwing.soundcore
Wiz Colour	Light Bulb	com.tao.wiz

This is expected because of how modern ads are targeted and auctioned, requiring dynamic information. IoTFlow only recovers the entry point for ads, but this is actually sufficient to determine that they are used. It also highlights an important issue: Considering all domains gives a false sense of accuracy toward dynamic analysis, many of which may not provide new insight. For example, while it confirms our findings of extensive tracking in IoT apps, the IKEA companion app contacts 48 domains in total, but it also contacted 20 advertising domains and four social network domains.

Focusing on *certainly* IoT domains, IoTFlow and dynamic analysis share 21 domains across all apps (min 0, avg 1.62, max 5), IoTFlow identified 33 domains that dynamic analysis missed (0/2.54/10), and dynamic analysis found 19 unique domains (0/1.46/4). That is, IoTFlow performs better than or equal to dynamic analysis for 9/13 devices and worse for only 4/13 devices (Fitbit smart watch, Hama light bulb, Soundcore headphones, and Wiz light bulb). For 2/4 of these apps, Fitbit and Wiz, IoTFlow correctly identifies the effective TLD of all domains we observed dynamically, that is, the operator, but it missed some subdomains. For Hama, it misses four IoT endpoints that we saw dynamically, likely because the device is a rebranded IoT device. For Soundcore, it misses one dynamically generated domain pointing to the device's most recent firmware.

Beyond domains, we also compare requests' paths. It allows us to assess which approach is more promising to comprehensively understand IoT device behavior, meaning if one discovers more IoT-related functionality or if they identify distinct (overlapping) sets of behavior. Both approaches identified the same 50 IoT-related paths over all 13 apps (min 0, avg 3.85, max 17). We statically identified an additional 231 IoT-related paths (min 0, avg 17.77, max 45) and 496 general paths (min 2, avg 38.15, max 77). Dynamic analysis found 110 unique IoT-related paths (min 0, avg 8.46, max 32) and 337 general paths (min 1, avg 25.92, max 54). For three apps (Fitbit, Hue, and Wiz), our static

analysis performs worse. Fitbit and Wiz use annotations to construct paths, which we cannot analyze, a limitation we share with state of the art (see Section 2.6). For Hue, our approach extracts 2 IoT-related path, while we observe 3 paths dynamically. IoTFlow performs better or equal for 10/13 apps, with a factor of at least 1.14x (Divoom, 41 vs. 36) and up to 31x (Flowercare, 31 vs. 1). For the IKEA app, dynamic analysis did not find any IoT-related paths, while IoTFlow found 45 paths.

Overall, IoTFlow performs better than dynamic analysis and extracts more IoT-related behavior statically from companion apps than dynamic analysis (54 domains and 281 paths vs. 40 domains and 160 paths) for most apps (9/13), it performs comparable for one app, and it performs slightly worse for the remaining apps (3/13).

IoTFlow Findings. Taking an in-depth look into IoTFlow's security and privacy findings for the 13 apps, we find that:

- 8/13 apps send information via unencrypted HTTP to third parties, which an attacker could eavesdrop on or modify (e.g., if they are on the network path or the same wireless network). If unencrypted data is used to configure or update the device, then taking over control could be possible [216]. The NUT Find3 item tracker retrieves notifications over unencrypted HTTP, allowing an attacker to modify a user's notification (e.g., to show that a lost item was found and where or that it has moved away).
- 5/13 apps use hard-coded symmetric encryption keys (e.g., for AES), which allows attackers to eavesdrop on their communication and can allow them to impersonate the remote end (e.g., to push configurations or updates by extracting the keys from the companion app) [210, 219].
- 2/13 apps send the hardware identifiers (IMEI) to countries outside of the EU, that is, outside of the GDPR region (one might send it to Russia and one to China to a remote endpoint that indicates tracking), using an API that is deprecated (see Section 2.4.5).
- 5/13 apps, while less critical, send country location information to remote endpoints.
- No apps use hard-coded authentication credentials, but this does not imply that they are secure because they might not use any authentication at all.

2.6 Limitations and Future Work

IoTFlow has limitations inherent to static analyses. Additionally, we utilize the existing frameworks Soot and FlowDroid, inheriting their limitations. For example, our resilience to obfuscation is limited, which can affect signature-based identification of sources and sinks. We find that they are only a minor share for companion apps (2.66% obfuscated). However, they are more prevalent in general-purpose apps (10.81% obfuscated) based on an APKiD [99] analysis of our datasets. Obfuscation is also an orthogonal problem, and new deobfuscation techniques can readily be adopted. Similarly, we focus our analysis on

the Dalvik bytecode of apps; that is, we do not support native code. We currently do not consider code annotations, which the *retrofit* library uses to specify request paths and network methods. Both techniques are infrequently used, and not tackling them is a limitation we share with prior work, as existing frameworks struggle to support them, and the required engineering effort to support them is substantial.

Our DFA supports ICC, but our VSA does not. We plan to extend ICC support to VSA in future work. Only 2.01% of reconstructed values contain ICC data, which does not invalidate our results. Currently, we limit ICC tracking to the same app, but theoretically, ICC can cross app boundaries or come from websites via deep links, providing further avenues for collusion.

For our analysis, we limit the number of backward steps and set a timeout, which trades between precision and resources but could lead to missing values and flows. We empirically determined our thresholds and other limits could yield more precise results.

Motivated by our results on certificate pinning and abandoned domains, we aim to study how companion apps evolve over time. Naturally, identifying network endpoints, protocols, and APIs is only the first step toward truly understanding the security and privacy of device-to-cloud communication in the IoT ecosystem.

2.7 Related Work

IoT Security Prior work in IoT security largely focused on identifying attacks on a small set of devices. Wood et al. [324] investigated how medical IoT devices communicate and transmit data, and they found them leaking information, like the measurement frequency, despite using encryption. Chu et al. [63] discovered kids' devices sending PII over unencrypted connections. Other work [29, 55, 100, 203, 314] focused on Samsung's SmartThings apps. SmartThings is a smart hub ecosystem unifying the control of compatible devices and allows event flow graphs, which are conceptually simple [202]. In contrast, IoTFlow analyzes arbitrary Android apps, which are more widespread and significantly more complex. Correspondingly, their techniques do not readily transfer to the entire IoT ecosystem. Related work also investigated the ecosystem via crowd-sourced network traffic collection [133] or telemetry data [158] of real-world user devices, which raises ethical and anonymization challenges.

IoT Companion App Analysis Different work investigated IoT companion apps *in combination with* physical devices [58, 77, 247, 253] to find security and privacy issues. For example, Zhou et al. [331] studied the interactions between IoT devices, cloud, and apps using state machines and found issues that can lead to device hijacking. However, they require the IoT devices, which prevents scalability. We overcome this limitation with our new static analysis approach design. Wang et al. [315] analyzed companion apps without the corresponding device. Instead of analyzing and determining *how* and *with whom* the apps communicate, as we do, they focused on identifying re-branding and propagation of known vulnerabilities. That is, they require prior domain knowledge about

other devices and existing vulnerabilities. Similarly, Jin et al. [149] aimed to identify companion apps at scale, and then to identify known vulnerabilities in the apps, such as outdated library versions. Nan et al. [197] analyzed IoT apps with machine learning to detect code that handles IoT-related data, and then assessed whether the behavior was communicated to the user. Naturally, their statistical machine learning approach fundamentally differs from our static program analysis approach. Other work [277, 278, 330, 333] aims to find BLE issues in mobile apps. IoTFlow is more general as we investigate communication beyond BLE, thus obtaining a better understanding of a greater part of the IoT ecosystem.

General-purpose App Analysis Understanding general-purpose mobile apps has seen significant work. Some approaches use dynamic analysis to run apps in controlled environments to observe their (network) behavior and endpoints [67, 170, 171, 238, 246, 254]. As we observed, the provided inputs impact the analysis, which is an ongoing research challenge [53, 62, 123]. Moreover, to adopt these approaches, one would need the actual IoT devices, making large-scale analysis infeasible. Several approaches extract information about apps through static analysis techniques, like VSA or flow analysis, such as network endpoints, API keys, protocol commands, etc. Gadient et al. [107] extract URLs and JSON schemas to study HTTP(S) usage, private APIs, and code injection vulnerabilities. Extractocol [153] reconstructs HTTP requests based on data flow analysis for automated protocol analysis, but does not scale. Stringoid [243] simulates string concatenations, but cannot reconstruct URLs built in other ways, such as *okhttp3.HttpUrl.Builder*. Leakscope [332] reconstructs API keys in mobile apps. Zuo et al. [333] reconstructed BLE UUIDs to identify vulnerable implementations of its pairing process. Wen et al. [317] reconstructed Controller Area Network (CAN) bus commands.

2.8 Conclusion

We introduced IoTFlow, a new technique for the large-scale security and privacy analysis of IoT devices through their companion apps. With Value Set Analysis (VSA), we extract network endpoints and protocols, which enables us to characterize IoT device behavior for local app-to-device communication and remote communication with cloud backends without requiring the physical IoT device. By cleverly combining VSA with Data-flow Analysis (DFA), we trace data flows from IoT devices and sensitive Android methods to understand better what data companion apps share and how. Leveraging IoTFlow, we analyzed 9,889 companion apps and 947 general-purpose apps. We identified striking differences between the two types of apps and discovered various security and privacy problems in the IoT ecosystem, such as abandoned domains, hard-coded credentials, expired certificates, or use of broken encryption algorithms. Our approach shows clear promise for identifying security and privacy issues of IoT devices at scale and it could be used to generate privacy labels or verify claimed behavior automatically.

CHAPTER 3

Large-Scale Security Analysis of Real-World Backend Deployments Speaking IoT-Focused Protocols

Internet-of-Things (IoT) devices, ranging from smart home assistants to health devices, are pervasive: forecasts estimate their number to reach 29 billion by 2030. Understanding the security of their machine-to-machine communication is crucial. Prior work focused on identifying devices' vulnerabilities or proposed protocol-specific solutions. Instead, we investigate the security of backends speaking IoT protocols, that is, the backbone of the IoT ecosystem.

We focus on three real-world protocols for our large-scale analysis: MQTT, CoAP, and XMPP. We gather a dataset of over 337,000 backends, augment it with geographical and provider data, and perform non-invasive active measurements to investigate three major security threats: information leakage, weak authentication, and denial-of-service. Our results provide quantitative evidence of a problematic immaturity in the IoT ecosystem. Among other issues, we find that 9.44% backends expose information, 30.38% CoAP-speaking backends are vulnerable to denial-of-service attacks, and 99.84% of MQTT- and XMPP-speaking backends use insecure transport protocols (only 0.16% adopt TLS, of which 70.93% adopt a vulnerable version).

3.1 Introduction

The number of active Internet of Things (IoT) devices is forecasted to reach more than 29 billion by 2030 [298]. They assist people with health sensors and improve living conditions in smart homes (e.g., alarms and thermostats) and smart cities (e.g., air quality monitors). These devices typically rely on backends, that is, *servers, commonly deployed in the cloud,*

that store and process data as well as control the connected devices. Backends thus play a vital role in the IoT ecosystem, and their security is crucial. Vulnerable backends enable a variety of attacks, such as information exfiltration or denial-of-service (DoS). Such attacks are far from hypothetical and unprecedented in their scale due to IoT devices' pervasiveness [116, 270]. Sabetan [260] discovered that over 40,000 Nexx's Smart Garage doors were vulnerable due to a misconfigured backend. An attacker could have opened any garage door from anywhere in the world as a single insecure password was used to protect data for all customers. Several issues also arose with connected kids' devices and their insecure backends [134, 135, 187]. CloudPets allowed parents and kids to record and receive audio files through Internet-connected plush toys. However, over 2M recordings of over 800k users containing personal conversations were publicly accessible as they were stored in a MongoDB database with hardly any authentication [136, 191].

There are only few standardized attempts for securing the IoT ecosystem, such as Manufacturer Usage Descriptions (MUDs) [83, 91, 162], but they are only high-level descriptions and not yet deployed [143, 195]. Incorrect and poor documentation also leads to vulnerabilities. Jia et al. [148] showed how 26 out of the 38 "best practices" in the official developer guide of Amazon Web Services (AWS), the leading IoT cloud platform, introduced vulnerabilities.

Albeit security is a primary concern for developers [85, 281], the vast amount of communication protocols and the heterogeneity of IoT environments make it difficult for them to fully comprehend the overall situation. The many protocol standards and the plethora of IoT devices, from pacemakers to smart refrigerators, further complicate the situation. Each device and deployment has different requirements and resource constraints, making developing and enforcing security and privacy measures challenging.

Previous studies focused on identifying device-based vulnerabilities or proposed protocol-specific solutions. In addition to extensive studies on devices' susceptibility to malware [5, 10, 76], some approaches leveraged companion apps to improve the scalability of analyses of the IoT ecosystem, but, again, mainly focused on device security [58, 247, 269]. There is a need for comprehensive studies assessing the security of IoT backends as they represent an easy entry point for attackers and allow the escalation of attacks to any devices connecting to them [325]. Backends are the backbone of the IoT ecosystem and ensuring their security is critical [97].

In this paper, we fill this gap by measuring the security posture of publicly accessible IoT backends, that is, servers speaking IoT-focused protocols, at scale. We investigate three application-layer messaging protocols widely adopted in the IoT: Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and Extensible Messaging and Presence Protocol (XMPP).

Maggi et al. [176] and Palmieri et al. [228] provided first insights into the security of MQTT and CoAP backends, identifying exposure of sensitive information, for example, ambulances leaking their geographical locations. Our work complements and expands on their work by scrutinizing more and different classes of vulnerabilities, such as DoS, and

characterizing the security posture of real-world IoT backend deployments at scale.

Note that we do not investigate HTTP backends because determining if it is used in the context of the IoT is extremely challenging as it requires a semantic understanding of the exposed API, demanding analyses that do not scale and require invasive and ethically questionable probing. These analyses would have to be performed over large parts of the Internet due to the popularity of HTTP when only a small to negligible number of HTTP backends are involved in the IoT. Generally, HTTP has shifted from an application-layer protocol for the web to a common transport protocol for many applications (e.g., DNS-over-HTTPS [44, 131], HbbTV [299, 300]). Thus, we focus on protocols that devices themselves "speak" and that are tailored for the IoT.

We leverage Shodan to crawl public backends since existing datasets contain insufficient IoT backends, preventing us from painting a complete picture of the ecosystem. We can gather 223 backends from prior datasets speaking the selected IoT protocols, while we collect and analyze 337,000 backends. We infer the deployed software versions, list their exposed topics and resources, and test if security and privacy measures are in place, e.g., authentication or Transport Layer Security (TLS). We discover thousands of vulnerable backends and cases of sensitive data exposure. Furthermore, we repeat our evaluation over time to provide a longitudinal view of the security and privacy of IoT backends.

Overall, we make the following contributions:

- We gather a dataset of 337,464 backends speaking MQTT, CoAP, or XMPP, and record 10.6GB of network traffic.
- We evaluate backends' security and privacy posture at scale, studying misconfigurations and vulnerabilities. We discover critical security issues, like weak authentication and the potential for amplification attacks that can enable DDoS.
- We investigate TLS adoption, analyzing their use of insecure versions, cryptography, and expired certificates.
- We report our findings via a Coordinated Vulnerability Disclosure (CVD) process to the backend operators and provide guidance to support their remediation efforts.
- We repeat our analysis and show that, despite improvements and our disclosure, some backends exhibit worse security and are affected by more vulnerabilities.

Ethical Considerations and Disclosure. Naturally, our measurement prompts ethics questions. We describe our precautions to prevent potential harms in Section 3.7.1. Our institution's ERB has reviewed and approved our study. We discuss our CVD process with the Dutch CERT and Cyber Security Center in Section 3.7.2.

Artifacts. Due to the sensitive nature of the data, we make our datasets available to other researchers on request. Our source code is available at https://github.com/SecPriv/IoTBackends.

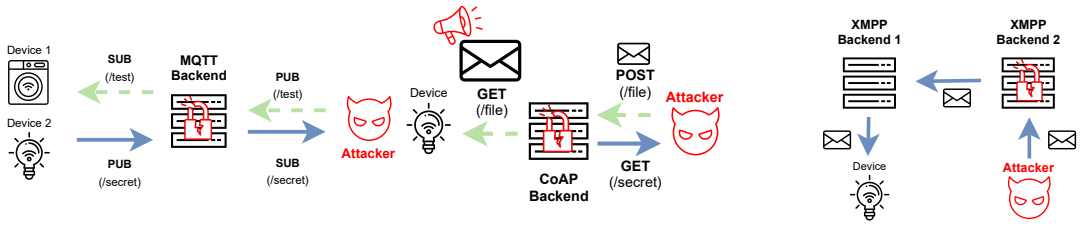
3.2 Background

IoT messaging protocols offer tailored functionalities that account for the characteristics of the devices, their resources, and network constraints. This makes them particularly interesting from a security perspective, as these trade-offs impact confidentiality, integrity, and availability. Following, we introduce the most widely adopted IoT protocols. We exclude HTTP from our study because discerning whether an HTTP backend serves (only) IoT content does not scale and requires invasive and ethically questionable analyses, while HTTP is also mostly spoken by companion apps and mostly used as a transport protocol for other protocols on top of it [269].

Message Queue Telemetry Transport (MQTT). MQTT is a lightweight publish/-subscribe IoT messaging protocol standardized by OASIS [34, 35]. Its operation revolves around three entities: the broker, that is, the *backend*, and one or more publisher and subscriber clients, for example, IoT devices. The broker is a centralized entity that receives PUBLISH messages. It routes them based on subscriptions, access control rules, and a Quality of Service (QoS) that can be both associated with the sender or the receivers (i.e., 0 = at most once, 1 = at least once, 2 = exactly once). Publishers typically open a connection with the broker via a CONNECT packet and send one or more messages on specific topics by indicating a QoS. Topics are hierarchically organized paths and can include wildcards ("+" and "#" respectively). Subscribers subscribe to one or more topics via SUBSCRIBE packets.

Constrained Application Protocol (CoAP). CoAP is a REST-oriented IoT messaging protocol focusing on resource-constrained Machine-2-Machine (M2M) communication [47]. A client requests a resource from a server, that is, a *backend*, via its URI (Uniform Resource Identifier). The request can be a confirmable (CON) or non-confirmable message (NON). The protocol is asynchronous and relies on UDP (User Datagram Protocol) as the transport, which is connection-less and does not provide a re-transmission mechanism. CON packets guarantee a certain reliability. CoAP methods mirror HTTP methods: GET to fetch a resource, POST, PUT, and DELETE to create, update and delete it. CoAP can be used with Datagram TLS (DTLS) to improve security, known as *coaps*, and it supports four security modes: No Security, Pre-Shared Key, Raw Public Key, and Certificates. Their choice is dictated by resource availability, security requirements, and deployment (e.g., Internet access).

Extensible Messaging and Presence Protocol (XMPP). XMPP is an application profile of the Extensible Markup Language (XML). It enables the near-real-time exchange of structured and extensible data between two or more network entities [262, 264], building on a distributed client/server model to exchange "stanzas" (XML data). XMPP relies on TCP and open XML streams to exchange stanzas. It allows client-to-server and server-to-server communication. In the latter, one server acts as a client with the difference that its addresses are known a priori. We consider XMPP servers as *backends*.



(a) MQTT (b) CoAP (c) XMPP

Figure 3.1: Attack Vectors for the Studied IoT Protocols. We consider three different architectures: Publish-Subscribe (MQTT), Client-Backend (CoAP), and Distributed Client-Backend (XMPP). The colored arrows signify different flows.

3.2.1 Security Concerns with IoT Protocols

Threats per Protocol

We focus on distinct threats for MQTT, CoAP, and XMPP, shown in Figure 3.1. We consider other attacks, like breaking cryptographic ciphers, out of scope.

MQTT. Authentication represents a problematic issue for MQTT. Enforcing authentication via credentials or certificates is possible but not required. Authorization-wise, only simple Access Control Lists (ACLs) are supported. Data integrity can be enforced via TLS. Authentication is not used by default, even when brokers support password-based authentication. Attackers can sniff credentials by intercepting CONNECT packets as they are sent in plaintext (no TLS). Successfully connecting to a broker puts confidentiality at risk. Attackers may also be able to subscribe to all topics with wildcards and potentially access sensitive information.

Data integrity is at risk if attackers can intercept messages, replay them, or alter their payloads. They could perform confused deputy attacks by modifying firmware update message files or references.

CoAP. Due to UDP's nature, CoAP is vulnerable to IP address spoofing attacks. UDP cannot authenticate a communication partner; endpoints cannot verify if a packet truly originated from the claimed source IP address. An attacker can send a request with a spoofed IP address, and the backend, since it might trust the source address, might process the request and respond to the spoofed IP address. The server then acts as a reflector [176]. CoAP is susceptible to IP spoofing only when not adopting adequate authentication. For example, DTLS enables verification of communication parties.

CoAP can also enable amplification attacks. As it uses the request/response model, the server responds when receiving a request. However, the response size can be substantially larger than the request. Adversaries can send many small request packets to generate large response packets. In turn, amplification combined with IP address spoofing allows attackers to launch DoS attacks at victims.

XMPP. XMPP uses TLS with the STARTTLS extension for session encryption, protecting communication from eavesdropping and tampering and allowing to upgrade existing insecure connections to secure TLS ones. However, a downside of STARTTLS is that it makes XMPP vulnerable to downgrade attacks, enabling Monster-in-the-Middle (MITM) attackers to read and modify XML stanzas [208].

Other issues lie in how encryption is used. A stanza can be sent over multiple XML streams, but there is no guarantee that all streams are encrypted. Therefore, end-to-end encryption is vital to protect the stanza on every hop, but the XMPP community does not yet provide a suitable technology [263].

3.3 Threat Model

Attackers have the following capabilities: They can connect to the vulnerable backends, which is possible in our case because all backends are publicly reachable on the Internet. Without a valid connection, the attacker would be unable to connect to them and exploit the vulnerabilities. They have the required knowledge about the protocols and vulnerabilities to successfully exploit them. For example, some weaknesses require specific message formats or payloads, such as, allowing "#" for listing all MQTT topics.

Information Leakage. IoT backends can unveil different types of information, ranging from software information (e.g., library versions) to exchanged messages. In the worst case, an unsecured MQTT broker can expose health monitors' (e.g., insulin pumps) data and patients' Personally Identifiable Information (PII) [176]. Attackers can also leverage other types of information to gain further access: they can exploit known vulnerable software versions (e.g., leading to crashes or taking over control) or target individuals. This threat class is connected to *Weak Authentication*, as an attacker who bypasses authentication can often access unauthorized data.

Weak Authentication. Weak authentication mechanisms are a known problem of the IoT. The resource-constrained nature of IoT devices makes adopting security features costly, and often, developers rely on security-by-obscurity, assuming the non-triviality of reversing devices' firmware [98]. Even when security best practices are adopted, such as TLS, they are often incorrectly implemented. Paracha et al. showed how most devices they tested use old or insecure protocol versions and cipher suites, and lack certificate validation [230]. Bypassing authentication can also allow attackers to gain full control of a system, allowing them to gain access to (sensitive) data, send crafted messages to clients, or spam fake data.

Denial of Service (DoS). The problem of DoS attacks is two-fold: (1) an attacker can target an IoT backend, or (2) the IoT devices. In the former case, a malicious actor can impede communication between clients by taking down the backend. In the latter case, the clients would become unresponsive or crash; thus, it would not be able to perform its task. Considering the potentially critical settings of some IoT deployments, like power plants, such attacks could lead to power blackouts in a geographical area. Moreover,

backends can act as amplifying reflectors for distributed DoS (DDoS) attacks when their response is larger than the request size. Given the limited resources of IoT devices, even a moderate amplification factor can overwhelm devices. If the victim device is medical (e.g., an insulin pump), then a DoS can cause its users serious life-threatening harm.

3.4 Motivation

We discuss two motivating examples that highlight the importance of our study to assess the (in)security of IoT backends. We show how weak authentication and information leakage can pose serious threats to users' security and privacy, as introduced in Section 3.3.

Methodology. Existing research used static analysis for IoT devices' companion apps to spot vulnerabilities in the devices and to extract their backends [148, 149, 150, 197, 201, 269]. We use IotFlow [269] to statically reconstruct the backends that IoT companion apps contact and investigate two example apps with insecure backends based on the associated devices and the data they exchange.

Heart & Lung Monitor. The first example is an MQTT broker for a wearable smart device that monitors a user's lungs and heart. We connect to the broker and subscribe to the wildcard topics "#" and "$SYS/*" with QoS 0 to avoid acknowledging messages intended for other connected clients and listen only passively. We retain only topic names and "$SYS/*" payloads and do not record any other information. We find that messages reveal PII of users, such as name, age, and gender, in addition to several health indicators, such as heart and breathing rate, and precise geographical location. An attacker listening to incoming messages can not only precisely identify users and their geolocation, but also alter health indicators. Furthermore, from "$SYS/*" payloads, we can infer the mosquitto library, version 1.4.15, released in February 2018. This version suffers from authorization and DoS vulnerabilities [212, 213, 214].

Smart Car Dongle. We reconstructed the second backend from an app associated with a dongle to bring smart car features to regular cars. Its functionalities include real-time geolocation monitoring, engine monitoring for anomalies, and anti-theft alarms. When connecting to the broker, we discovered that messages reveal sensitive information about the cars' brand and type, location, fuel consumption statistics, and speed. The broker also unveils users' email addresses and the anti-theft alarm's status (i.e., On/Off). The anti-theft status combined with cars' type and precise geolocation makes a perfect list of valuable cars for thieves to target.

Summary. To illustrate the sensitive nature of user data backends can expose we investigated two MQTT backends extracted from mobile companion apps.

Listing 3.1 shows an example messages an attacker could read from the MQTT backend of a *Heart & Lung Health Monitoring Device*, exposing a users age, gender, location, and health indicators. Similarly, Listing 3.2 shows an example message obtained from the

```
"topic": "-/-/live-broadcast",
"payload": {
    "gender": "-",
    "age": -1,
    "time_zone_utc_offset":-1,
    "altitude": -1,
    "longitude": -1,
    "latitude": -1,
    "type": "",
    "userName": "User Name",
    "shock": -1,
    "breathingRate": -1,
    "uniqueID": -1,
    "strain": -1,
    "hearRate": -1,
    "cadence": -1,
    "distance": -1,
    "pace": -1,
    "userID": -1
}
```

Listing 3.1: Example MQTT message and topic from the *Heart & Lung* monitor.

```
"topic": "-/-",
"payload": {
    "speed": -1,
    "rpm": -1,
    "coordinates": [-1, -1],
    "distance": -1,
    "coolant": -1,
    "voltage": -1,
    "trip_time": -1,
    "status": "-",
    "fuel_consumption_1": -1,
    "gps_satellite_count": -1,
    "gsm_signal_quality": -1,
    "load": -1,
    "IMAP": -1,
    "IAT": -1,
    "air_flow": -1,
    "fuel_ratio_coefficient": -1,
    "direction": -1
}
```

Listing 3.2: Example MQTT message and topic from the *Smart Car* dongle.

MQTT backend of a *Smart Car Dongle*, exposing the car's location, engine statistics, and status of the anti-theft alarm.

For both examples we omit any information that could identify the user or the company. The developer of the health monitor has started remediation procedures following our disclosure. The developer of the car dongle has not responded to our disclosure.

In both cases, weak authentication, or rather a lack of authentication, allows arbitrary anonymous users to connect and read messages. The exposed data we identified is clearly sensitive, highlighting the severity of the problem and potential consequences. We first responsibly disclosed the issues to the developers in May 2023. The developers of the health monitor app replied to our second email. They have deprecated the identified backend and are moving their services to an AWS-managed backend. The legacy backend remains available for backward compatibility until the remaining users update the app. Unfortunately, we have not received any response from the car dongle manufacturer, even after repeated follow-up emails.

3.5 IoT Backend Datasets

The two motivating examples already show how misconfigured and vulnerable backends can impact users of IoT devices. There is a clear need for a comprehensive analysis of publicly accessible IoT backends at scale to identify and understand ecosystem problems and propose informed and viable solutions.

Table 3.1: Datasets Used in Our Study. We report the number of unique IoT devices used to capture the network traffic, the number of unique backends (based on IP or domain), and the number of IoT-specific backends (MQTT, CoAP, XMPP). We also collected a dataset from Shodan, thus, only the IoT backends are available.

Dataset	Where?	When?	# IoT Devices	Unique Backends All	Unique Backends # IoT (%)
IoT Sentinel [190]	FI	2016	31	101	0 (0.00%)
UNSW [279]	AU	2016	28	9,610	125 (1.30%)
IoTLS [230]	US	2018–2020	40	1,495	68 (4.55%)
YourThings [11]	US	Q1 2018	45	7,172	32 (0.45%)
Mon(IoT)r [253]	US+UK	Q1 2019	81	3,570	17 (0.48%)
IoTFinder [234]	US	09/2019	53	7	0 (0.00%)
PingPong [303]	US	11/2019	19	6,848	25 (0.36%)
HomeSnitch [222]	US	Q1 2020	24	1,436	57 (3.97%)
Edge IIoT [102]	DZ	01/2022	>10	38	0 (0.00%)
SHODAN-22	-	07/2022	-	-	901,295 (-)

To collect a comprehensive dataset of real-world backends speaking IoT protocols, we investigate datasets from prior work and collect our own. Table 3.1 provides an overview.

Existing IoT Traffic Datasets. We collect nine IoT traffic datasets from prior work. We extract IPs, associated ports, and DNS information from traffic dumps (pcaps) and map IPs to domains. IP addresses can vary over time, but this is less likely for domain names, which means they can yield more accurate results. When we cannot match IP and domain, we retrieve the (historic) reverse DNS names for the IP via Shodan [276]. We acknowledge that Shodan's databases may be incomplete or outdated, which is a known limitation of it and related approaches [326]. For our analysis, we consider unique backends for which the associated port is a default IoT protocol port, namely 1883 and 8883 for MQTT, 5683 and 5684 for CoAP(s), and 5222, 5269, 5280, and 5298 for XMPP. Overall, the datasets contain only 45 MQTT, 3 CoAP, and 175 XMPP unique backends. The datasets contain little network traffic for the three IoT protocols, highlighting their partial and limited coverage of the IoT ecosystem.

Note: We tried to include the dataset by Saidi et al. [261], but they could not share it because it contains proprietary data from Farsight. This makes it impossible for us to reproduce, validate, or extend their work.

Shodan Crawl (SHODAN-22). Given the limited coverage of the IoT backend ecosystem of previous datasets, which do not allow a large-scale analysis of backends, we crawl Shodan [276] for Internet-connected devices. We search Shodan in the last week of July 2022 for the keywords *mqtt, coap(s), and xmpp*, and find 425,571 MQTT, 474,878 CoAP, 4 CoAPs, and 702 XMPP results, without restricting us to standard ports. We store IPs, available hostnames, and ports and add extra information, such as connection codes.

3.5.1 Dataset Augmentation

We augmented our dataset with additional information, namely the geographical location and whether backends are hosted on widely adopted cloud platforms. We use Shodan and `WHOIS` to determine the country of the backend. We analyze backends at a country-level granularity, and our geolocalization is sufficiently accurate, as country misclassification is rare. We also gathered the IP address ranges for ten major cloud providers that manage IoT services: Amazon Web Services (AWS), Google, Cloudflare, Microsoft Azure, Alibaba, Oracle, IBM, DigitalOcean, Yandex, and Salesforce [7, 12, 65, 81, 114, 138, 189, 225, 267, 327]. We determined if backends' IP address belonged to one of the providers. If it does not belong to one, we classify its provider as *Other*. IP address ranges of cloud providers might change over time, which is why we include them in our artifact. Finally, we use the regular expression by Saidi et al. [261] to distinguish between self-hosted and managed AWS backends (i.e., hosted on the AWS cloud vs. managed by AWS).

Countries. Most SHODAN-22 IoT backends are located in South Korea, followed by China and the Philippines. Interestingly, breaking this down at the protocol level, most MQTT backends are located in South Korea (276,100, 64.88%), followed by China (46,391, 10.90%) and Japan (18,204, 4.28%). For CoAP, most backends are in the Philippines (167,849, 35.35%), followed by Russia (104,639, 22.03%) and China (80,619, 16.98%). We explain geolocation trends based on where most manufacturers and vendors are located. Shadowserver observed a similar trend for CoAP backends [152, 272]: While they only probe for services exposing the resource *.well-known/core*, they saw the same distribution, with the Philippines, China, and Russia making up nearly 93%. Maggi et al.'s study [176] showed a different country distribution for MQTT and CoAP backends in 2018. They identified numerous backends in the US. One reason could be dynamic changes in IoT backends' locations over time, aligning with major vendors' (re)locations, or increased white-labeling of IoT devices. Another reason could be that some backends are not IoT but have adopted IoT protocols (e.g., due to similar resource constraints). Given ethical considerations, a precise distinction is impossible, as it would require invasive measurements, making our data an upper bound (see Section 3.8). Moreover, other backends using IoT protocols might suffer from the same issues as IoT backends.

Providers. We identified the hosting providers for 31,785 backends, of which 25,146 (79.00%) belong to AWS, 3,381 (10.62%) to Azure, 2,292 (7.20%) to Google, 674 (2.12%) to Oracle, 282 (0.89%) to Alibaba and the remaining ten 10 (0.03%) to Cloudflare. Our trend reflects the market share ranking for cloud providers in 2022, with AWS, Azure, and Google being the top three providers [151]. Particularly interesting is that AWS hosted almost 80% of backends with 34% market share, while Azure and Google account for 11% and 7% backends at 21% and 11% market share, respectively. We found no backends hosted on DigitalOcean, Yandex, IBM, or Salesforce.

3.6 Large-Scale Security Assessment

We describe our methodology for large-scale security measurement and assessment of the identified IoT backends. We discuss the vulnerabilities and weaknesses that we study and explain their relevance. We also expand on how we implement our approach for each messaging protocol. The protocols that we investigate have different architectures and face different threats (discussed in Section 3.2 and illustrated in Figure 3.1): Publish-Subscribe (MQTT), Client-Server (CoAP), and Distributed Client-Server (XMPP). Therefore, they require different measurement approaches, and we describe our methodology and results per protocol. We also investigate TLS usage for the TCP-based protocols MQTT and XMPP. TLS is a widely adopted security measure for TCP-based protocols. Since most CoAP deployments rely on UDP, TLS cannot be used, and DTLS is required. However, DTLS analysis remains an open research area, and only 4 of 474,882 CoAP backends even support DTLS (0.0008%), thus, we leave it for future work.

Considering the low number of IoT backends from existing datasets and that most were unreachable as of September 2022 (93% were unreachable; we could only connect to one MQTT, zero CoAP, and 14 XMPP backends), we exclude them from our analysis for clarity and readability. We attempted to include additional backends by extracting them from companion apps utilizing IotFlow [269], but we could only identify less than 100 backends in over 4,000 manually verified companion apps. Likely, most IoT devices do not rely on the mobile app but communicate with IoT backends directly, which is intuitive, as otherwise, they could stop functioning when the phone is unavailable. Thus, we perform our security analyses on the SHODAN-22 backends.

For our analyses, we act as unauthenticated users without prior knowledge of the target backend. Our measurements do not require privileges (e.g., admin) or authentication. While we can identify vulnerabilities without requiring access privileges, this does not imply that they are always exploitable. We also cannot test for actual exploitability as this would clearly violate ethics and possibly disrupt or compromise services.

Finally, we repeat our analyses after one year to determine if operators improved the security of vulnerable IoT backends over time. We also evaluate the stability of our dataset because IoT traffic datasets tend to age quickly, as discussed in Section 3.6.7.

3.6.1 General Approach

We identified the main security issues in IoT backends by leveraging prior work and following our classification in Section 3.3. Prior work identified insufficient authentication and authorization measures as key issues [97, 199], as well as the adoption of outdated libraries with known vulnerabilities [176], helping us understand how weaknesses in our threat classes can be discovered in the backends. We then listed all potential attacks for each class, such as using default credentials in case of *Weak Authentication*. Particular caution is required because we interact with real-world backends, which might provide critical functionalities. For this reason, after thoroughly studying and evaluating the

possible ethical implications of our study (see Section 3.7.1), we removed all attacks from our list for which existence testing *could* alter the backends' correct functioning. For example, we decided against testing for known/default credentials as (1) it could cause DoS in case requests are too frequent or possibly prevent authentication by a legitimate user because of too many unsuccessful authentication attempts, and (2) it likely would also cross legal boundaries in some jurisdiction, as we would access a system that could be considered legally protected.[1]

Importantly, we test for specifically selected vulnerabilities and threats, meaning our results may not generalize to other vulnerabilities or threats. Nevertheless, our results provide new insight into the security state for IoT backends, as the classes of vulnerabilities we analyze are general and well-known threats. Moreover, our analysis allows us to investigate how the backends' operators consider and approach security. The specific vulnerabilities and threats we analyze differ in how long mitigations and patches have been available or known, how easy they are to deploy, and their severity/impact.

We first collect general information about the backends and identify the libraries they use, including their versions. This information can tell us whether a system under analysis adopts the most recent security patches. In this way, we can determine if backends suffer from known vulnerabilities. When available, we also collect complementary information, like the authentication mechanisms and the number of connected clients. We then investigate possible information leakage from these backends. First, we analyze the communication structures and determine how messages and resources can be retrieved. We then scrutinize data for privacy issues and confidentiality violations. Considering that we act as unauthenticated users, we should not have access to any sensitive information. Figure 3.2 illustrates our complete methodology.

3.6.2 Measurement Setup

We perform our initial analyses on 425,571 MQTT, 474,882 CoAP, and 702 XMPP backends between August and November 2022 from the Netherlands. We perform all tests on a Ubuntu 22.04 virtual machine (VM) with eight CPU cores (Intel Xeon Silver 4110) and 32 GiB memory, with a timeout of 60 seconds per backend. We analyze up to 10 MQTT and CoAP backends in parallel to minimize network input/output wait times. To foster further analysis, we record all our analysis traffic (10.6 GiB) using `tshark`, filtered on the backends' IP, which we make available upon request.

Reproducibility and Tooling. For some analyses, we adopt and adapt existing tools. This is motivated by their stability and reliability, making it easier to reproduce our results. We recommend IoT developers adopt them for vulnerability testing of their deployments,

[1] We initially considered default credentials, and an expert assessment indicated that this would be legal in at least some of our jurisdictions, as protection with default credentials should not be considered adequate protection. However, our research touches many jurisdictions, and the scientific benefits of doing so are limited.

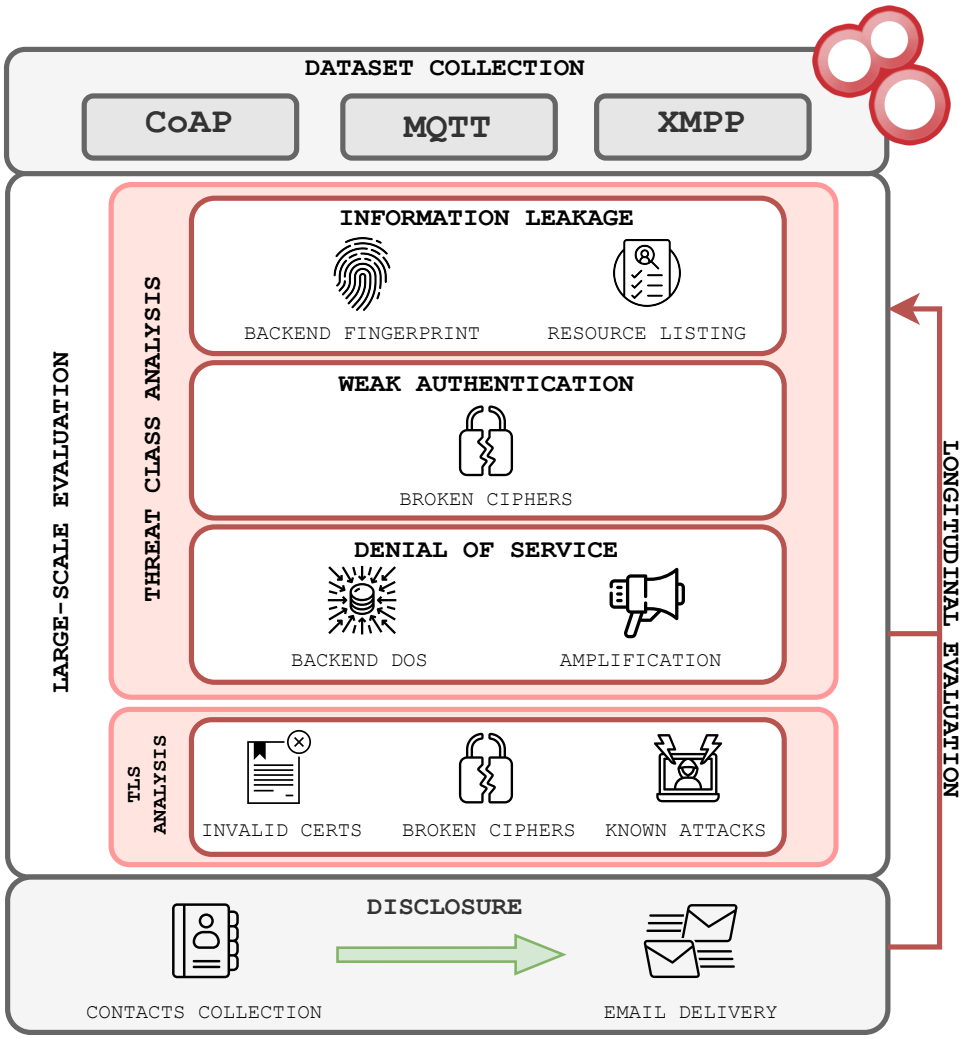


Figure 3.2: Methodology Overview. We first collect our backends' dataset via Shodan. Then, we perform our large-scale evaluation for the identified threat classes. For TCP-based protocols (i.e., MQTT and XMPP), we also evaluate TLS adoption and implementation. We repeat our analysis over time to understand how security posture evolves, and to consider the impact of our coordinated disclosure.

making our results approachable and understandable while fostering scalability. We report the tools and how we utilized them in the approach paragraph for each protocol.

3.6.3 MQTT

Approach. We identify MQTT backends' software version and analyze existing topic names using the *Paho* library (v1.6.1) [168]. We complement the MQTT backends we gathered in SHODAN-22 with connection codes: If the connection is successfully

established, it returns code 0. Otherwise, codes 1 to 5 mean an error related to connection parameters. We define three cases based on the return code: for code 0, we can connect and proceed; for code 1, we try a different protocol version (e.g., MQTTv5); for codes 2 to 4, we mark it as available but requiring authentication or authorization. We connect to them via IP/hostname, port, and protocol version.

After a successful connection, we subscribe to the wildcards "$SYS/#" and "#" with QoS 0 to avoid acknowledging messages intended for other clients. With QoS 0, we do not acknowledge messages, and the broker will re-attempt sending them to other (legitimate) clients until they are acknowledged (by them). We listen for incoming messages for 40 seconds. We record the topic names and count the received messages. We do not record message payloads, except for "$SYS/" topics, which provide relevant information for backend fingerprinting and security analyses, like the broker version or the number of connected clients. While some topic names could include potentially sensitive information, we only record minimal metadata and deem this risk acceptable.

We test for two vulnerabilities via *cotopaxi* (v1.6.0) [268], an IoT pentesting tool by Samsung: CVE-2019-9749 [217] causing the crash of Fluent Bit brokers (version $< 1.0.6$) and CVE-2018-19417 [211], a stack-based buffer overflow in Contiki OS MQTT brokers (version ≤ 4.1), by checking the broker version. The former vulnerability makes the broker unavailable via a DoS attack, thus impeding communication between IoT devices. The latter allows remote code execution on the broker and full memory access.

Results. Overall, we successfully connected to 251,382 out of 425,571 (59.07%) MQTT backends, which we consider the baseline.

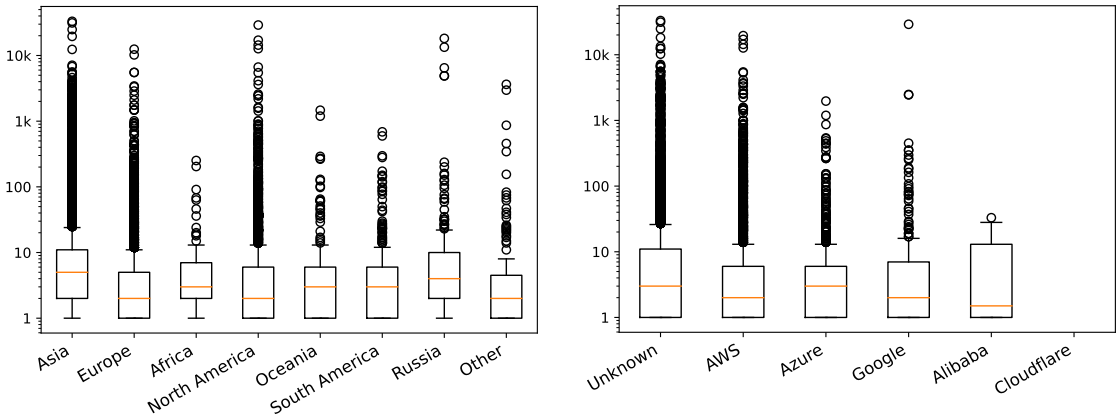
Weak Authentication

We observed 12,071 backends (4.80%) that returned a code 4 upon connection, which indicates that they use username-password-based authentication. These credentials are sent in plaintext, enabling eavesdroppers to intercept them and connect to the MQTT broker if TLS is not used properly.

Information Leakage

Demographics. MQTT allows us to identify the number of unique clients, that is, IoT devices, connected to a broker. Brokers with more clients could unveil more information as more messages might be exchanged, and they can be more impactful targets for DoS attacks. The average number of connected clients is 11.7 (σ=538.04), peaking at 33,134 for a single broker. Per geographical region and provider, the average number of connected clients is less than ten, but with long and dense outlier tails (see Figure 3.3a and Figure 3.3b).

Software. We identify, when possible, the library and its version adopted by the MQTT broker via messages sent on the topic "$SYS/broker/version." If this topic is restricted



(a) Number of connected clients to individual MQTT backends grouped by continents. Asia shows the densest tail, indicating many MQTT backends are located there. Africa, Oceania and Russia show lower densities.

(b) Number of connected clients to individual MQTT backends grouped by providers. Alibaba brokers show fewer connections, but due to small sample size no strong trend can be inferred.

Figure 3.3: Comparison of connected clients to MQTT backends, grouped by (a) continent and (b) provider.

or we cannot connect to the backend, we cannot infer the software version. We analyzed version information for 22,986 backends. After removing artifacts, almost all brokers (22,978, 9.14%) use mosquitto, an open-source MQTT broker by Eclipse [169]. Worryingly, 10,627 backends (4.23%) adopt a library version that was released more than five years ago (version < 1.5). Outdated library versions can indicate poor security practices and allow the exploitation of vulnerabilities addressed in later versions. Updating versions is, however, not always possible, as some devices might not allow updates. Investigating the vulnerabilities that old mosquitto versions suffer from, we find that versions 1.0 to 1.4.15 (10,627 backends, 4.23%) are vulnerable to a null pointer dereference that can cause a broker crash (DoS) [212]. 11,804 backends (4.70%) use versions older than 1.5.5, which exposes them to two vulnerabilities: Malformed data contained in a password file [214] or an empty ACL file [213] allows attackers to circumvent authentication and authorization checks and access all the information exchanged through the broker.

Topics. We perform an in-depth analysis of the topic names we observed to understand what type of data is exchanged. For example, to determine whether it is IoT device data or unrelated. We observed 1,766,804 unique topics (average 39.45 per backend, peak of 36,254 unique topics for one backend) of which 50,167 are "$SYS/" topics. We use the zero-shot classification model by Laurer et al. [161] to analyze if the remaining 1,717,765 topic names fall into nine major IoT-related categories: *health*, *home*, *security*, *update*, *sensor*, *location*, *industry*, *transportation* and *identifier*. The classifier assigns a score for each topic name and category from 0 (no match) to 1 (perfect match). We require a minimum score of 0.85, as recommended by Laurer et al., for a topic to match a category.

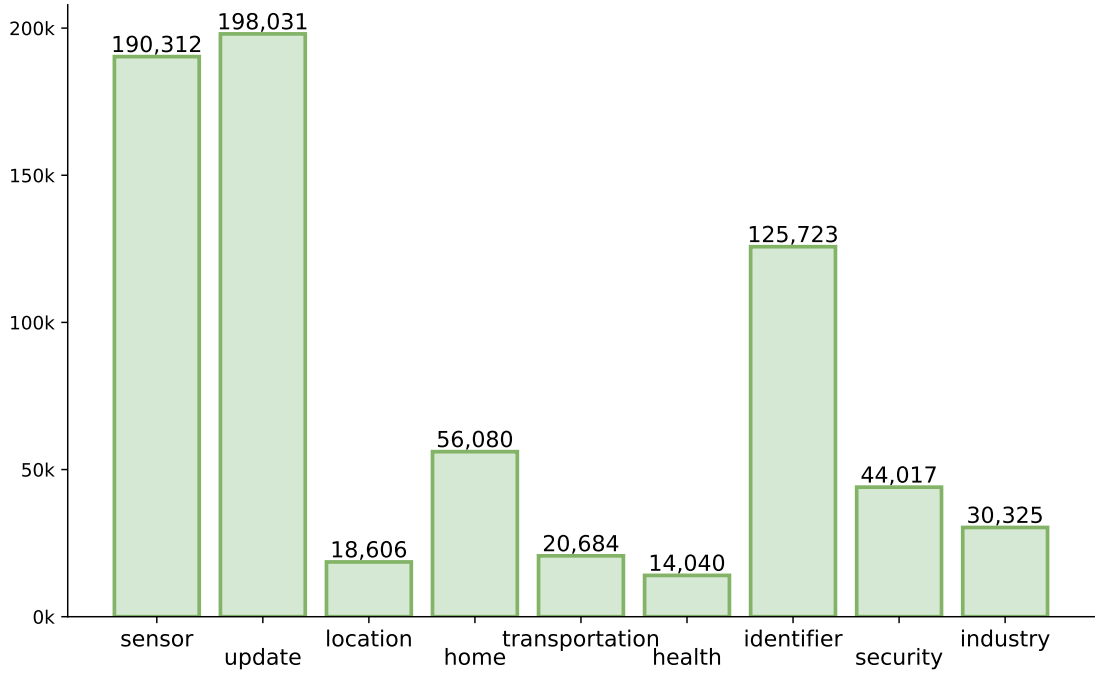


Figure 3.4: Classification Distribution of 697,818 MQTT Topics. The most common topic categories are *sensor* and *update*, with *security* topics also occurring frequently.

Topics may match zero categories (no score ≥ 0.85) or multiple (2 or more scores ≥ 0.85). If a topic matches multiple categories, we count it only for the top-scoring one. Overall, we classify 697,818 topics (40.62%) into the nine IoT-related categories (see Figure 3.4), providing strong evidence that the brokers are indeed used for IoT communication.

Manually analyzing the categorized results, we identified topics that might contain sensitive information that should not be publicly accessible. Following, we redact some of the information to avoid identifiability. For example, *cmnd/-/mqttpassword* appears to unveil the MQTT password, while *Security/GarageDoorFront* and *-/Living_Room/Front_Door_Sensor/-/Home_Security/Cover_status* appear to allow controlling smart home devices. Several other topics are potentially associated with a firmware update functionality. Delivering an update over plaintext channels is a major security concern for smart devices. If the update is not cryptographically signed, then an attacker can replace it with custom malicious firmware [139].

Case Studies. We investigate two specific backends in more depth: (1) the backend for which we collected the most topics and (2) the broker with the most connected clients.

Analyzing the former's topics, it appears to correspond to a power plant in China: Two of its topics are *signal-values/admin/-/_station_efficient/power-facility-value* and *-/station_management/alarmSeverityMap*. Exposing such sensitive information is a clear and severe security threat that could have disastrous consequences, considering how critical the power grid to modern society is.

For the latter, we could not collect any topics. However, it relies on an outdated version of the mosquitto (v1.4.14 from July 2017), which suffers from two authentication and authorization vulnerabilities. Therefore, attackers might be able to exchange messages with many IoT devices (33,134) by exploiting the vulnerabilities and circumventing authorization. This exposes the devices to multiple risks. For example, an attacker could send carefully crafted messages to devices and exploit other vulnerabilities to make them join a botnet, or eavesdrop on their communication.

Denial of Service

Finally, we investigate if backends suffer from two known vulnerabilities that can be used to launch DoS attacks, namely CVE-2019-9749 and CVE-2018-19417. We identified 214 and 196 backends affected by CVE-2019-9749 and CVE-2018-19417, respectively. CVE-2018-19417 allows remote code execution, including letting attackers take the broker offline or disconnect clients, while CVE-2019-9749 crashes the broker directly.

Recommendations: We encourage developers to update their broker libraries, thus fixing known vulnerabilities in older versions. We found vulnerabilities affecting broker versions older than five years, signaling bad security and update practices. In some cases, updating might not be trivial, for example, when IoT devices are running old and broken software versions that are incompatible with backend updates.

We suggest that stakeholders carefully evaluate what information needs to be accessible and by whom and to adopt authentication measures and ACLs. With mosquitto, the most widely used MQTT broker we found, this can be done with a text file that lists each user's permission for specific topics.

Encrypted communication (e.g., TLS) should be used when possible. If the broker does not have to be accessible to the Internet, access should be restricted via a firewall.

3.6.4 CoAP

Approach. We identify the adopted software and the number of connected clients of CoAP backends with *cotopaxi* (v1.6.0) [268]. We compile a list of 30 resources we check for: *helloWorld, test, login, admin, administrator, adm, .passwd, passwd, history, certificates, logout, password, log, logs, about, actions, advanced, auth, backup, .well-known/core, .history, certs, config, configuration, data, dev, files, help, resources,* and *items*. For example, a CoAP backend exposing the `password` resource without protection measures might indicate a sensitive information leak. We also include `.well-known/core`, a default URI used as an entry-point for listing the resources hosted by a backend (but not always available). We perform a HEAD request for each resource with a sleep of one second and look at the return code. We mark the resource as available if we receive the return codes 2.05 (Content) or 2.03 (Valid).

We test for two traffic amplification vulnerabilities with *cotopaxi*, CVE-2019-9750 [218] targeting IoTivity (an open-source framework for device-to-device connectivity) and ZYXEL_000 affecting Zyxel Keenetic routers. For both, we send a message and check the response size. If the amplification factor (AF) is greater than 100%, we flag the backend as vulnerable. Such vulnerabilities allow attackers to abuse CoAP backends as reflectors to take down connected IoT devices. We also test for the DoS vulnerability CVE-2018-12679 [215]. In this case, the target of the DoS attack is the backend itself, which can no longer serve content to its clients.

Results. We successfully connected to 85,957 out of 474,882 (18.10%) CoAP backends, which we consider the baseline.

Information Leakage

Software. We identify the software for 2,864 backends: 1,886 (2.19%) adopt *coap-rs* [71], followed by 932 (1.08%) *FreeCoAP* [72], and 19 *aiocoap* [13] instances.

Exposed Resources. All 30 resources we defined are available. The resource available across most backends is `.well-known/core` (767 backends, 0.89%), followed by `test` (10 backends) and `help` (6 backends). The resource `admin` is publicly accessible for six backends and `password` for seven. Exposing potentially sensitive content without further security measures is a severe security threat. In four cases, the resource `config` returned a code 4.01 (Unauthorized), meaning that some authorization measure is in place. Overall, we observed 841 2.05 (Content), four 4.01, 105,673 4.04 (Not Found), and 24 4.05 (Method Not Allowed) return codes.

One backend exposes all its sensors, actuators, and values/states via a GET request to `.well-known/core`. To avoid negatively impacting the backend, we have not further studied these exposed resources. However, motivated attackers do not restrain themselves and can misuse the exposed resources to gather intelligence and possibly perform remote actions. For example, for a smart building, this setting could allow for understanding if it is occupied via its sensors and opening doors via its actuators.

Denial of Service

We find 25,928 CoAP backends vulnerable to ZYXEL_000 (30.16%) and 25 vulnerable to CVE-2019-9750 that can be abused to launch amplification attacks. We record the AF for all backends, that is, how much larger the response size is than the request. We only consider AFs $\geq$100%. For ZYXEL_000, we find a maximum AF of 849.06% with an average of 240.61% (σ=25.80). For CVE-2019-9750, we find the same maximum value but a higher average of 517.43%, at a larger standard deviation (σ=360.07), signifying that there is a difference in AF across vulnerable backends. The AF for ZYXEL is mainly between 200% and 400%, while the distribution is more sparse for CVE-2019-9750. AFs for other protocols, like the Network Time Protocol (NTP), can be higher, up to 200 times the request size. However, CoAP is a protocol for resource-constrained environments,

in which even a lower factor can cause DoS. Finally, we found 212 backends (0.25%) vulnerable to CVE-2018-12679. If exploited, such vulnerabilities could cause the collapse of the entire backend by a DoS attack, making all operations and resources unavailable.

Recommendations. We suggest developers adopt ACLs to limit access to resources on CoAP backends following the principle of least privilege, i.e., a user should only have access to the resources they need to operate. Further, we deem it important to prevent access to the `.well-known/core` resource, as this would reveal the structure of the backend. Communication should be encrypted when possible (e.g., if connected clients support it). This can be achieved by adopting DTLS. We also encourage updating and applying patches for old vulnerable library versions, as those allow known attacks, such as DoS.

3.6.5 XMPP

Approach. We use *nmap* (v7.80) to gather information about the XMPP backends, including their name, version, authentication mechanisms (e.g., PLAIN, DIGEST-MD5), and TLS support [157]. We also employ the *XMPP Compliance Tester* [117] to try registering the account `user` and to test the backend for a set of compliance requirements, like TLS encryption ciphers etc. [319].

Results. We connected to 125 out of 702 (17.81%) XMPP backends, which we consider the baseline, while 136 (19.37%) backends were unresponsive (filtered/closed port).

Weak Authentication

PLAIN is the most common authentication mechanism (56 backends, 44.80%). According to the AUTH PLAIN specifications, username and password are sent from the client to the backend as a base64-encoded string. Sending credentials in this way, that is, without encryption, is a security threat. Malicious actors passively listening to the communication can decode the credentials and use them to log in. PLAIN, being the top authentication mechanism, shows a widespread insecurity of XMPP backends. We also discover DIGEST-MD5 (36 backends, 28.80%) and CRAM-MD5 (33 backends, 26.40%) as authentication methods. Both methods adopt a client-server challenge-response authentication mechanism. In their response, credentials are hashed using the password as the secret key. Generally, DIGEST-MD5 is more secure than CRAM-MD5 as it prevents chosen plaintext attacks [137]. However, considering current computational capabilities, MD5 collisions are becoming achievable, rendering MD5-based authentication methods obsolete and insecure. Finally, we find nine backends (7.20%) with ANONYMOUS authentication, allowing unauthenticated users to access the backend's content.

With *XMPP Compliance Tester*, we can register a dummy user (with username `user`) with no authentication for four backends (3.20%). Registering a user without requesting a password indicates an insecure implementation of XMPP backends. A malicious actor

can send and receive messages without proper authorization, undermining the system's confidentiality and integrity.

Information Leakage

Demographics. The main languages of XMPP backends are Russian for 68 (54.40%) backends and English for 30 (24.00%) backends, providing some insight into their geographical distribution.

Advertised Features. Among other features, TLS (see Section 3.6.6 for a detailed investigation into TLS) occurs most often (64 backends, 51.20%). Following TLS are *Roster Versioning* (29 backends, 23.20%) [266] and *In-Band Registration* (24 backends, 19.20%) [265]. Rosters are users' contact lists on the backend. *Roster Versioning* saves bandwidth by not sending unmodified rosters.

Recommendations. We encourage developers to drop support for weak and broken authentication mechanisms, like PLAIN, as they undermine the integrity and confidentiality of XMPP backends, potentially allowing attackers to impersonate legitimate users or register new accounts. Moreover, we recommend that operators prohibit user registration without a password.

3.6.6 TLS Adoption for MQTT and XMPP

The IoT ecosystem is highly heterogeneous in systems, devices, and underlying topologies. No standard security solution can be readily applied to all of them. Each scenario requires careful adaption. Nevertheless, TLS provides a fundamental security building block that is widely adopted and supported [49, 230]. Unfortunately, TLS adoption in the IoT domain has been limited because of power/energy concerns [199]. Paracha et al. [230] also found that many IoT devices have wrong TLS configurations, making them vulnerable to attacks. Following, we scrutinize the corresponding backends. We analyze TLS adoption across our TCP-based protocols, MQTT and XMPP. We leverage *testssl* (v3.2rc2) [318] to analyze TLS support and whether cryptographic flaws exist. We extract the supported protocol versions and test for 17 vulnerabilities (e.g., Logjam [207], BEAST [206], SWEET-32 [209]). Additionally, we check whether the certificate has expired or if it was revoked, i.e., if it is in Certificate Authorities (CAs) Certificate Revocation Lists (CRLs).

We test all XMPP backends and sample a random subset of 100,000 MQTT backends (around 25%), which provides statistically significant insight. We automate and parallelize our analysis, using a first timeout of two minutes per backend. We perform our analysis between December 2022 and January 2023.

Table 3.2 summarizes our results. We successfully connected to 54,503 MQTT and 497 XMPP backends via TLS. Dahlmanns et al. [75] also observed a low TLS adoption rate

Table 3.2: TLS-enabled Backends. We report the number of TLS-enabled backends together with the adopted protocol versions (from oldest to most recent) and their vulnerability to attacks.

		MQTT	XMPP	Total
TLS support	Number	73	13	86
	Fraction	0.13%	2.62%	0.16%
TLS version	Version 1	52	7	59
	Version 1.1	54	7	61
	Version 1.2	73	13	86
	Version 1.3	42	5	47
Vulnerabilities	BEAST	50	7	57
	SWEET-32	37	3	40
	Logjam	12	3	15

Table 3.3: Supported Ciphers by Protocol Version. Top three most adopted cipher suites by TLS backends for the analyzed TLS versions. We mark the protocols not recommended by IANA (✖).

	Cipher Suite		No.
v1	RSA_WITH_AES_256_CBC_SHA	✖	58
	ECDHE_RSA_WITH_AES_128_CBC_SHA	✖	58
	RSA_WITH_AES_128_CBC_SHA	✖	58
v1.1	RSA_WITH_AES_256_CBC_SHA	✖	60
	RSA_WITH_AES_128_CBC_SHA	✖	60
	ECDHE_RSA_WITH_AES_256_CBC_SHA	✖	59
v1.2	ECDHE_RSA_WITH_AES_256_GCM_SHA384		83
	RSA_WITH_AES_128_GCM_SHA256	✖	83
	ECDHE_RSA_WITH_AES_128_GCM_SHA256		73
v1.3	AES_256_GCM_SHA384		47
	AES_128_GCM_SHA256		47
	CHACHA20_POLY1305_SHA256		45

in their analysis (around 6%). We find a worryingly low fraction of MQTT backends adopting TLS (0.13%) in our dataset, especially considering how critical some of the exchanged messages are. Still far from Dahlmanns et al. results, 2.61% of XMPP backends adopt TLS.

Protocol Versions and Supported Ciphers. A great share of backends adopts outdated protocol versions, with only 54.65% supporting the latest standard (i.e., only 47 backends support TLS v1.3). Among the TLS-enabled backends, we find that 68.60% support TLS v1 and 70.93% TLS v1.1. The Internet Engineering Task Force (IETF) deprecated both versions in June 2018 based on the severity of discovered cryptographic attacks. Correspondingly, we test whether backends exhibit weaknesses that could be used

to mount attacks. We find that 57 backends (66.28%) are vulnerable to BEAST, which affects TLS versions ≤1 and allows attackers to capture and decrypt sessions, rendering encryption useless. Additionally, 40 backends (46.51%) are vulnerable to SWEET-32, a weakness in block ciphers discovered in 2016. Although the vulnerabilities are exploitable, some preconditions must be met. For SWEET-32, the exploitability depends on whether the affected ciphers are indeed chosen (proposed by the client, picked by the server), and it generally requires a large number of payloads, the threshold of which may or may not be realistic for an IoT device (depending on connection lengths, etc.). Finally, we find 15 backends (17.44%) potentially vulnerable to Logjam, a flaw affecting systems adopting the Diffie-Hellman key exchange with the same prime number, first discovered in 2015. Since then, a 2048-bit shared prime number is considered required.

We additionally analyze the cipher suites the TLS-enabled backends support and report the three most common ones per TLS version (see Table 3.3). Most of them are not recommended by the Internet Assigned Numbers Authority (IANA), that is, they have not been through the consensus process or have limited scope. Some backends also adopt known weak (broken) cryptographic protocols and hash functions. Specifically, two backends adopt RC4 (2.33%), 40 adopt 3DES (46.51%), 75 adopt SHA-1 (87.21%), and two adopt MD5 (2.33%). These algorithms and hash functions have (long) been deprecated because they are vulnerable to attacks.

Certificates. We find two expired certificates. On average, certificates have an expiration date of ~200 days with one extreme outlier of 982 years. When available, we also retrieved the CRLs from the CA for each backend's certificate and determined if the backend's certificate was revoked. We found no revoked certificate.

Additionally, we analyzed whether the hostname and Common Name (CN) or Subject Alternative Names (SANs) contained in the certificates match. We find 26 of 85 certificates mismatch (30.59%). This suggests that IoT devices do not correctly validate TLS certificates, rendering them susceptible to MITM attacks, or that they use certificate pinning with its associated problems. This is important as devices might contain old certificates that have been replaced on the backend, but cannot reach the backend. This mismatch "bricks" the devices or exposes them to security issues: Devices can no longer recognize valid backends and cannot download new certificates or security patches [139].

3.6.7 Longitudinal Analysis

We performed our initial measurements between August and November 2022. Here, we investigate on how the availability of backends changed over time and how their security posture evolved in September/October 2023 and January 2024.

IoT Protocols Analysis. We repeat our security assessment between September 15th–30th, 2023. We study 29,077 MQTT, 28,974 CoAP, and 124 XMPP vulnerable backends. Overall, 13,257 MQTT (45.59%), 13,573 CoAP (46.85%), and 35 XMPP (28.23%) backends are unresponsive or offline. IP addresses can be volatile, and backends

may no longer be reachable at the same IP. We overcome this shortcoming by scrutinizing the domains we collected and analyzing the backends' new IP addresses, focusing on 17,742 MQTT, 14,288 CoAP, and 48 XMPP vulnerable backends. In some cases, Shodan does not report any associated domains with the IPs, leading to fewer domains.

Some responsive backends no longer exhibit any security vulnerability, indicating that security issues were addressed. We find that 314 MQTT (1.08%), 149 CoAP (0.57%), and six XMPP (6.06%) backends are no longer vulnerable as of September 2023. Unfortunately, other backends have worse security. While 14 CoAP backends (-6.60%) are no longer vulnerable to CVE-2018-12679, 23 new backends (+10.85%) are now vulnerable. Similarly, 147 more MQTT backends (+35.85%) turned vulnerable to DoS threats, while only 84 backends (-20.49%) addressed the vulnerabilities. On a positive note, 185 CoAP backends (-23.75%) exposed fewer resources than in our previous analysis (16, +2.05%, exposed more resources). Finally, 262 MQTT backends (-2.17%) suffer from fewer CVEs from adopting older library versions. This is reflected in the number of backends that use newer updated library versions: 527 MQTT backends use newer software. Interestingly, we also discovered 40 backends that went backward to an older and vulnerable version.

We repeat our analysis a second time between January 23rd–31st, 2024, after we sent our disclosure emails, to understand whether developers who had been made aware of the vulnerabilities addressed them (see Section 3.7.2 for an in-depth discussion of our disclosure process). Overall, we encounter a similar instability as in our second scan and find 15,909 MQTT (54.71%, +12.12%), 15,203 CoAP (52.47%, +5.62%), and 38 XMPP (30.65%, +2.42%) backends unresponsive or offline, showing how, over time, datasets age quickly, resulting in 13,168 MQTT, 13,771 CoAP, and 86 XMPP responsive backends.

In addition to the backends that fixed their vulnerabilities in 2023, 74 MQTT (0.56%) and 72 CoAP (0.52%) backends no longer suffer from vulnerabilities. Similar to 2023, 145 MQTT (+30.66%) backends became more vulnerable to DoS attacks, while only 127 (-26.85%) addressed the issue. For the remaining vulnerabilities, we observe slight improvements. We report the overall trends for MQTT and CoAP backends in Figure 3.5. We do not show MQTT backends that suffer from weak authentication, i.e., the backends with credentials-based authentication, because they have not changed over the years besides the number of backends that became unreachable.

TLS Analysis. We repeat our TLS analysis between October 15th–30th, 2023. Given reachability instability for IP-based backends, we successfully analyzed 38,034 MQTT and 48 XMPP backends, of which only 25 MQTT (0.07%) and 3 XMPP (7.89%) backends support TLS. Interestingly, the two backends that previously served expired certificates provide the same old certificates, signaling poor security practices. We find 4 MQTT backends that show worse security in their TLS configuration. One now supports the outdated SSLv3 version, making it vulnerable to more attacks (e.g., SWEET-32).

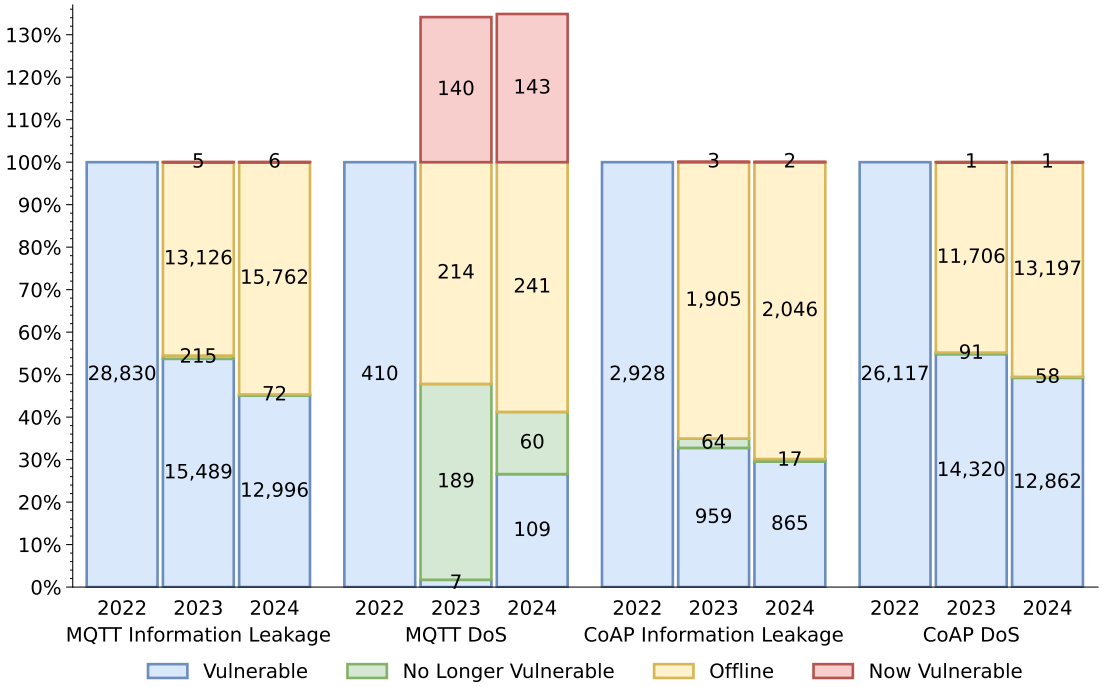


Figure 3.5: Vulnerable MQTT and CoAP Backends over the Years. We omit XMPP backends from the figure for clarity due to their low numbers. We clearly see the volatility of IoT datasets: after just one year, almost half of the backends are no longer reachable. Overall, more backends have fixed the vulnerabilities over the years. Interestingly, for MQTT backends, the number of newly vulnerable backends for DoS threats also greatly increases, signaling that, in such cases, older software versions are being rolled out.

3.7 Discussion

Following our study of the individual protocols and TLS, we discuss characteristics and statistics of our results, contextualize findings, and provide details on trends.

Results per Threat Class. We summarize our analysis results in Table 3.4. Considering the number of vulnerable backends per category, we see that a large fraction of reachable MQTT (11.47%) and XMPP (72.95%) backends are vulnerable to information leakage threats, with topic enumeration being a superset of backend fingerprinting for MQTT. For all reachable MQTT backends for which we collected topic names, we obtained fingerprint information, such as the software version or the number of clients. CoAP backends are particularly vulnerable to DoS attacks or are even enabling them (30.38%). This may be due to CoAP being UDP-based. Noteworthy is the large amount of CoAP backends that act as amplifying reflectors (30.18%), posing a severe risk to the Internet.

Results per Geographical Location. Concerning MQTT topics by country, most information leakage occurs for backends located in China (729,425), followed by the US

(422,771) and Germany (247,663). Except for China, this trend does not reflect the geographical distribution of brokers we observed in Section 3.5.1. On average, these countries reveal more topics per backend than the countries with most MQTT brokers.

We observe that 24,519 (94.57%) CoAP backends that are vulnerable to ZYXEL_000 are located in Russia, and they represent the vast majority of Russian CoAP backends (99.68%). The vulnerability ZYXEL_000 affects various Keenetic routers and enables DDoS attacks, potentially rendering thousands of devices unavailable. One reason for this localization in Russia could be an ISP providing vulnerable routers to its customers. Albeit Russian CoAP backends account for the majority of DoS amplifiers and reflectors, they expose fewer resources, with only 0.21% backends exposing any resources. Somewhat counterintuitively, backends in Europe and the US exhibited more information leakage in 2022, with 30.98% and 39.62% backends, respectively, exposing resources. This is despite stricter privacy regulations, like the European General Data Protection Regulation (GDPR). Interestingly, this improved in 2023, with now only 20.77% European and 27.99% US CoAP backends exposing resources. One reason might be that, in addition to the GDPR, the California Privacy Rights Act (CPRA) [51] came into effect on January 1st, 2023, requiring companies to put more care into handling users' data.

XMPP backends in the EU are generally more secure than those in the US, China, or Russia: 25.58% EU backends are vulnerable compared to 44.44–56.25% in other regions.

Overall, in 2023, compared to our 2022 results, the percentages of vulnerable backends in the analyzed countries slightly decreased for all vulnerability categories except for DoS threats for MQTT, for which we instead witnessed a slight increase in vulnerable backends in Europe (21 to 30), the US (18 to 22), and China (259 to 294).

Results per Cloud Providers. Taking a look at backends' deployments, we observe that MQTT backends hosted on larger cloud providers (AWS, Google, Azure, Alibaba) exhibit worse security posture than those hosted on *Other* (see Table 3.5). While we cannot identify a clear trend for XMPP, it unambiguously reverses for CoAP: Cloud providers generally show better security.

The trend of generally better security is, however, not uniform. Cloud-hosted CoAP backends leak more information than *Other*. In 2022, over one-third (34.76%) of AWS-hosted backends exposes at least one resource, contrary to only 0.68% *Other* backends. This improves substantially in 2023 for cloud-hosted backends: Only 17.63% AWS-hosted backends leak resources and the number of Google-hosted backends that expose resources halves (50.77% to 24.61%), indicating that security is receiving some attention. Fortunately, considering their network capacity, almost no cloud-hosted backends are vulnerable to DoS amplification vulnerabilities (0.75%), while almost one-third of *Other* backends are (30.58%). This number remains stable in 2023, suggesting that these providers adopt the latest security updates to mitigate abuse while other operators do not. At the same time, backend operators are still responsible for configuring the cloud-hosted backends properly to prevent information leakage, and the complementary disparity we observed provides a unique opportunity for future human factors research.

Table 3.4: Vulnerable Backends per Threat Class. We group the results of our individual analyses by vulnerability to provide a more comprehensive overview. A considerable share of backends in our dataset is vulnerable, potentially affecting the confidentiality, integrity and availability of user data.

	Threat Class & Analysis	# Vulnerable	Fraction
MQTT	*Information Leakage*	28,830	11.47%
	Backend Fingerprint	23,120	9.20%
	Topic Enumeration	28,830	11.47%
	Weak Authentication	12,071	4.80%
	Denial of Service	410	0.16%
	Known Vulnerabilities (backend)	410	0.16%
CoAP	*Information Leakage*	2,928	3.41%
	Backend Fingerprint	2,864	3.33%
	Resource Listing	779	0.91%
	Denial of Service	26,117	30.38%
	Amplification Factor (client)	25,939	30.18%
	Known Vulnerabilities (backend)	212	0.25%
XMPP	*Information Leakage*	89	72.95%
	Backend Fingerprint	89	72.95%
	Weak Authentication	59	48.36%
	Supported Authentication Mechanism	56	45.90%
	Compliance	4	3.28%

Table 3.5: Vulnerable Backends per Provider. Number of vulnerable backends by provider and protocol together with the respective fraction (computed on the total backends belonging to a specific provider). Considering their low numbers, we merge Cloudflare and Oracle with *Other*.

		AWS	Google	Azure	Alibaba	Other
MQTT	Total	4,036	606	854	131	245,753
	Vulnerable	2,883	409	625	45	25,091
	Fraction	71.43%	67.49%	73.18%	34.35%	10.21%
CoAP	Total	397	65	100	6	85,389
	Vulnerable	3	0	0	0	26,114
	Fraction	0.76%	0.00%	0.00%	0.00%	30.36%
XMPP	Total	7	-	1	-	118
	Vulnerable	5	-	1	-	93
	Fraction	71.43%	-	100.00%	-	78.81%

Self-hosted AWS vs. Managed AWS. Matching hostnames in our dataset against regexes of hostnames of services managed by AWS [261], we find only 125 instances that are managed MQTT backends (3.10% of AWS backends). Interestingly, we failed to connect to all 125 backends, possibly because they were unavailable or implement

Table 3.6: Count of Responses. We grouped the responses we received from the disclosure emails we sent into categories.

Type	Count
"Listed CVEs do not apply. Provide more information."	23
"We have informed the responsible parties."	15
"We fixed the issues with your information."	11
"We do not have time."	3
"We will let you know what we will do."	2
"The vulnerable client has been blocked."	1
Automatic Reply	428
Failed Delivery	700

ACLs properly. Indeed, AWS IoT adopts certificates to authenticate clients, which may prevent us from successfully connecting. They might also use Amazon Cognito to obtain (temporary) limited-privilege credentials. However, this does imply that these backends are secure. Companion apps might use credentials to authenticate their connection to the backends and carelessly hardcode the credentials in the app code [150]. Since we do not know if backends are associated with any app, we leave managed AWS backends for future work and consider them secure.

Looking at Table 3.5, we can see that around 71.43% self-hosted AWS backends are vulnerable to some threat class, from information leakage to DoS. Hence, we highlight the risk of inexperienced users misconfiguring AWS instances and potentially exposing sensitive information; this is less likely in the case of instances created directly by AWS, as our results show.

3.7.1 Ethical Considerations

We performed large-scale active measurements, also called scans, of real-world deployments. This prompts important ethical considerations, similarly to prior studies [68, 237, 284]. We followed the guidelines and best practices established in the Menlo report [32] and also discussed in recent work on cybersecurity research and network measurements [175, 232, 233]. Our study has been approved by the Ethics Review Board (ERB) of the University of Twente in the Netherlands, from where we also performed our measurements. The ERB assessed our setup and measurement methodology. In recognition of a historic lack of computer science expertise in the ethics review process, our ERB operationalizes the inclusion of cybersecurity expertise in its review, including guidance on coordinated vulnerability disclosure [250] (see also Section 3.7.2).

Active Measurement Setup. To prevent potential harm to the scanned backends, we put the following precautions in place: first, we limited the number of requests we perform with our scanners to limit our impact on backends. The machine we used to perform our measurements has a static IP address in our IP address space from the University of

Twente and a clear registered abuse handle. We also set up a reverse DNS entry with a descriptive DNS name for the IP address, and we host an informative web page on the same machine (reachable directly via IP address and DNS name). These measures aimed to quickly and clearly inform the backends' developers and maintainers about the nature of our measurement (e.g., by directly seeing `iotscan.eemcs.utwente.nl` in their log files and then visiting the website that provides more details), so that they could understand the scope of our study, and could contact us to request even more details or to be excluded from our study. We did not require a reason to be excluded from our measurements. Note that we received no exclusion requests.

Second, we only conducted *non-invasive* tests not to alter the state of the analyzed backends, carefully considering the trade-off between the utility of our study and potential harms. We only detect vulnerabilities, and we did *not* attempt to exploit them. This limits the accuracy of our findings, as in some cases the identified vulnerabilities may not actually be exploitable. We thoroughly evaluated the trade-offs that could cause unintended harm by our measurements with respect to its benefits, which include providing valuable knowledge to the scientific community and practitioners to protect users by discovering and reporting critical vulnerabilities, so that they can be addressed. The contributions we make in our work, especially considering the plethora of potential threats we discovered within the IoT to privacy, self-determination, and even life, outweigh the remaining minimal risks of our measurement.

Third, we did not read or store sensitive data. Albeit data could include sensitive information, such as usernames, we only collect the minimally necessary metadata (i.e., topics and resource names) to perform our assessment. We do not collect any potentially sensitive content, to mitigate potential unknown risks. Thus, our analysis is a lower bound, as other (and more) sensitive data might also be exposed. We test if resources (e.g., `/password`) are exposed via HEAD requests and the response code, that is, an existence check, and we do not request or receive any content.

Finally, we do not disclose the affected backends and put particular care into anonymizing our results by only presenting aggregates. When discussing case studies, we do not provide information that links to the specific backend. We will allow researchers to access our anonymized dataset after verifying their roles and institutions.

3.7.2 Vulnerability Disclosure

Given the scale of our study and the many vulnerable backends we identified, responsible disclosure is particularly challenging: it involves the operators of tens of thousands of backends. Therefore, we are collaborating with the Dutch Computer Emergency Response Team (CERT) and Cyber Security Center. We are following the coordinated vulnerability disclosure approach proposed by Reidsma et al. [250] and with the support of Ting-Han Chen and Jeroen van der Ham at the University of Twente.

First, we started our disclosure process with the two motivating examples we discussed in Section 3.4. We emailed the affected parties via the contact information we found

on their respective websites in May 2023. We sent an informative email stating who we are, the scope of our research, the vulnerabilities we found, how they could affect their backend, and suggestions on how to address them. We then sent follow-up emails in the following weeks, the first after 21 days and the second after 60 days. We only received feedback from one of the two impacted apps.

Second, for our large-scale study, we check whether an IP address falls within the address ranges of major cloud service providers (AWS, Google Cloud, etc.), in which case we disclose our findings directly to the providers. For the remaining backends, we use `WHOIS` data to extract their associated email addresses. We extract this information twice, for our first measurement (October 2022) and for our second one (September 2023), as addresses might have changed [297]. We extracted one or more email addresses for 273,151 MQTT, 290,901 CoAP, and 125 XMPP backends. We started our large-scale disclosures in November 2023 for the vulnerable backends affected by a CVE, accounting for the need to understand the evolution of security and privacy in the IoT ecosystem undisturbed while minimizing risks. Importantly, we have been working with the Dutch Cyber Security Center since May 2022, i.e., well before our first measurement, to determine the best and most effective disclosure approach that minimizes overall harm. We received 1,173 responses as of January 2024, categorized in Table 3.6.

We grouped all backends for which we reconstructed the same email address in a single email to reduce the total number, such as those by cloud providers and other large hosting providers. Some maintainers and developers may require more information and suggested solutions, which is why we drafted a file with an in-depth description of the identified vulnerabilities and how they impact their services. Moreover, although we cannot give detailed information on how to address vulnerabilities as we do not have access to their backends, we provide some general guidelines. For example, when vulnerabilities stem from old library versions, we suggest to update to a newer version. Similarly, for information leakage issues, we advise to:

1. Adopt authentication measures, e.g., (at least) passwords.
2. Adopt ACLs to prevent users from reading (all) messages, e.g., so that only admins can read sensitive topics.
3. Encrypt communication, e.g., using (D)TLS.
4. If the backend does not have to be exposed to the Internet, protect it behind a firewall (blocking incoming connections from outside the organization).

We refer the readers to further details on our experience as of January 2024, including still open challenges with vulnerability disclosure at scale in our workshop paper [59].

3.8 Limitations & Future Work

Our results show a problematic security immaturity in the IoT ecosystem. Concurrently to our work, Yaben et al. [326] identified over 1M endpoints using ZMap in December 2023, and painted a similar bleak picture on "abandoned or neglected" servers based on unpatched/outdated software and misconfigurations, such as weak authentication and expired certificates. We leave a comparison for future work. Following, we discuss possible threats to the validity of our study and further challenges for future work.

First, we acknowledge that our dataset might intrinsically contain geographical or provider bias. We could not find many backends in prior IoT traffic and were unsuccessful when asking other researchers to share theirs. At the same time, the high number of backends on Shodan shows a problematic lack of coverage of existing datasets and limited visibility into the ecosystem. Further, our dataset is mostly IP-based, as not all Shodan results include a hostname. As discussed in Section 3.6.7, relying on IPs can hinder stability. In future, we aim to gather a more heterogeneous dataset.

Second, we were limited in the range of vulnerability analyses we could perform. To preserve the correct functioning of the analyzed services and avoid disruptions or interruptions, we did not perform invasive measurements, for example, testing carefully crafted malformed payloads. Our analysis cannot guarantee complete insights about the security posture of IoT backends because there might be other potential threats they are exposed to, which we did not investigate. Nevertheless, our results clearly show that additional steps must be taken to improve IoT security.

Further, we cannot rule out that some backends we analyze are not part of the IoT. Some might be IoT-related or IoT-adjacent. Performing the necessary experiments to accurately assess whether a backend speaking an IoT-focused protocol is truly IoT would cross ethical and likely legal boundaries. Specifically, if we wanted to investigate the backends further and test if the connected clients are IoT devices, we would need to perform invasive measurements of the connected clients and instruct them to perform some action that we can use to determine that they are IoT. This is clearly much more ethically and legally challenging, if not downright impossible.

We also cannot rule out that our dataset contains honeypots. To the best of our knowledge, only one IoT-focused honeypot exists currently [286], which provides only basic functionality and has not been deployed widely. To quantify this issue, we employ Shodan Honeyscore [275], which has been integrated into the regular crawlers since its first release. We find only 36 instances of the MQTT backends for which the "honey*" keywords have been set, indicating that they almost certainly represent negligible noise. Unfortunately, without more invasive measurements, we cannot rule out that other backends are honeypots.

Finally, we focused on three widely adopted IoT protocols and considered more general protocols, like HTTP, out of scope. Despite HTTP usage in the IoT ecosystem [83], distinguishing IoT HTTP backends from non-IoT ones is extremely challenging. It

requires a semantic understanding of its API and manual inspection, which does not scale and requires invasive analyses. We leave their study for future work. Other non-application-layer IoT protocols, like Z-Wave, Zigbee, or RFID, have been studied by prior work [97, 329] and are local only, while we analyze public remote backends.

3.9 Related Work

IoT Protocol Security. Maggi et al. [176] investigated the security of MQTT and CoAP. They found sensitive healthcare data exposed by insecure MQTT brokers, such as patients' PII and ambulance locations. Additionally, they found 365,000 CoAP backends exposing network credentials. Palmieri et al. [228] showed the insecurity of MQTT backends, with 24,361 backends (60.38%) allowing clients to simply connect. They proposed MQTT-SA, a tool to assess MQTT deployments' security and detect possible misconfigurations. Jia et al. [148] successfully exploited MQTT device sharing or access revocation weaknesses to send unauthorized messages using "Will and Retained." Andy et al. [22] also investigated the implementation issues of MQTT, such as lack of authentication and encryption. Paracha et al. [230] studied how different IoT devices use TLS by collecting device traffic for two years. Their results show that some devices adopt old or insecure protocol versions or lack certificate validation. However, the studies and methods of prior work do not scale and are not applicable for publicly exposed backends, as they are invasive and potentially cause crashes.

IoT Analysis at Scale. Saidi et al. [261] studied the geographic location of IoT backend providers. They found that ~35% of IoT traffic at a major European ISP is going to providers outside of Europe, raising regulatory concerns. Srinivasa et al. [285] perform Internet-wide scans on six protocols, including those in our paper. They find over 1.8 million misconfigured IoT devices that can either be infected with bots or be leveraged for a (D)DoS amplification attack. Dahlmanns et al. [75] studied TLS adoption of ten Industrial IoT protocols, showing a low deployment rate (6.5%) and other widespread security issues (e.g., outdated protocol versions). Other work focused on large-scale identification of IoT devices and related events based on network traffic characteristics and packet signatures [82, 190, 222, 234, 279, 303]. Recent work investigated the IoT ecosystem at scale by focusing on mobile companion apps [58, 77, 247, 253], looking for security and privacy issues of devices without direct access to them. For example, most recently, IoTFlow [269] statically reconstructed network-related data such as URLs, including backends, contacted by 9,889 companion apps. However, prior work did not study IoT backend security at scale except for concurrent work by Yaben et al. [326].

3.10 Key Takeaways

Security Immaturity of IoT Backends. Prior work on studying the backend security for IoT protocols lacks coverage. We fill this gap by gathering an extensive dataset of

over 337,464 active backends (251,382 MQTT, 85,957 CoAP, and 125 XMPP). One in six backends (17.24%, 58,175 backends) suffer from Weak Authentication, Information Leakage, or DoS.

Weak Authentication. 59 XMPP backends (48.36%) adopt weak authentication mechanisms, like PLAIN, and 12,071 MQTT backends (4.80%) use credentials-based authentication in plaintext, enabling attackers to exfiltrate data easily.

Information Leakage of Sensitive Data. 31,847 MQTT, CoAP, and XMPP backends (9.44%) expose data ranging from version information to topic/resource names to messages for up to 40,017 security-sensitive MQTT topics.

Denial of Service. Nearly one third of CoAP backends (25,939; 30.18%) enable DoS amplification attacks. Although patches for the underlying vulnerabilities are already available, most backends use outdated, vulnerable software, enabling DoS attacks.

An Absence of Transport Layer Security (TLS). Today, a negligible 0.16% of the TCP-based backends use TLS, and the majority of them use old versions (70.93%, <v1.1) and suffer from known vulnerabilities (66.28% BEAST, 46.51% SWEET-32). Adapting and adopting TLS in resource-constrained devices is a simple approach to improve security quickly.

Cloud Hosting. Most cloud-hosted backends (30,819; 96.96%) are run by AWS, Google, or Azure, supporting their dominance. However, utilizing a large cloud provider does not automatically lead to better security, for example, cloud-hosted MQTT backends exhibit worse security.

Large-scale and Highly Problematic. The identified issues affect non-critical and critical infrastructure alike: an MQTT backend, likely related to a power facility, exposes 36,254 topic names, while 99.68% of Russian CoAP backends enable amplification attacks due to the widespread adoption of Zyxel routers.

3.11 Conclusions

In this paper, we perform a large-scale measurement of the security posture of over 337,000 IoT backends that use MQTT, CoAP, or XMPP, focusing on three main threats: information leakage, weak authentication, and DoS potential. We find that many deployments for all three protocols are vulnerable: 31,847 of the reachable backends (9.44%) expose (sensitive) information, a conspicuous fraction of CoAP backends (30.18%, 25,939 backends) are vulnerable to amplification attacks, and only a negligible number of MQTT and XMPP backends adopt TLS (0.16%), of which 70.93% use outdated protocol versions (<v1.1). Our study provides evidence for a troubling immaturity of security in the IoT ecosystem, which was not analyzed thoroughly at scale before. We responsibly disclosed the identified issues to the affected parties, support their remediation efforts, and hope to improve their security awareness.

CHAPTER 4

Are You Sure You Want To Do Coordinated Vulnerability Disclosure?

The rising numbers of vulnerabilities and security issues stemming from the rapid iteration and development of the IoT have introduced new challenges for the stakeholders to mitigate in time. To effectively bring researchers, vendors, and end users together to address such problems, Coordinated Vulnerability Disclosure (CVD) has become standard practice. Although general CVD procedures for practitioners to follow exist, adapting them to the specific circumstances has proven to be complicated in practice.

In this paper, we document our experience of reporting various security vulnerabilities for 15,820 IoT backends. The discovery and scanning have been part of a separate research project, in this contribution we focus on the disclosure to the backends' operators in a large-scale coordinated vulnerability disclosure effort, following the latest disclosure guidelines. We discuss what we have learned to inform others who want to engage in large-scale CVD, we compare the steps and tradeoffs of our effort with current CVD suggestions, based on our measurement before and after the disclosure, and we describe how adapting our approach can improve CVD best practices.

4.1 Introduction

The widespread adoption of the Internet of Things (IoT) has raised many challenges in ensuring the safety of Information and Communications Technology (ICT) infrastructure and users' privacy on an Internet-wide scale. The number of discovered vulnerabilities has increased rapidly over the past years, affecting the growing number of network-connected devices and their, typically publicly reachable, backend services. That is, stakeholders may

expose vulnerable network backends and devices that attackers could abuse maliciously. To improve security, including Internet security and IoT device security, vulnerability disclosure has taken an essential role, encouraging active collaboration between security researchers, practitioners, operators, stakeholders, and end users. For example, to mitigate identified issues before they are exposed, and attackers could abuse them. Intuitively, the accuracy of disclosure information shared between the parties determines if security actions are taken promptly and without hesitation, and, with disclosure, the process extends beyond a purely technical understanding of the vulnerability to also include the intricacies and challenges of human-to-human communication.

Coordinated Vulnerability Disclosure (CVD) aims to establish a constructive and practical communication between security researchers and stakeholders, by guiding the researchers in conducting vulnerability disclosure and initially limiting reporting of the discovered vulnerabilities to vendors and affected parties, before eventually publicly revealing them. This allows both sides to collaborate and mitigate the problems before the information becomes public and attackers could abuse it. Finally, the vendor informs the users to act on the discovered and addressed issues. CVD has enhanced vulnerability disclosure, but there are still tradeoffs and concerns to consider [132]. For example, researchers may struggle to locate the correct contact for stakeholders [57], especially when engaging in large-scale notifications. The level of detail to disclose and the required methods to remedy the issue can also trigger unwanted behavior from the stakeholders. Similarly, it can be challenging for the stakeholders to identify the true severity of the disclosed issues, which might hinder their mitigation efforts. Fortunately, prior work proposed well-structured strategies, based on experience from researchers, practitioners, and operators in the last decade, to address these concerns [196, 312]. We conduct our worldwide disclosure process leveraging their suggested methodologies and we provide our own experience as new feedback to improve the CVD process.

In earlier work, we analyzed and measured the security posture of IoT backends, that is, backends speaking IoT-focused protocols (e.g., MQTT), at scale, focusing on understanding if backends suffer from several known and high impact vulnerabilities [211, 212, 213, 214, 217]. Overall, we identified 15,820 potentially vulnerable backends and prepared the disclosure of the discovered vulnerabilities to the responsible parties. We followed current CVD guidelines to encourage stakeholder cooperation and prompt mitigation. Indeed, our disclosure efforts were successful in that some stakeholders fixed their vulnerable backends. However, we also received unexpected responses and faced difficulties when informing affected parties with complex operational structures. By discussing our experience and actual consideration of each step of our disclosure process, we shed light on formulating an adequate disclosure for large-scale vulnerability disclosure, and we provide new insight on the connection between CVD as a model and implementing it.

4.1.1 Contributions

In this paper, we make the following contributions:

- We performed large-scale responsible disclosure for IoT backends suffering from vulnerabilities, based on current Coordinated Vulnerability Disclosure (CVD) best practices, and we measured stakeholders' responses after our disclosure notifications.
- We provide comprehensive guidelines to facilitate large-scale disclosure notifications and handle stakeholder communication.
- We discuss what we learned from our disclosure and we provide practical suggestions to further improve CVD best practices.

4.2 Background and Related Work

This paper focuses on our Coordinated Vulnerability Disclosure experience. We briefly describe our measurement results. We then discuss recent updates of CVD practices and how we adopt them into our disclosure operation.

4.2.1 Security Assessment

We leveraged Shodan [276] to identify backends that speak common IoT communication protocols, like MQTT, CoAP, and XMPP. Together with the IP addresses and hostnames for the backends, we also collected Connection Codes and geolocation information. We then performed our security evaluation, focusing on three main attack classes: information leakage, weak authentication, and denial-of-service. We define the testing pipeline for each protocol and provide more details in our prior work [301]. In this paper, we instead focus on the disclosure process for the discovered vulnerable backends. Overall, we discovered numerous backends suffering from known vulnerabilities, identified via Common Vulnerability Exposures (CVEs), and other common security pitfalls, such as not requiring authentication or unintended exposed access. We then compiled a list of vulnerable backends with their IP addresses, CVEs or vulnerabilities if no CVE was assigned, and other issues that we identified, which we utilized to disclose our findings to the backend operators following current CVD best practices.

We first analyzed the security of the backends in September 2022, and later repeated our analysis in September 2023. Table 4.1 shows the number of vulnerable MQTT backend for September 2023 for each of the five CVEs that we tested for. Since CVE-2018-12550 and CVE-2018-12551 always occurred together, we combine them in the table. All CVEs were published over three years before our scan, and mitigations have been available in the form of an updated version since the vulnerability became public, which should be ample opportunity to update and resolve the issue. We do not provide data for CoAP and XMPP because our disclosure process for backends speaking these protocols is ongoing.

Table 4.1: We report the number of backends affected by CVEs we tested for grouped by protocol. CVE-2017-7655, CVE-2018-12550, and CVE-2018-12550 affect the Eclipse Mosquitto MQTT server, CVE-2018-19417 affects Contiki-NG, and CVE-2019-9749 affects Fluent Bit.

Protocol	CVE	CVSSv3	# Found
MQTT	CVE-2018-12550 [213] CVE-2018-12551 [214]	8.1	6,236
	CVE-2017-7655 [212]	7.5	4,241
	CVE-2018-19417 [211]	10.0	186
	CVE-2019-9749 [217]	7.5	151

4.2.2 Coordinated Vulnerability Disclosure

CVD is a well-known process that has become accepted best practice and the standard operating procedure for most security researchers. Its idea is to contact vendors or operators of affected services and products, like backends, before publicly disclosing the existence of vulnerabilities, so that they can address the vulnerability during an embargo period. There is no minimum or maximum time length for an embargo, but approximately 90 days has become typical. Only after the embargo period, the parties publish about the vulnerability, so that others can take knowledge about the vulnerability into account for their decision-making processes, including the remedy, or learn from it.

This basic approach is widely accepted and described [145, 146, 223]. In practice, however, this turns out to be more complicated, especially if multiple parties are involved [103], and this procedure must be adapted. This has led the University of Twente, where we initiated our disclosure process, to define an outgoing vulnerability disclosure policy [251, 308]. This policy also explains to the recipients of our notifications how we approach the disclosure process, for example, clarifying our timelines and expectations. The policy builds on the disclosure guidelines of the Dutch National Cyber Security Centre (NCSC-NL) [198]. For our CVD, we strictly follow these policies.

We apply methods recommended by the Dutch Institute for Vulnerability Disclosure (DIVD) and will describe our results and experiences using their framework and C-HIP model [323]. Moreover, we use the investigation of the state of CVD by van der Horst to address difficulties of large-scale disclosure efforts for our process [309]. The DIVD has extensive experience in large-scale vulnerability disclosures, but has focused primarily on disclosing critical corporate infrastructure bugs, such as the Citrix vulnerability in early 2020 [74]. In contrast, we investigate the disclosure of IoT backend vulnerabilities, a setting distinctly different in terms of security considerations.

We locate the contacts of IoT backends via the WHOIS databases [165, 293]. We use simple language for our disclosure notifications, see below, and we also set up a

static webpage, detailing the research to inform stakeholders[1], akin to prior work [46, 307]. Since some IoT backends are hosted on public clouds, it can be challenging to identify the actual operator. Therefore, we also notified cloud providers, with the request to forward our notification to the affected customer. During our interactions with stakeholders, we observed that not all explicitly specify their disclosure policies or procedures. Unfortunately, the recommended practice for vendors or operators to serve an informative `security.txt` file containing security contact-related information for security researchers to contact [105] is not yet widely used. Moreover, it only applies to web-based systems or requires knowledge about who the actual operator of the vulnerable service is and their website, which is particularly difficult to determine for services deployed on public clouds, further limiting its usefulness for our work.

Dear Sir/Madam,
As a collaborative project from the University of Twente and TU Wien, we have been investigating the security of publicly available IoT services. We examined the server endpoints that IoT devices and smartphone apps connect to. Our study has revealed a security issue that might require your immediate attention. We conducted a scan and found the following vulnerable machines (IP address, Found Time, and CVEs):
`192.0.2.42 2022-08-24 00:03:02.000000000 +0000`
`CVE-2018-12550, CVE-2018-12551`
You can find out more about our study and an explanation of the results on our project website: https://iot-disclosure2023.dacs.utwente.nl
These findings have been kept confidential and all the rules stated by our Dutch National Cyber Security Center (NCSC) [1] were adhered to during this research. We are planning to follow the procedure as outlined in our public disclosure policy [2].
If you are not running this server yourself, but know the responsible party (e.g., because they are your customer), please forward this information to them, you have our explicit approval for this.
Regards,
IoT Disclosure Project, UTwente
[1]: https://english.ncsc.nl/get-to-work/implement-a-cvd-policy/finding-vulnerabilities-in-it-systems
[2]: https://www.utwente.nl/en/service-portal/research-support/procedures-facilities/coordinated-vulnerability-disclosure-policy-for-research

4.3 Methodology

Following, we describe how we structured our large-scale disclosure. We first gathered the required contact information and composed the disclosure notifications to the correspond-

[1] https://iot-disclosure2023.dacs.utwente.nl/

ing contacts. We then notified the stakeholders at scale and tracked the conversations via ticketing methods.

4.3.1 Disclosure Procedures

Contact Information Retrieval

We identified 15,820 IP addresses of vulnerable IoT backends based on our security assessment [301]. We retrieved the corresponding contact information for the IP addresses via the WHOIS databases of APNIC, RIPE, etc. Eventually, we gathered email addresses and contact details for 15,046 IP addresses (95.11%). Unfortunately, for some backends, the WHOIS database contains no useful information, that is, it simply returns a general abuse contact for the corresponding Regional Internet Registry.

Disclosure Messages

We composed our disclosure notification template to ensure our disclosure messages are clear and informative to bootstrap conversations with the stakeholders (see Section 4.2.2). In our initial message, we inform each stakeholder about the identified vulnerable backends, providing information on the IP addresses, CVEs, and vulnerabilities we found. We briefly describe the severity of the vulnerabilities that require the stakeholder's prompt action and we point them to a dedicated website with more detailed information. The website provides an overview of the issues, links to additional material by MITRE, and an email address to contact us.

We reply with a second message (see below) when stakeholders return to us with additional questions. Our reply includes details of the vulnerable backends and suggestions for mitigations. The exact content may vary depending on the security issues of backends and the stakeholders' responses.

> Dear Sir/Madam,
> As a collaborative project from the University of Twente and TU Wien, we have been investigating the security of publicly available IoT backends. We examined the server endpoints that IoT devices and smartphone apps connect to. Our study has revealed a security issue that might require your immediate attention. We conducted a scan and found the following vulnerable machine (IP address, Found Time, CVEs):
> `192.0.2.42 2022-08-24 00:03:02.000000000 +0000`
> `CVE-2018-12550, CVE-2018-12551`
> In our scanning method, we attempted to subscribe to your MQTT services with wildcards "$SYS#" and "#" on Port 1883. We collected the number of topics (680) and the number of connected clients (2) at the time of our scan. We did not explore your system further. Yet we believe an attacker can use the same way to extract all MQTT content and may get access to sensitive information such as Personal Identifiable Information or security device status of associated systems.

Our suggested improvements are:

- Adopt some authentication measures, e.g., password-based authentication.
- Adopt some Access Control Lists to prevent all users from reading all messages, e.g., only admins can read certain sensitive topics.
- Encrypt the communication, e.g., by using TLS (some lightweight versions of it can work with IoT devices, e.g., https://www.wolfssl.com/)
- If the broker does not have to be exposed to the Internet, protect it behind a firewall (blocking connections from outside the organization) or a NAT.

You can find out more about our study and an explanation of the results on our project website: https://iot-disclosure2023.dacs.utwente.nl
These findings have been kept confidential and all the rules stated by our Dutch National Cyber Security Center (NCSC) [1] were adhered to during this research. We are planning to follow the procedure as outlined in our public disclosure policy [2]. If you are not running this server yourself, but know the responsible party (e.g., because they are your customers), please forward this information to them, you have our explicit approval for this.
Regards,
IoT Disclosure Project, UTwente
[1]: https://english.ncsc.nl/get-to-work/implement-a-cvd-policy/finding-vulnerabilities-in-it-systems
[2]: https://www.utwente.nl/en/service-portal/research-support/procedures-facilities/coordinated-vulnerability-disclosure-policy-for-research

4.3.2 Notification System

To notify the contacts at scale, we set up a dedicated email account with the domain name utwente.nl to send our disclosure notifications. We chose this approach to ensure our notifications have a high chance of being delivered [292, 309] and that they are not discarded by recipients, for example, because of a low sender reputation.

We used the email server of the University of Twente, which uses Microsoft Exchange. We sent out notifications in batches, due to limits enforced by Microsoft Exchange service policies. To facilitate large-scale disclosure, we implemented a custom notification script in Python, which interacted with our SMTP server to send disclosure notifications while respecting the limits of the mail server. For each message, our implementation also logs the timestamp, contact email addresses, and identifiers.

Before sending notifications, we grouped the vulnerable backends by contact address, that is, we combined those for which the contact address is the same. This grouping enabled

us to drastically reduce the emails we had to send from 15,046 to 2,132. On average, we reported seven vulnerable backends per disclosure notification (max 1,555). We ensure brevity and understandability of our disclosure message by listing only a subset of up to ten vulnerable backends in the body of our message and including the complete list of vulnerable backends as a text file attachment.

4.4 Results

Following, we discuss the results of our disclosure effort with the stakeholders, how we initiated the conversations, the challenges we encountered, and what we learned.

4.4.1 Measurements and CVD Timeline

We performed our initial security assessment in September 2022, followed by a second assessment in September 2023, to understand how the security posture of backends evolved. After evaluating the results of both measurements, we sent disclosure notifications from November 10th, 2023, to November 22nd, 2023. Overall, we sent 2,132 emails to notify contacts associated with 15,046 IP addresses. We received 32 requests for additional information. After responding to those 32 requests, 28 stakeholders then initiated an active conversation with us, who we helped locate and fix their issues. The remaining 4 did not further respond and did not fix the identified security issues. An additional 311 stakeholders fixed the identified vulnerabilities without responding to us.

4.4.2 CVD with Stakeholders

After disclosing the issues, we received 1,173 responses to our notifications from stakeholders between November 2023 and January 2024. Table 4.2 summarizes the numbers. Most of them (428, 36.48%) were automatic responses from the stakeholders' ticketing systems. We also received replies from individuals, such as service operators, company incident response teams, or users, who were notified by the cloud service providers. Unfortunately, we received 700 (59.67%) delivery failure notifications, which we discuss further in Section 4.4.2.

Automatic Responses

Ticketing systems generated automatic responses. Most stakeholders, such as cloud service providers, universities, and business organizations, use ticketing systems to handle incoming email messages or abuse reports. These systems received our disclosure notifications and forwarded them to the responsible parties within these entities. In 27 cases, the ticketing systems required us to sign in to their portal with automatically generated accounts to confirm our disclosure, likely to prevent spam. We followed the instructions to confirm our messages and ensure our notifications were delivered successfully. However, three backends not only required us to provide our contact information but also requested us to accept their policies, for example, privacy policies,

Table 4.2: We manually cluster the stakeholder responses in four main categories. We mark as *Automatic Response* those that were automatically sent by ticket systems etc. We flag as *Manual* the responses of interested parties that contacted us after our disclosure emails, split into *Question* for inquiries about more details and *Update* for informing us about the issue being resolved. We denote as *Delivery Failure* notifications that we sent but which were returned by the recipient email server as undeliverable.

Type		Core Message	Count
Automatic Response		Thank you for your disclosure. We received your message and will inform the responsible parties.	428
Manual	Question	Please provide more details about the vulnerabilities and remedies.	32
	Update	We mitigated the disclosed issues.	13
Delivery Failure		Contact not reachable.	700

which were only available in their native languages. We opted not to accept these policies with uncertain impact, which may have led these parties to ignore our disclosures. Besides the situations above, 51 ticketing systems correctly forwarded our notifications to the affected parties. Seven cloud providers' systems informed us about updates on their communication and customer progress. In two cases, they established direct communication between their customers and us through their ticketing system.

Manual Responses

We consider replies to our disclosure notifications that are from stakeholders who are directly responsible for the vulnerable backends as manual responses. We separate them into two classes, first, questions concerning the disclosed vulnerabilities, and, second, updates or comments by them regarding the coordinated disclosure process (e.g., that a vulnerability was mitigated). We received 28 positive replies and valuable suggestions, as well as 4 complaints. In Section 4.4.3, we discuss how we handled all manual responses and the corresponding active coordination of our CVD.

Delivery Failures

Not all our disclosure notifications were delivered successfully to the responsible parties. Delivery failures can occur for several reasons, such as contact addresses no longer being valid or the recipient email server employing restrictive filtering rules. We received 700 delivery failure notifications and attempted to cross-check these with our outgoing emails, as our disclosure notifications could have multiple recipients, a 1-to-1 matching is not possible. For 103 out of 2,132 messages (<5%), we received failure notifications for all recipients. This affected 210 backends (<2%) for which we were not able to notify the operator at all. For the remainder, notifications failed to deliver only for some recipients, but not all, and we assume that at least one notification was successfully delivered. We describe some reasons for delivery failures below but have not been able to categorize all cases exhaustively, which is why we can not provide exact numbers.

If a delivery failure occurred, we could not notify the affected parties if we could not find any alternative contact method. The contact email addresses' filtering rules might have also blocked our messages. Some email servers restrict the length of content messages, attachment type, and file size, including to abuse and other contact addresses for operators. To our surprise, some even reject small text file attachments. This was problematic for our disclosure because we included a text file attachment with the complete list of affected vulnerable IP addresses, ports, timestamps of scanning, and CVEs for all emails and only included up to 10 results in the email itself to prevent messages being marked as spam. If the recipient server rejects our text file attachments, then this further complicates the process of notifying the affected stakeholders.

4.4.3 Handling the Responses from Stakeholders

A core component of CVD is establishing a trusted and honest communication channel with the stakeholders. Our initial disclosure message aims to draw the recipient's attention to the discovered issues and establish a communication channel to coordinate with the affected stakeholder on the embargo timeline and how to remedy the reported issues. During this process, the stakeholders sent us questions on the details of our findings, suggestions on our disclosure method, and different perspectives on our vulnerability findings. We also received complaints regarding our disclosure operations. Following, we detail in depth how we interact with the stakeholders, what we have learned from the disclosure process, and how we incorporated their suggestions into our later disclosures.

In our initial batch of notifications, we sent notifications to recipients associated with vulnerable IP addresses for MQTT backends, both with and without detected CVEs. We also included additional information and a reference to our website that included more information about our scans and disclosure practices.

After our notifications, 32 recipients replied to our message requesting additional information. We provided them with additional information (see Section 4.3.1), such as how we found the vulnerability and security advice on mitigating the issues. After this, 28 respondents reported that they were working on locating the problems and informing related parties, such as their clients and other operators.

Over 60% of 15,820 backends were hosted on cloud platforms, preventing us from directly notifying the operator and only notifying the cloud providers. While we could not contact the operators directly, the cloud providers relayed our notifications to their customers and other responsible parties. After we informed the responsible parties, we received their feedback by email directly or through the cloud providers' ticketing systems.

We also received eight complaints that the vulnerable backends were not part of the contacts' operation and not under their control. Since we collected the contact email addresses via WHOIS, the contact addresses may have been inaccurate.

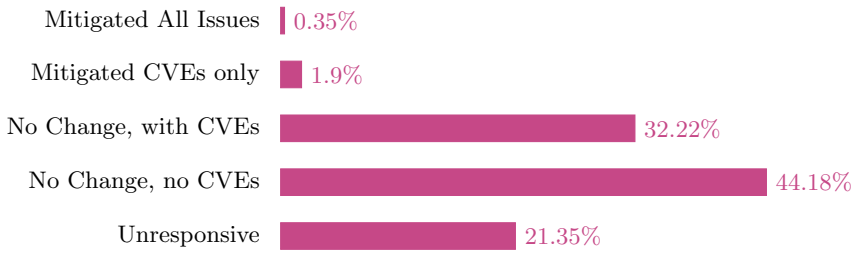


Figure 4.1: Security posture of vulnerable IoT backends as of January 2024, approximately two months after we sent our disclosure emails in November 2023. All numbers are relative to the 14,836 IoT backends for which at least one disclosure notification was delivered, that is, the number of backends we consider as successfully notified.

4.4.4 Before and After Our Disclosures

We conducted a third security assessment in January 2024, after our disclosures, to measure and analyze if the stakeholders mitigated the vulnerabilities and adopted our recommendations to improve their security.

Figure 4.1 provides an overview of the results, as a percentage of the 14,836 backends that we successfully notified (notifications for 15,046 backends total, but delivery failed for all contacts for 210 backends, which we consider as not notified). The operators of 52 (0.35%) backends addressed the vulnerabilities and also implemented our security recommendations, such as requiring authentication. Meanwhile, for 282 (1.90%) backends the operators only mitigated the reported vulnerabilities identified by their CVEs, but they did not implement our other recommendations. For example, we could still observe exchanged messages, without needing to authenticate. Unfortunately, 3,168 (21.35%) backends became unresponsive, likely because of IP address instability. Although only few 334 (2.25%) backends mitigated vulnerabilities, this is not unexpected given the limited accuracy of WHOIS contact addresses. We have received a reply from 31 service providers that they have forwarded our disclosure notifications to their customers, but we cannot be certain that they reached the customers unless they replied to us.

Worse, 4,780 backends (32.22%) remain vulnerable. The remaining 6,554 (44.18%) backends have no known vulnerabilities tracked by CVEs. Still, they expose resources to the Internet and do not require any authentication, which, for example, leaves broadcast messages unprotected. From our CVD, we learned that these backends range from university projects to commercial applications to enterprise-grade IoT backend services.

4.5 Ethical Considerations

Throughout our measurements [301] and the disclosure process, we kept the impact of our work in mind and followed best practice and CVD guidelines. We received approval for our measurements from the Ethics Review Board (ERB) at the University of Twente.

In this paper, we share our CVD approach and experience, so that the community and others can benefit from them and further improve CVD processes. Our disclosure operation was the first large-scale CVD implementing the outgoing vulnerability disclosure policy of the University of Twente [251, 308]. No respondents remarked on this policy explicitly, but it provided a public guideline for timelines and our predictable actions. Some respondents might have decided to not respond and simply mitigate the issues without engaging with us after taking note of it. We recommend other researchers also clearly communicate and publish such an outgoing CVD policy.

4.6 Reflections on our CVD

We set up our disclosure notification operation based on the guidelines given by DIVD [309], which is why we compare it to the DIVD suggestions, the existing theories, and best practices. We follow the Communication-Human Information Processing (C-HIP) model [323], focusing on the following elements, to better present the comparison.

4.6.1 Channel & Delivery

We used email as our notification method. The vulnerable backends are maintained by diverse organizations with different disclosure practices across countries. Correspondingly, for our large-scale disclosure notification for these backends, the WHOIS abuse contact information for the affected IP addresses is the most reliable way to reach as many responsible parties as possible. It was not feasible to establish a second channel, such as a notification-sharing platform, since the contacts are not within the scope of an organization. For backends hosted on cloud infrastructures, we informed the providers and asked them to forward our messages. We received copies for some cloud providers, who sent a request to their clients to fix the issues or take down the backends. For other disclosures to cloud providers, we were not privy to any notification to their customers.

4.6.2 Attention Maintenance & Comprehension

The recipients of disclosures must quickly identify them and keep track of the vulnerability disclosure process. In our disclosure template (see Section 4.3.1), we provided brief but essential information to identify the vulnerable backends. Our goal was to draw the recipients' attention without overwhelming them with information. However, there is a trade-off between message length and effectiveness. Stakeholders typically use email ticketing systems, which record our disclosures and inform the system operators. Certain cloud providers have also stringent restrictions on email content, filtering messages that are too long or include attachments that are too large. Using HTML-based content, such as colors, different fonts, or other graphic content in our disclosure message is also not ideal, because it could trigger spam filters.

4.6.3 Attitudes & Motivation

To communicate positively and encouraging with the affected stakeholders and find a balance to motivate them to actively engage in the disclosure process, we have incorporated stakeholders' responses and suggestions into our notification methods. Early respondents requested that we include a brief description of the severity of vulnerabilities, IP addresses, port numbers, and preferably corresponding CVEs, which they see as the optimal information to include in the disclosure messages. We remedy the fact that CVEs may not be understood by all recipients equally by providing additional security suggestions in simple text, which we hope helps reduce the efforts of stakeholders to mitigate the vulnerabilities. We also specify our contact information in our disclosure email, and we provide a link to our webpage that provides more details. Contacts that actively engaged in the CVD informed us that they found this helpful. Another important concern is the language to communicate with the stakeholders. Since our disclosure were large-scale and world-wide, we composed our messages in English. Interestingly, we received 112 responses, mainly from stakeholders' ticketing systems, that were in the stakeholders' native languages, which increased our efforts to react to those responses.

4.6.4 Receiver & Behavior

WHOIS email contact information may not always be accurate, which means that we could potentially leak information on vulnerabilities to the wrong recipient. Indeed, this was the case for eight recipients, who complained that we informed the wrong parties. We aim to mitigate this situation by limiting the vulnerability details in our first message and not providing every detail. However, this can conflict with the interest of stakeholders, who prefer a clear and informative initial message, for example, to triage it internally. Another consideration is the tendency of stakeholders to start mitigating the vulnerabilities without notifying us. Whether stakeholders informed us about their process appears to also depend on their own vulnerability management policies and willingness to participate in our CVD. Another contact method than email could improve on these aspects, for which our next step is to construct a reliable disclosure channel with the stakeholders. The responses we received show that well-known cloud providers and organizations host numerous IoT backends. Establishing long-term connections and communication channels with these organizations could allow us to directly reach the responsible parties and improve CVD effectiveness.

4.7 Future Work

Traditionally, meta data regarding IP addresses, like contact addresses, is accessible through WHOIS, which is unstructured text for which it can be difficult to retrieve the correct email address. The Registration Data Access Protocol (RDAP),[2] which provides structured access to this meta data, has been introduced as a successor to WHOIS and

[2] https://about.rdap.org/

could improve delivery reliability for CVD notifications. Unfortunately, while RDAP is in the process of being deployed, WHOIS remains more widely supported, reliable, and more frequently updated [101]. Future work, especially once RDAP becomes more widely used, should explore the benefits of RDAP for retrieving accurate contact information in more depth. Moreover, given its extensibility, RDAP provides an opportunity to include a disclosure contact address, which could improve CVD effectiveness.

4.8 Conclusion

Coordinated Vulnerability Disclosure (CVD) is a well-established and proven approach to unite security researchers, vendors, operators, and end users to mitigate the identified vulnerabilities in a joint effort. Unfortunately, as we have experienced in our large-scale CVD of vulnerabilities spanning over 15,820 IoT backends that we discuss in this paper, it remains challenging for researchers to set up reliable infrastructure to facilitate large-scale CVD and establish practical and trusted communication with stakeholders when considering the intricacies of real-world deployments. Following current best CVD practices to contact operators, we could still not inform the operators of 984 backends (6.3%) about their vulnerable systems. CVD does lead to improvements in security for IoT backends, these are limited to a low 2.28% of backends, while a total of 77.5% of backends see no security improvements after disclosure. That is, our findings suggest a troubling immaturity regarding backend security awareness and the adoption of CVD in the IoT ecosystem. Overall, leveraging our own experience, in this paper, we provide new insight on the struggles when performing large-scale CVD at the example of IoT backends and we make new suggestions to improve current CVD practices.

CHAPTER 5

I Still Know What You Watched Last Sunday: Privacy of the HbbTV Protocol in the European Smart TV Landscape

The ever-increasing popularity of Smart TVs and support for the Hybrid Broadcast Broadband TV (HbbTV) standard allow broadcasters to enrich content offered to users via the standard broadcast signal with Internet-delivered apps, e.g., ranging from quizzes during a TV show to targeted advertisement. HbbTV works using standard web technologies as transparent overlays over a TV channel. Despite the number of HbbTV-enabled devices rapidly growing, studies on the protocol's security and privacy aspects are scarce, and no standard protective measure is in place.

We fill this gap by investigating the current state of HbbTV in the European landscape and assessing its implications for users' privacy. We shift the focus from the Smart TV's firmware and app security, already studied in-depth in related work, to the content transmission protocol itself. Contrary to traditional "linear TV" signals, HbbTV allows for bi-directional communication: in addition to receiving TV content, it also allows for transmitting data back to the broadcaster. We describe techniques broadcasters use to measure users' (viewing) preferences and show how the protocol's implementation can cause severe privacy risks by studying its deployment by 36 TV channels in five European countries (Italy, Germany, France, Austria, and Finland). We also survey users' awareness of Smart TV and HbbTV-related risks. Our results show little understanding of the possible threats users are exposed to. Finally, we present a denylist-based mechanism to ensure a safe experience for users when watching TV and to reduce the privacy issues that HbbTV may pose.

5.1 Introduction

As of 2021, 1.72 billion TV households exist worldwide [296], and each viewer, on average, spends around three hours per day watching TV [294, 295]. Thus, TV content can significantly impact society, for example, depending on the content and spin of news headlines—in addition to being a valuable target for advertisers. However, traditional "linear TV" (i.e., content broadcasted as scheduled programs, including commercial breaks) through satellite or cable has faced stiff competition from new on-demand streaming services, like Netflix, Hulu, HBO Max, Amazon Prime Video, and Apple TV+.

Thus, to combine standard TV's broadcast content delivery with the powerful digital content delivery of the new platforms and improve the viewing experience for users, an industrial consortium launched the *Hybrid Broadcast Broadband TV (HbbTV)* [122] initiative in 2009. HbbTV sets a standard for a broadcast/broadband hybrid protocol to deliver content to Smart TVs, set-top boxes, and other connected multiscreen devices in an interconnected environment. In this setting, an HbbTV application is loaded and executed by a Smart TV's built-in browser and displayed as a graphic overlay on top of regular broadcast content. In addition, HbbTV transforms the traditional TV viewing experience from merely *receiving* content to also *transmitting* data, enabling new functionality for users (e.g., interactive programming and shopping) and broadcasters (e.g., measuring viewing preferences).

The adoption rate and support of HbbTV have been growing steadily. As of 2022, HbbTV has been adopted across European countries, Australia, Russia, and Vietnam [121]. Germany is the leading country in HbbTV adoption and was the first to adopt this standard: over 90% of Smart TVs sold support the HbbTV standard [121]. Looking at the general HbbTV adoption in the European countries we focus on, compared to the general numbers of TV households (i.e., households with a TV set), 8.95 million out of 25 million (44.75%) in Italy, 2.5 million out of 28.8 million (6.68%) in France, and 18 million out of 38.52 million (46.73%) in Germany have HbbTV-enabled devices [113, 121, 163].

Thus, unlike previous work that studied security and privacy aspects of the Smart TV's firmware and apps, we focus on the transmission protocol itself. Recent studies have shown that HbbTV provides users with little or no security and privacy. Most notably, Ghiglieri et al. [109, 110, 111, 112] assessed HbbTV's privacy posture, highlighting the severe risks users were exposed to. Additionally, little or no control is given to the viewer; they cannot detect whether a connection is secured, which data is transferred, or how it is used. HbbTV's security and privacy issues are manifold. They range from a simple echo request from the broadcaster to check if the user is still watching to content-based attacks that replace URLs to show viewers different content than was intended. These issues have also been abused in practice, most recently in May 2022 when hackers exploited HbbTV broadcasts of Russian TV stations to show anti-war messages [1].

When combining the insecurities of the HbbTV protocol with tracking and data analytics, users' privacy is at greater risk: TV viewing behavior provides "very detailed and sensitive insights into what users think, know and believe" [144]. A new form of advertisements

(ads) aims to mine this data and take advantage of the dynamic content delivery through HbbTV: *Addressable TV (ATV)*. With ATV, the static delivery of ads over the standard broadcast signal is expected to be replaced by dynamic ad insertion, i.e., instead of all viewers of a TV program seeing the same ads during commercial breaks, targeted ads can be delivered over the Internet [163, 274]. Thus, ATV might increase privacy risks for TV consumers. Several companies have recently developed new solutions to collect and analyze data for ATV, ranging from Smartclip, over Equifax, to Castoola [54, 89, 163]. Furthermore, the numerous players in the Smart TV ecosystem and the scattered and outdated legal framework create additional complexity in data handling [92].

Some solutions to protect users' privacy in the Smart TV domain are already available, such as DNS blocking of tracking domains [110, 183, 310]. However, they are not sufficient, since they either block all incoming traffic, or rely on incomplete and, thus, ineffective denylists. In general, most solutions are not yet designed to address TV tracking. Furthermore, the (Smart) TV landscape is rapidly changing. The HbbTV protocol itself is improving with a shift towards HbbTV 2.0 with new security measures (e.g., increased use of HTTPS over HTTP). Thus, in this paper, we revisit and extend previous studies on the privacy posture of HbbTV in light of protocol changes and increased adoption. We start by studying how the TV landscape has changed after the more widespread adoption of HbbTV to see if broadcasters take better care of users' privacy by looking at the traffic between a Smart TV and the servers offering HbbTV applications. In particular, we focus on Italy, Germany, France, Austria, and Finland, countries that (1) are actively adopting the HbbTV [121] and (2) are known for strong awareness and support for privacy and data protection regulations [73, 95]. In these countries, we perform active measurements on 36 different TV channels using both Smart TVs and off-device protocol inspection to investigate privacy aspects of the channels' HbbTV implementation in 2021 and 2022. We complement these experiments by studying whether users have become more security and privacy aware after being more frequently exposed to HbbTV applications.

In summary, our main contributions are as follows:

- We show little progress in the protection of data sent between broadcasters and Smart TVs over the years: users are still exposed to privacy risks, such as tracking before expressing consent and the transmission of credit card details through insecure plaintext traffic.
- We investigate users' privacy and security awareness, showing a general lack of knowledge of HbbTV's risks. We also show great concern for their data when confronted with risks linked to Smart TVs and HbbTV.
- Based on the results of our technical and qualitative analysis, we design an "HbbTV Blocker" to intercept and block unwanted traffic from the Smart TVs to the broadcasters (and vice versa).

Responsible Disclosure. We put particular care into identifying problems concerning users' privacy in light of current legislation: (1) the *General Data Protection Regulation (GDPR)*, which has been effective since 2018 to protect European citizens' data and privacy [93], and, given the geographical setting of our analysis, (2) the *Garante per la Protezione dei Dati Personali (GPDP)* [108], the Italian administrative authority for data protection (see Section 5.8).

We are working with the GPDP and the Computer Emergency Response Team Austria (CERT.at) [56] to responsibly disclose the issues found to the broadcasters. We will provide updates on the outcome of this process in the repository below.

Artifacts. The source code of our experiment setup, the HbbTV Blocker, and the collected denylist of tracking domains are available at https://github.com/SecPriv/hbbtv-blocker.

5.2 Background

5.2.1 Hybrid Broadcast Broadband TV (HbbTV)

HbbTV Specification. HbbTV, as stated by the HbbTV Association [122], is "... a global initiative aimed at harmonizing the broadcast and broadband delivery of entertainment services to consumers through connected TVs, set-top boxes, and multiscreen devices." In other words, it represents both a widely adopted standard (the ETSI Technical Specification 102 796 [94]), and a driving force to promote a unified hybrid TV delivery across different platforms [40] offering broadcast and broadband content to viewers. The initiative dates back to 2009 when a group of industry leaders, led by the German broadcaster RTL, introduced a different form of Teletext using HbbTV and the CE-HTML interface language, an XHTML-based standard for websites with remote user interfaces typically used in consumer electronic devices.

The HbbTV standard works either via broadcast or via IP link; however, it is most powerful in an Internet-connected environment where a combination of broadcast and broadband networking can deliver additional content to the user. For HbbTV to work, the TV must support it, and then the broadcaster must provide at least one HbbTV application for the user to interact with. When such an application is delivered to the user, they are typically informed that some extra HbbTV content is available with a relevant icon. Such additional information can be in the form of program guides, viewer interaction (e.g., with quizzes during a show), lyrics of music videos, additional advertising, and customized content.

To interact with such extra content, until HbbTV version 1.5, the user could use the Smart TV remote control, more specifically through the colored buttons. Instead, HbbTV 2.0 offers the possibility of connecting different devices, such as smartphones and tablets, allowing multi-device interactions.

Unlike the Internet Protocol TV (IPTV), HbbTV still relies on the standard broadcast signal to deliver the initial application URL and launch it as an overlay on top of standard TV. It does not represent an alternative way to receive content but rather an enhanced version. Additionally, HbbTV, except for premium features, is free for the user; instead, to benefit from IPTV, users typically have to pay a subscription fee. Those two considerations combined make HbbTV more appealing to advertisers and trackers who can reach a greater audience.

HbbTV Communication. A Smart TV (i.e., a hybrid terminal) can support two different connections in parallel; on one side, it is connected to a broadcast Digital Video Broadcasting (DVB) network. On the other side, it is connected to the Internet via a broadband interface. The TV receives standard broadcast Audio/Video (A/V) content through the first one and allows for the signaling of stream events to an application. The Internet connection allows for bi-directional communication with the provider and can receive non-linear A/V content. The broadband interface may also connect with other HbbTV terminals or "Companion Screen Devices" (e.g., smartphones and tablets) on the same local network. We show the interaction between the different actors in Figure 5.1.

Through the Broadcast interface, the terminal also receives application data and stream events that are transferred using Digital Storage Media - Command and Control (DSM-CC) objects. Non-realtime content is transmitted using the File Delivery Protocol (FDP) protocol. The data is sent to the Runtime Environment of the terminal composed of the Application Manager, the Browser, and the Companion Screen Interface. Via the Broadband interface, the Smart TV also connects to the Internet. This connection provides a way to request application data from a provider's servers. Data collected in this way is again transferred to the Runtime Environment [120].

The TV as a Browser. The Internet-delivered HbbTV applications are embedded as URLs in the DVB stream sent by broadcasters. The Internet Protocol Processing component parses the data from the Internet and passes the information to the Runtime Environment. This environment includes the TV's browser, responsible for presenting and executing the application. Any website written with standard web techniques (e.g., HTML, CSS, JavaScript) can serve content. When the application is loaded, the browser displays a notification overlay to the user showing that the application is ready to be activated through the remote control (via the standard *Red Button*).

5.2.2 Security & Privacy Concerns with HbbTV

Despite statistics showing an ever-increasing adoption of HbbTV, little or no literature on its security and privacy issues is available. The main focus of researchers has been vulnerabilities linked to physical access to such devices either through USB port or local network [124, 255, 271]. However, as mentioned above, Smart TVs that support HbbTV can access online content and websites through the integrated web browser, opening up many different attacks. Before delving into the details of potential attacks, it is worth mentioning that the HbbTV specification presents a security-related chapter. It states

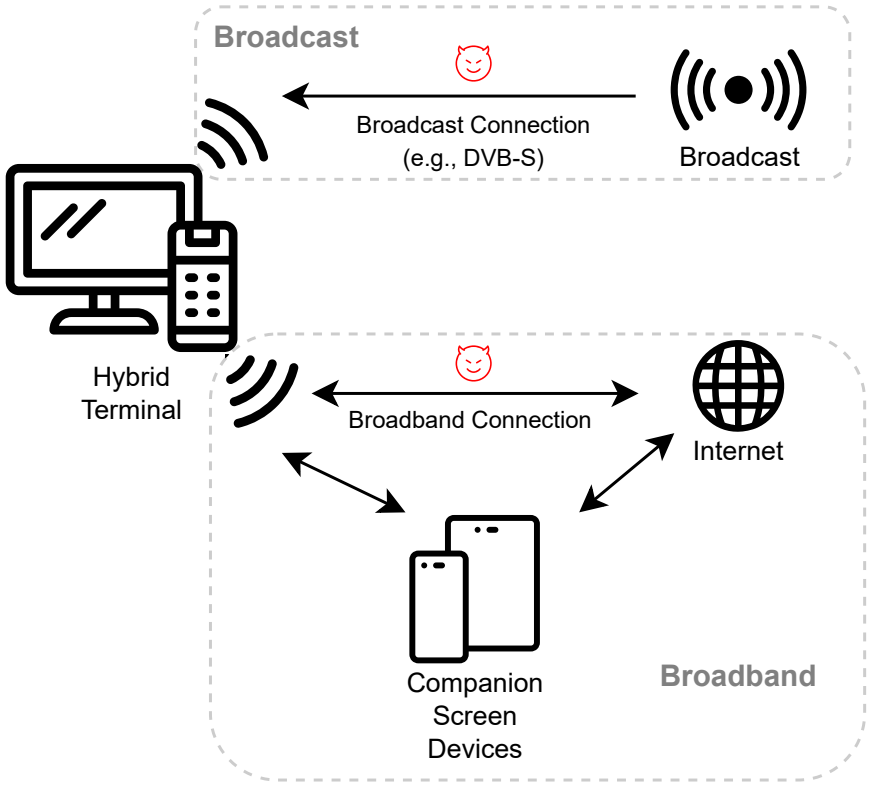


Figure 5.1: HbbTV System Overview: The hybrid terminal represents a Smart TV with dual connection (i.e., broadband and broadcast). We highlight the attack channels we focus on during the remainder of our paper.

that the user shall trust only broadcast-related applications, not broadcast-independent ones. However, it does not mention how to perform such a control, and additionally, the user can bypass such a restriction by making broadcast-independent applications trusted.

Encryption and Certificates. On the broadband side, the HbbTV specification mentions that security is provided by adopting the Transport Layer Security (TLS) protocols. The standard presents several requirements concerning the adoption of TLS, such as the supported cipher suites, the minimal key length, and the forbidden use of compression algorithms. At the same time, adopting TLS (or, more specifically, HTTP over TLS (HTTPS)) is strongly suggested; whether it is implemented depends on device manufacturers and the actors delivering content to the user [120].

Privacy and Tracking. The standard allows the user to specify their tracking policy by choosing between two alternatives: Do Not Track (`DNT`) set to 1, i.e., no tracking consent, or set to 0, i.e., tracking consent. To explicit user's tracking preferences, the `DNT` parameter is included in every outgoing HTTP request. However, again, in this case, it is up to application developers and device manufacturers to correctly implement

this. Several problems might arise if tracking websites are allowed, especially in autostart applications (i.e., applications that run without the user knowing and without the need for their consent) or even if persistent cookies are stored. Persistent cookies remain until the expiry date and, as reported in Section 5.3, can be highly problematic since the date set is far in time, allowing tracking over a long period [120].

As previously described, most HbbTV apps run in a built-in browser that displays HTML content and runs JavaScript code. The DVB stream contains the HbbTV URL (retrieved from specific web servers) that is opened in the browser and shown as a semi-transparent HTML layer that overlaps the actual TV program. In such a way, the TV becomes visible to the broadcaster even before the user consents to it, possibly breaching their privacy. We report more details in Section 5.3.

As with other web content delivered to desktop and mobile browsers, third-party tracking represents a problem in this scenario. A study conducted in 2013 over 66 different German stations showed that 13 among them used Google Analytics to track users [124]. This could not only impact users' privacy, but an attacker can exploit such a feature to spam fake analytics via proxy networks simulating actual TVs and influence broadcasters' decisions, e.g., to discontinue a specific show.

5.2.3 Threat Model

In our threat model, the Smart TV is connected to the Internet. The broadcaster communicates with the device over traditional broadcast and broadband signals. We identify two potential threat actors. The first is an ill-intentioned broadcaster that tracks users' preferences and viewing times without asking for their consent via cookies and by sending profiling data over HbbTV. This data can later show targeted and personalized advertisement mining users' autonomy. More generally, the broadcaster and any party providing HbbTV content, including advertisers and trackers, can compromise users' privacy on purpose or through misconfiguration and negligence. The second threat actor is represented by anyone with access to the traffic generated by the TV (e.g., an Internet provider or an actor connected to the same network) that can intercept unencrypted HTTP traffic, possibly sniffing sensitive information such as usernames and passwords.

5.3 Related Work

Smart TV and IoT Security. Weak certificate validation is a known issue with Smart TVs. Paracha et al. show that most IoT devices (including Smart TVs) implement TLS incorrectly, using deprecated ciphers, not correctly validating certificates, or containing deprecated root certificates [230]. Aafer et al. assessed the security level of Android TV boxes by fuzzing target vendors' APIs, finding 37 unique vulnerabilities [2]. Bachy et al. showed the feasibility of attacks via local loops supporting the ADSL network and DVB, finding several security vulnerabilities. They also report different techniques for extracting and analyzing the firmware of Smart TVs [30]. Moghaddam et al. investigated the privacy

of two OTT devices, Amazon Fire TV and Roku TV, showing that, respectively, 89% and 69% of the top 1,000 viewed channels for each platform contact at least one tracking domain even when the user explicitly selects the enhanced privacy feature [193]. In addition, an attacker can use the Smart TV as a starting point to access the user's private WiFi network. Barre et al. demonstrated the feasibility of using the Smart TV as a relay to attack further devices [36]. Acar et al. showed how machine learning techniques could help identify IoT devices (potentially including Smart TVs) only by passively listening to network traffic [3]. Puche Rondon et al. investigated Enterprise Internet-of-Things (E-IoT) showing that despite its "secure" reputation, several issues are present mining users' privacy and security [257] and also investigated one High Definition Multimedia Interface (HDMI) component, the Consumer Electronics Control (CEC) protocol, that takes an important role in receiving A/V content [256]. Instead of studying issues related to specific Smart TVs and local connections, our work focuses on issues in the HbbTV content delivery protocol used by broadcasters.

HbbTV Security and Privacy. Several proof-of-concept attacks showed HbbTV's susceptibility to content injection. In 2014, Oren and Keromytis manipulated an HbbTV URL at the DVB level, causing several devices to receive malicious content [226]. An attacker can exploit DVB/DSM-CC injection to replace content into streams, directly specifying the URLs pointing back to their malicious content. A recent example of this is the injection of anti-war messages in Russian TV programs [1]. Cabrera [50] showed an attack scenario using drones to replace the legit broadcast signal with crafted streams. In 2019, Massimo Bozza showed the feasibility and extreme easiness of hijacking HbbTV DVB connections through the use of the HiDes UT-100c, a modulator (transmitter)[125] and the C++ library TSDuck [164]. This weakness in the DVB architecture allows an attacker to perform a Monkey-in-the-Middle (MITM) attack replacing the original content of the HbbTV application with arbitrary and/or malicious content, such as fake news banners to spread misinformation, redirections to a malware-download website, or scam/phishing sites [48]. Similarly, Michéle et al. performed such an attack using a Terratec TStick+ as modulator and, on the software side, different libraries such as *tzap* [188]. Users might be tricked into clicking a malicious link, and JavaScript code can be run without the user's knowledge. For example, attackers can exploit TVs' CPUs to mine cryptocurrency using JavaScript-based code [61, 155, 259].

Ghiglieri and Waidner [112] conducted three different tests in 2012, 2014, and 2015 to analyze the HbbTV data flow from the Smart TVs to the broadcasters and vice versa, finding several privacy issues. A German channel transferred a user's login without HTTPS, thus allowing potential attackers to record the complete login process and later exploit it. In 2015, many channels switched to HTTPS for securing HbbTV applications indicating that some steps towards security and privacy maturity were being made. Despite this progress, no governing rules were still present, and such technology did not completely enforce users' privacy [188]. We revisit and extend this study to assess the current state of HbbTV across Europe.

Anti-Tracking and Security Solutions. Varmarken et al. assessed the (in)effectiveness of existing Smart TV DNS denylists showing that they do not successfully block all tracking [310]. Ghiglieri et al. proposed the Privacy Protector that allows users to control their data by barring channels from loading Internet data unless the user presses the *Green Button* on the TV's remote [110]. Mandalari et al. proposed a solution for generic IoT devices that could also be used for Smart TVs. Their idea is to flag domains as either "essential" or "non-essential" for the correct functioning of the device and block the latter [179]. Matejka et al. proposed a Security Manager [183] to ensure a secure authentication and authorization mechanism for users. Such an approach should verify that users are who they claim to be and later enforce some access control policies.

Users' Risk Awareness. A survey by Ghiglieri et al. [111] in Germany revealed that only a small percentage of respondents know of privacy and security risks, and even fewer can mention a concrete consequence. At the same time, when confronted with the risks, almost no one is willing to fully disconnect their device from the Internet. Malkin et al. conducted a similar survey on the risks of Smart TVs in the U.S., showing that participants are confused about what data are collected and how and by whom they are analyzed [177]. Nevertheless, respondents considered it unacceptable that their data is being repurposed and expected manufacturers to protect their data.

5.4 HbbTV Protocol Testing

To shed light onto current privacy issues introduced with the ongoing deployment of HbbTV in Europe, we start by describing our testing environment that allows us to intercept and analyze the content delivered by any TV channels supporting HbbTV. We present the issues we uncovered when studying 36 TV channels in Italy, Germany, France, Austria, and Finland using this environment in Section 5.5.

Methodology. Our testing environment has two components, as shown in Figure 5.2:

- *On-TV HTTP(S) Traffic Capture:* Listen and capture the traffic between the Smart TV and the servers to later analyze what domains are contacted and search for cookies and users' data (see Section 5.4.1).
- *Off-TV HTTPS Traffic Inspection:* Extract the URLs contacted by the Smart TV to launch the HbbTV app [48] and open them in a Chrome browser while a transparent proxy listens (see Section 5.4.2).

Environment. We use three main devices for our experiments: a Sharp Aquos LC-32Bi6, a Xiaomi Mi 4A Smart TV (Android 9), and a Samsung M5500 Smart TV (Tizen 3.0). In addition, we connect a laptop running Ubuntu 20.04 and Wireshark [322] through the Ethernet interface with the home router, and we enable its WiFi hotspot. We then connect the Smart TV to the WiFi network of the laptop.

Technically one Smart TV should suffice. However, two channels (see Section 5.5.1; i.e., Mediaset and La7) did not receive any HbbTV application on the Android device, probably because of compatibility issues. Thus, we analyzed these channels on a Samsung device (where the HbbTV application is available and usable). Note that our off-TV testing approach does not require access to a Smart TV at all—as long as the targeted HbbTV URLs are known.

5.4.1 On-TV HTTP(S) Traffic Capture

We record traffic for one hour in four phases (timings are the same adopted by Ghiglieri [110] to foster comparability):

1. Listen for 15 minutes without any interaction to spot information transmitted before user consent or explicit user action to enable the HbbTV application.
2. Give consent and interact for 20 minutes with the suggested buttons, different for each channel, to see what data is sent and if we spot HTTP connections in the extra features offered by the HbbTV application.
3. Revoke user consent (if possible) and listen for 10 minutes without interaction.
4. Restore consent, change the channel, re-tune back and listen for 15 minutes without any interaction.

We perform a factory reset for each channel analysis to prevent interference in the traffic.

We automate traffic analysis for the Android device, with a bash script to precisely time the phases above. To make the analysis process faster and more efficient, using Tshark [321], the command line version of Wireshark, we convert the *.pcap* file into a *.csv* file and extract the following information for HTTP(S) packets: *domain* (the contacted host), *occurrences* (the number of requests to a specific host), and *consent_status* (tags identifying the four periods the testing phase has been divided into).

Additionally, we manually label each contacted service based on its intent and common usage (e.g., *Tracking*) by using publicly available online sources for each domain, e.g., we search for the organization owning the domain and analyze the main solutions offered by it. We exclude from our labeling domains that are contacted by the Smart TV itself and not because of the HbbTV application. Thus, we remove domains that are present in more than half of the traffic captures and that include the Smart TV vendor in the domain name, e.g., data.mistat.intl.xiaomi.com. Additionally, for unencrypted packets, i.e., using HTTP, we inspect them, looking for cookies, parameters, and API calls.

All the domains that we labeled as *Tracking* can signal misbehavior of the provider, i.e., not waiting for explicit user's consent before delivering tracking and targeted content—depending on whether we observed it either before the consent is given or after it is revoked. At the same time, even the non-tracking domains found before accepting the

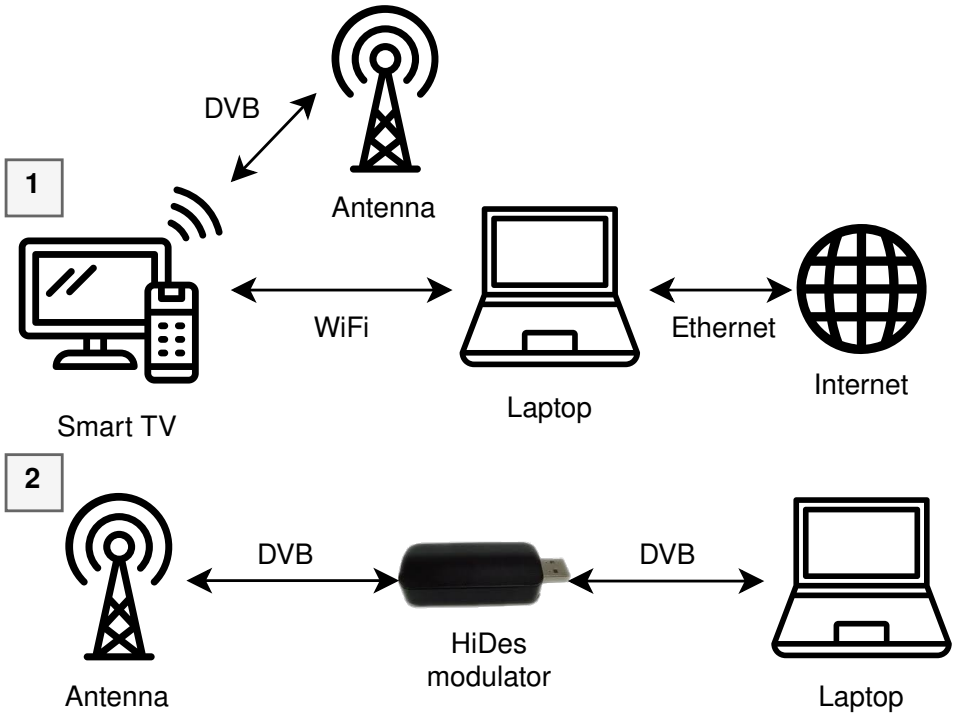


Figure 5.2: Overview of our Testing Environment: (1) *On-TV HTTP(S) Traffic Capture* and (2) *Off-TV HTTPS Traffic Inspection*.

privacy notice are not in line with the HbbTV protocol, as no communication should occur unless the user agrees.

Furthermore, we analyze the timing between the requests to show whether there is a particular frequency of requests by computing the average time between requests to the same domain. The standard deviation shows if such a pattern is reliable (i.e., a high standard deviation means that the time window between requests varies significantly; therefore, we cannot identify a specific pattern).

5.4.2 Off-TV HTTPS Traffic Inspection

For the second test, we start by extracting the HbbTV URLs from the DVB stream using the TSDuck library and the UT-100c HiDes modulator. As mentioned in Section 5.2, the DVB stream includes the URLs of the HbbTV applications; thus, by analyzing the broadcast stream, we can extract them directly.

We perform a complete scan of the Ultra High Frequency (UHF) channels to identify the frequency of the channels to analyze using the `tsscan` function of TSDuck. Then, with `tsp`, we can capture the DVB stream on a specific frequency passing through the HiDes modulator in Transport Stream (TS) format. We then convert the file into a *.txt* format, and we extract the segments related to the Application Information Table (AIT), which contains the HbbTV information together with the start-up URLs of the applications, by

specifying their respective Program IDs (PIDs). The `application_type` parameter should equal `0x0010`, meaning that the information is related to HbbTV.

We then open the extracted URLs in a Chrome desktop browser (version 91.0.4459.2), deleting the browser's data and cookies before each test to reset it, and following two approaches to mimic the Smart TV environment since some of the analyzed links can be opened only in one of the settings:

- *Browser Extension:* We emulate a Smart TV's browser with a Chrome extension (RedOrbit HbbTV Emulator [249]) that recognizes and renders `application/vnd.hbbtv.xhtml+xml` content.
- *Spoofed User Agent:* We change the User-Agent (UA) to one of a real Smart TV, e.g., `HbbTV/1.4.1 (+DRM+MEDIA360;Samsung;Smart TV2017;T-KTSDEUC-1290.3;;)+TVPLUS+ Smart HubLink Chrome`.

To collect traffic, we set up a transparent proxy using mitmproxy [69] on the same machine where we opened the extracted links to intercept HTTPS traffic and capture its plaintext. We manually inspect these files for analysis using the mitmdump component of mitmproxy. We perform this second test to bypass our limitations when analyzing encrypted traffic in Section 5.4.1. One would expect the collected traffic to be similar in the two test setups. However, since this latter approach might alter the communication between the Smart TV and content providers, we deem the first test necessary for completeness. In addition, the domains contacted with this second test are for sure generated by interacting with the HbbTV application; thus, we can better filter out domains contained in the *On-TV* test that are not related to HbbTV but contacted by the Smart TV itself. Furthermore, this second setup provides better scalability and reproducibility, allowing anyone without access to a Smart TV to test selected HbbTV applications.

The methodology resembles our approach in Section 5.4.1, but we capture only 30 minutes of traffic: listen for 10 minutes without interaction, accept the privacy notice and interact for 10 minutes, revoke consent, and listen for 10 minutes. We extract the contacted domains and IPs with the known purpose of the service and tracking cookies.

5.5 HbbTV Protocol Issues in the Wild

5.5.1 Selection of TV Channels

Table 5.1 summarizes the selected channels, whether public or private, and their national audience share (if available). We performed a first round of tests between February and May 2021 in European countries where the selected channels are available. We conducted a second round of tests in November 2022 and marked the new channels accordingly. In 2022, we executed the *On-TV* test for 30 minutes more to spot potentially missed trackers (1h30). As reported in Table 5.1, for some channels, we performed the *Off-TV* test only

Table 5.1: Selected TV channels, their target country, whether the broadcaster is public or private, and their audience share [4, 27, 28, 258, 287, 288, 289, 290, 291]; ● signifies we both performed *On-TV* and *Off-TV* traffic inspection, ◐ means only the former and ◑ only the latter. In addition, we mark the channels we only tested in 2022 with *.

	Country	Private/Public	Audience Share	Tests
Rai1	IT	Public	11,54%	●
RDS	IT	Private	-	●
RealTime	IT	Private	1.31%	●
SportItalia	IT	Private	0.22%	●
RTL	IT	Private	0,17%	●
Spike	IT	Private	-	●
Canale 5	IT	Private	15,96%	●
La7	IT	Private	3,11%	●
Radio Kiss Kiss	IT	Private	-	●
Radio Libertà*	IT	Private	-	●
BOM Channel*	IT	Private	-	●
NOVE*	IT	Private	1.92%	●
Caccia e Pesca*	IT	Private	-	◑
QVC*	IT	Private	-	◑
TeleNordEst*	IT	Private	-	◑
TeleChiara*	IT	Private	-	◑
SuperTennis*	IT	Private	0.25%	◑
LineaGem*	IT	Private	-	◑
Warner TV*	IT	Private	-	◑
TV 8*	IT	Private	2.36%	◑
HSE	DE	Private	-	◐
SWR BW	DE	Public	-	◐
Arte	DE	Public	1.3%	◐
ZDF	DE	Public	14.2%	◐
Anixe	DE	Private	-	◐
RTL*	DE	Private	7.3%	◐
Das Erste*	DE	Public	11.9%	◐
Sport1*	DE	Private	0.6%	◐
Arte	FR	Public	1.3%	◐
NRJ12	FR	Private	1.2%	◐
ATV*	AT	Private	3.3%	●
ORF1*	AT	Public	8.4%	●
Servus TV*	AT	Private	4.7%	●
SchauTV*	AT	Private	-	●
MTV3*	FI	Private	17.7%	◐
Yle TV1*	FI	Public	27%	◐

for 10 minutes (5 without consent to the data policy, 5 with interaction), allowing us to respect our time constraints while fostering reproducibility and scalability.

The Italian channels we selected belong to different broadcasters, either public or private. We chose these based on average audience share, broadcaster, offered content, and, of course, enabled support for HbbTV. For further comparison, we replicated the traffic

analysis in Germany, France, Austria and Finland to reveal differences or similarities in adoption of the HbbTV protocol. For these, the procedure is similar but simpler. We carried out only the *On-TV* test (see Section 5.4.1), and we analyzed 30 and 90 minutes of traffic, in 2021 and 2022 respectively. For Germany, five of the total eight channels are a subset of the ones studied by Ghiglieri and Tews [110]: Arte, Anixe, SWR BW, HSE Live, and ZDF. Ghiglieri and Tews divided the channels they analyzed into four groups based on privacy invasiveness. For our study, we selected one representative channel from each group to see how the situation changed over five years. We selected HSE (a shopping channel) considering the gravity of security issues researchers found, e.g., handling credit card details over plain HTTP. In France, Austria, and Finland, the adoption of HbbTV is not yet in full swing, and few channels can interact with such applications. We report the channels we considered in Table 5.1.

5.5.2 Privacy Related Findings

Italy. We report the extracted HbbTV start links for the Italian TV channels we use for the in-depth test, including HTTPS interception, in Table 5.2. Both tests (*On-TV* and *Off-TV*) aim to show what information is exchanged between the Smart TV and the broadcasters to investigate potential privacy risks. We thus aggregate and then present them in the following section.

Fifteen out of 20 Italian channels show connections to at least one tracking service (with possible profiling cookies) before the user can decide whether to accept the privacy notice. Table 5.3 summarizes which channels connect to which tracking service before users' consent, including different Google tracking and analytics services. In addition, three channels make POST requests to an Amazon Web Services (AWS) API /audiencesavemessage with the user ID, model, and device brand as parameters for profiling.

In general, cookies have long expiration dates ranging from 2021 to 2048. Such cookies can track users' behavior with potential linkage to other data; therefore, such persistence poses privacy risks. The situation improved in 2022, as the longest expiration date we found was December 2023. Additionally, given the adoption of HTTP by some services, such cookies are sent in plaintext. If an attacker is sniffing the communication channel over which the information is sent, they can intercept it. For example, RealTime sends plaintext cookies identifying the user's geographical location and Internet Service Provider (ISP) and sets Google Analytics ones even before their explicit consent.

Six channels do not present any privacy policy when accessing the HbbTV application for the first time, and the user starts to be profiled without having provided consent. Such a policy is nowhere to be found even in the sub-menus of the app. On the other hand, since 2021, two channels, RDS and Rai, that did not show any policy, now prompt the user with a privacy banner showing some progress.

Some channels offer the possibility to revoke the consent to data processing, while others do not. Since deleting cookies on the Smart TV is not trivial (in fact, such a procedure requires a factory reset) and given the cookies' long expiration dates, it is fair to provide

Table 5.2: HbbTV start URLs for 20 Italian channels.

Channel	Start Link (URLs)
Sportitalia	(http) www.sportitalia.kbbtv.tech/hbbtv/sportitalia/sportitaliachannel/index.html
RDS	(http) hbbtv.rds.radio
RealTime	(http) discovery.castoola.tv/realtime
RTL	(https) cloud.rtl.it/hbbtv.rtl.it/rtlchannel/index.html
Rai 1	(https) www.raiplay.it/hbbtv/launcher/RemoteControl/index.html?delivery=2 (https) www.raiplay.it/hbbtv/RaiPlay2020/index.html
Spike	(http) www.kbbtv.tech/viacom/viacomchannel/index.html
Canale 5	(http) hbbtv.mediaset.net/app/mplayhbbtvgold/backdoor.shtml (http) hbbtv.mediaset.net/app/mplayhbbtvgoldzoo/dev/index.html (https) mhptivu.mediaset.net/app/mplayhbbtvtivu/index.html (https) tivuon-hbbtv-lativu.tivu-alchemy.net/index.html?configuration=prod
La7	(https) ht.la7.it/index.php
RadioKissKiss	(http) www.kisskiss.kbbtv.tech/hbbtv/kisskisschannel/index.html
Radio Libertà	(https) hbbtv.persidera.it/hbbtv/jump/index.html?channelId=38893
BOM Channel	(http) 95.110.225.170/hbbtv_bootstrap/index.php
NOVE	(http) discovery.castoola.tv/nove
Caccia e Pesca	(https) app.cacciaepesca.tv/hbbtv-cp/
QVC	(http) qvc-italy-hbbtv-app.qvc-italy.c.nmdn.net/redbutton
TeleNordEst	(http) hbbtv.tdbnet.it/run.php?pid=2049
TeleChiara	(http) iphd.it/hbbtv/telechiara/index.php
SuperTennis	(https) hbbtv.persidera.it/hbbtv/launcher/index.html?appId=25970
LineaGem	(http) www.grupposciscione.kbbtv.tech/hbbtv/lineagem/index.php
Warner TV	(http) it.container.enhanced.live/warnertv/
TV 8	(https) data-hip-gcdn-skycdn-it.akamaized.net/iapp/produzione/hbbtv/Addressable/index.html

the user with the possibility to revoke the consent to data processing. Specifically, ten channels out of the 20 we examined do not allow users to withdraw consent.

Moreover, despite presenting the possibility of revoking consent, the channel RTL, in reality, does not delete profiling and identification cookies, and requests to any tracking services are still made even *after the user revoked their consent*.

We further found that a widely used tracking technique by broadcasters is the *tracking pixel* [104], i.e., tracking user behavior by uploading a 1×1 pixel image when the user visits a website or opens a particular content. Given its small size, it is invisible to the naked eye. Still, it can provide valuable data to advertising or analytics companies that can infer user preferences in this way. In particular, seven channels adopt this technique by returning 1×1 pixel GIF89a objects in requests.

Finally, similar to the results reported by Ghiglieri et al. [112], some channels perform periodic requests to check if the user is still watching. Almost all channels show frequent requests to profiling domains (on average, around every minute); for example, SportItalia makes requests to Smartclip (an advertisement broker in the Addressable TV ecosystem) around every 70 seconds, while RDS contacts Google Analytics around every 14 seconds. A low standard deviation indicates that we identified a recurrent pattern.

Table 5.3: Summary of Results: (1) tracking services contacted before consent, (2) presence of a privacy notice, (3) if revoking consent was possible, (4) tracking pixels, (5) periodic requests to tracking domains and (6) plain HTTP use. ● = "Yes", ○ = "No", ◐ = "Partially". The latter applies when we could only partially revoke consent (e.g., disable only tracking pixels). We mark the channels we only tested in 2022 with a *.

	Channel Name	Tracking Services	Privacy Policy	Revoke Consent	Tracking Pixel	Periodic Requests	HTTP
IT	**SportItalia**	DoubleClick POST to /audiencesavemessage (AWS) ip-api	●	●	○	●	●
	RDS	-	●	◐	○	○	○
	RealTime	Google Analytics Google Tag Services discovery-log-view.castoola.tv atv-discovery-microservices.castoola.tv	●	○	●	●	●
	RTL	-	●	◐	○	●	○
	Rai 1	Scorecard Research	●	◐	●	●	○
	Spike	Google Tag Services SecurePubAds	●	●	●	○	●
	Canale 5	tags.tiqcdn.com (Tealium Inc.)	●	●	○	●	●
	La7	tags.tiqcdn.com (Tealium Inc.) DoubleClick cdn.permutive.app smetrics.rcsmetrics.it	●	◐	●	●	○
	Radio Kiss Kiss	POST to /audiencesavemessage (AWS)	●	○	○	●	●
	BOM Channel*	-	○	○	○	○	●
	Radio Libertà*	analytics.persidera.it	○	○	○	○	○
	NOVE*	Google Analytics Google Tag Services	●	○	●	●	●
	Caccia e Pesca*	tags.tiqcdn.com (Tealium Inc.) smetrics.rcsmetrics.it components2.rcsobjects.it/rcs_tracking-service	○	○	●	○	○
	QVC*	INFOnline GmbH	●	●	●	●	●
	TeleNordEst*	-	●	○	○	○	○
	TeleChiara*	DoubleClick	○	○	○	○	●
	SuperTennis*	analytics.persidera.it	○	○	○	●	○
	LineaGem*	DoubleClick POST to /audiencesavemessage (AWS) ip-api	●	●	○	●	●
	Warner TV*	-	●	◐	○	●	●
	TV 8*	DoubleClick Google Tag Services	○	○	○	○	○
DE	**HSE**	hse24.tvtelemetrie.de	●	●	●	○	●
	SWR BW	-	●	◐	●	●	●
	Arte	XiTi by AT Internet	●	◐	●	○	●
	ZDF	XiTi by AT Internet	●	◐	●	●	●
	Anixe	Google Analytics	●	●	●	●	●
	Das Erste*	-	●	●	●	●	●
	RTL*	tvping.com Smartclip nmrodam.com by Nielsen	●	●	●	●	●
	Sport1*	Smartclip tvping.com	●	●	●	●	●
FR	**Arte**	XiTi by AT Internet Médiamétrie	●	●	●	●	●
	NTJ12	DoubleClick mediarithmics XiTi by AT Internet	●	○	○	●	○
AT	**ATV***	tvping.com	●	◐	●	●	●
	ORF1*	tvping.com	●	◐	●	●	●
	Servus TV*	Google Tag Services tvping.com	●	○	●	●	●
	SchauTV*	Smartclip tvping.com	●	●	●	●	●
FI	**MTV3**	-	○	○	○	○	○
	Yle TV1	-	●	◐	○	●	○

Germany. All five channels present privacy policies. However, just two of them, Arte and HSE, show the privacy policy as soon as the user opens the respective channel. In the other cases, we have to search for it in the sub-menu of the HbbTV app or click the blue button. All channels offer the user the possibility to revoke their consent. Still, SWR, Arte, and ZDF offer the user only the option to disable the tracking pixels.

All five channels adopt tracking pixels. In addition, in 2021, we observed more "in-house" tracking (using smaller and/or local companies), unlike the Italian channels that rely on larger services such as Google Analytics or Smartclip to collect user information. This phenomenon faded in 2022 as we noticed the use of larger services also by the German channels. By nature, larger third-party services, such as Google, have the means to aggregate more user data, e.g., with information collected from websites and mobile apps. This opens the potential for more targeted content and advertising, possibly across a user's devices (not limited to the Smart TV).

We also found greater use of HTTP, i.e., unencrypted traffic, than we observed in Italy. This leads to serious security issues: two channels, Arte and HSE, allow users to log in using their credentials linked to an account holding sensitive information, such as their address and credit card information. The credentials are sent in plaintext, allowing an attacker to intercept them. This issue was already explicitly reported in a previous paper for HSE in 2014, but is still not fixed [110].

Overall, we observe that the security and privacy posture of German TV channels, unfortunately, has not evolved much from what Ghiglieri et al. already observed in 2015.

France. On a positive note, both channels show the privacy policy before interacting with the app and allow the withdrawal of consent. Conversely, as in Germany, Arte supports the aforementioned tracking pixel. We also observe greater use of smaller/local tracking companies. As in Germany, Arte offers the possibility of logging in, but sends the authentication code in plaintext, thus allowing attackers to intercept it.

Austria. All four channels adopt the tracking pixel; in particular, track.tvping.com, which returns a 1×1 pixel PNG, is contacted by all of them with a request *every second* even before consent is given. Despite all the channels presenting the privacy policy, this is shown only once the user starts interacting with the HbbTV application. Finally, although Servus TV presents the option to revoke cookies, this functionality is not correctly implemented, and tracking cookies are always on.

Finland. For Finland, we observed the absence of a privacy policy and the option to revoke consent for the MTV3 channel. In addition, both channels uniquely rely on HTTPS for communication and do not adopt the tracking pixel.

5.5.3 Identified Trackers

During our analysis, we identified (i.e., manually attributed) 70 domains to advertisement and tracking services in 2021, as well as 43 in 2022 (19 domains are shared between both

Table 5.4: The fraction of tracking domains we found in our HbbTV analysis (70 in 2021 and 43 in 2022) included in widely used tracking denylists. We examine Pi-hole [128] (version of May 23, 2022) and EasyList [84], a popular set of rules for Adblock (versions of May 23, 2022, and March/April 2021).

	Configuration	Total Rules/ Domains	2021 Blocked (out of 70)	2022 Blocked (out of 43)
Pi-hole	Default [41]	115,065	31 (44.29%)	35 (81.39%)
	SmartTV [235]	167	1 (1.43%)	2 (4.65%)
	Samsung SmartTV [316]	73	0 (0%)	0 (0%)
	SmartTV2 [126]	213	2 (2.86%)	6 (13.95%)
EasyList	March/April 2021	57,276	5 (7.14%)	4 (9.30%)
	Privacy – March/April 2021	22,136	3 (4.29%)	3 (6.98%)
	May 2022	60,680	11 (15.71%)	5 (11.63%)
	Privacy – May 2022	26,939	7 (10.0%)	6 (13.95%)

years). In Table 5.4, we report the number of tracking domains we found during our analysis and whether they are contained in (and thus could be blocked by) currently available denylists. Similar to related work on web tracking, our results show limited coverage of these lists [104]: in the best case, 31 out of 70 (44.29%) of the tracking domains are known and thus can be blocked. This "best-case scenario" is Pi-hole [128], a network-level ad and tracker blocking application based on DNS sinkholes. It is designed to block general web trackers, mobile app trackers, as well as trackers part of the firmware and applications installed on certain Smart TV models.

When comparing the results from March/April 2021 (when we performed our network captures) with the most current EasyList as of May 2022, we observe a slight increase in the number of included domains (5 to 11 in the general version and 3 to 7 in the Privacy version). This suggests that filter lists are catching up to new players in the Smart TV advertising ecosystem, but overall coverage remains low.

5.6 Users' Risk Awareness

Our findings, presented in Section 5.5 on a sample of 36 TV channels, highlighted how broadcasters do not always protect users' privacy. However, the adoption of the HbbTV protocol by broadcasters, as well as the integration of new, and more privacy-invasive, features (such as targeted ads via Addressable TV) is still ongoing. To understand whether European users are aware of the risks associated with this new TV viewing experience, we conducted an anonymous survey on a sample of 174 individuals in Italy. We reached the sample by sharing the URL of the online survey on social media platforms (e.g., Facebook groups), aiming to minimize bias in the selection process, specifically by targeting groups without a specific background or interest. However, this cannot be ruled out. The only requirement to participate in our survey was that participants were older than 18. The survey was conducted in Italian.

Our survey builds upon a study conducted by Ghiglieri et al. [111] in 2015, which assessed users' awareness levels regarding privacy and security with HbbTV. Ghiglieri et al. conducted their questionnaire in Germany. As reported in Section 5.3, it confirmed a generally low level of awareness of privacy- and security-related risks in this context. It also showed that, even if exposed to the risks associated with the uncontrolled use of Smart TVs, users are not willing to fully disconnect their devices from the Internet to avoid losing the additional features offered. Thus, a solution that provides security (against ill-intentioned attackers), privacy, and functionality is needed.

Methodology. We generally follow the approach adopted by Ghiglieri et al. [111], with the following differences: (1) we perform our survey six years after theirs (i.e., "post-GDPR"), (2) in a different country (Italy instead of Germany), and, most importantly, (3) with more targeted questions that focus on HbbTV rather than general Smart TV security. Our goal is to determine whether users' awareness has improved with the more widespread adoption of HbbTV and Smart TVs, and whether Italian users have a different approach to privacy compared to German ones. We employ a mixed-methods design, incorporating both quantitative and qualitative methodologies. We include both closed questions (multiple or single-choice) and semi-structured open-ended ones. We analyze the latter by adopting an open coding approach, where we cluster answers into categories and assign them a code to facilitate additional analysis later. We present the initial codebook in Table 5.5. We built the study using SoSci Survey, a German platform that allows for heavy customization of sections [282]. Their servers are located and operated in Germany, and data is processed in accordance with GDPR specifications.

Ethical Considerations. When conducting the survey, we respected the ethical guidelines of our affiliated organization, and received ethical approval from our university.

In addition, on the first page of the questionnaire, we report our contact information for the participants and explicitly state that participation is voluntary; respondents can stop the survey at any point, and we do not consider the answers they give if they decide to do so. Furthermore, we inform participants that we use the gathered data only in the context of this study.

We put particular care into anonymizing the results; although we do not ask for Personal Identifiable Information (PII), participants might disclose such information in the open questions. When defining the codebook, we exclude any PII.

Survey Structure. Due to space constraints, we only provide a summary of our survey structure, but provide the full survey briefing and questions online [52].

1. *Introduction*: we inform participants about the topic of the survey. We omit some technical details not to influence their answers. We include survey details, such as its anonymity and the duration.
2. *(Smart) TV Demographics*: we ask participants whether they own a TV or a Smart TV. We ask those who do not own a Smart TV if they would consider buying one.

Table 5.5: Codes defined for open-format questions of the awareness survey.

	Codes / Answers	Count
Identified Risks	Privacy (tracking and profiling)	26
	Data and credential leak	17
	Entry point for attackers (TV to access (W)LAN)	11
	Hacking (viruses, broken protocols)	8
	Outdated and vulnerable software/firmware	4
	Access to microphone/camera of the TV	2
	Children accessing not suitable content	2
	Same risks as PCs	1
	Vendors blocking content	1
Countermeasures	Firewall or TV in DMZ/separate LAN	9
	Uninstall unused apps and use only trusted ones	4
	Parental control	3
	Block TV webcam and place TV in a “non-sensitive” area	2
	Disable Automatic Content Recognition or Cookies	2
	Data encryption	2
	Antivirus	2
	Two-factor authentication for accounts	1
	Block bootloader	1
	Refresh MAC address	1
	Buy TV with customer support	1
	Use with care	1
Collected Data	Viewing preferences and times	36
	Personal information (date of birth, email, political orientation)	10
	Geographical location	4
	Credentials	3
	List of installed apps	3
	Purchase-related information	2
	TV model	1
	Nearby WiFi networks	1
	Devices on same network	1
	Website history	1
	Voice	1
Desired Features	Easy to use and deploy	28
	Customizable (expert / non-expert)	25
	Does not hinder TV experience	14
	Secure	8
	Frequently updated	4
	Safe logins (encrypted credentials)	3
	Affordable	3
	Blocks all tracking	3
	Child-safe	1
	Request log	1
	Blocks purchases	1
	Expandable to other smart devices	1

Only those who own a Smart TV, or want to buy one, continue to the next section. We redirect the others to “Final Questions.”

3. *Awareness of Security and Privacy Risks*: we ask participants if they are aware of security and privacy risks of Smart TVs; if yes, they should enumerate them and eventual measures to counteract these risks.
4. *HbbTV Statistics*: we ask participants whether they ever encountered HbbTV notifications and if they are aware of how such protocol works.
5. *Risk Assessment of Scenarios*: we give participants eight different risky scenarios for HbbTV (one per page) in random order. For each scenario, we ask them to give a score based on how critical they think it is. The score ranges from 1 (very low risk) to 5 (very high risk) similar to a Likert scale. Additionally, we ask participants to justify their rating. We provide the full list of scenarios below:

    a) The channel you are watching gets information about when and how long you watch it. For broadcasters with multiple channels, there is the possibility that they will merge the information from channels.

    b) Your usage habits (i.e., what you use your Smart TV for, when, and how often) are stored by the TV broadcasters. The information collected about you is analyzed to show you personalized (i.e., tailored to you) advertising.

    c) Your usage habits (i.e., what you use your Smart TV for, when, and how often) are stored by the TV broadcasters. The purpose and the way such data is stored are not explicitly stated and not certain.

    d) Your usage habits (i.e., what you use your Smart TV for, when, and how often) are stored and analyzed by the TV broadcasters.

    e) A TV broadcaster offers you the possibility to direct home shopping of the item that is being advertised by simply entering your credentials and credit card information on its website.

    f) A TV broadcaster offers you the possibility to direct home shopping of the item that is being advertised by simply entering your credentials and credit card information on its website. It cannot be ruled out that such information is not only received by the broadcaster itself.

    g) TV broadcasters may rely on and aggregate data about you coming from bigger services, such as Google and Facebook, to better tailor their content to your preferences.

    h) TV broadcasters may rely on and aggregate data about you coming from bigger services, such as Google and Facebook, to better tailor their content to your preferences. This might also be used to show you targeted advertisements. It cannot be ruled out that such information is sold to other parties.

6. *Privacy Policy Questions*: we ask participants whether they read privacy policies when accessing digital services and if they were ever shown banners asking for data treatment consent when watching TV.

Table 5.6: Five modes of connecting a Smart TV (rows) and their properties (columns).

Modality	**Internet Features**	**Risk**	**Additional Effort**	**Additional Cost**
Smart TV connected to the Internet without further precautions	No restrictions	Potential risks	None	None
Smart TV not connected to the Internet	None	None	None	None
Smart TV used only as external monitor (no Internet)	Only standard computer functions—no updates	None	Computer configured and connected	None
Smart TV secured by *user-installed* protection software before Internet use	No mandatory restrictions	None, limited	One-time 15 min configuration	One-time 20 €
Smart TV secured by *pre-configured* protection software before Internet use	No mandatory restrictions	None, really limited	None (pre-configured)	One-time 40 €

7. *Selection Grid*: we present participants with a table showing five modalities, reported in Table 5.6, of connecting the TV to the Internet with different security levels and extra functionalities available. We ask them to vote for their preferred one and also mention all the desired features that a tool to enforce security in this context should have.
8. *Final Questions*: we ask participants their age, gender, and their area of expertise to have insights on the demographics of our participants.

General Statistics. 817 people received the URL pointing to the survey. 174 participants completed it, with 55 providing female (32%), 116 male (67%), and 3 (1%) other as gender. The youngest participant was 18, the oldest 80, with a mean age of 37.6 years and a standard deviation of 14.8. Out of the 174 participants, 132 answered that they either possess a Smart TV or would be willing to buy one. Those continued the study while we redirected the others to the Final Questions section. In the remainder of this section we only consider the 132 responses and omit the remaining 42 ones.

Security and Privacy Awareness. 90 participants (68%) did not mention any risk—confirming an alarmingly low level of awareness, although showing a significant improvement from the 2015 survey where 84% of the participants did not report any risk; 26 (20%) participants identified only one risk; the remaining (12%) identified either 2, 3, or 4 risks. The most frequently mentioned risk relates to privacy and consists of tracking and profiling (26 participants, 20%). The second most frequent answer (17 participants, 13%) is data and credential leakage due to unencrypted traffic or unreliable services. Lastly, only 26 participants (20%) mentioned at least one security measure to prevent such risks, with firewalls the most cited answer (9 participants, 7%).

Table 5.7: Mean risk scores assigned to the eight scenarios and their standard deviation (σ) compared to Ghiglieri et al.'s survey [111] from 1 (very low risk) to 5 (very high risk).

Scenario	**IT (2022)**	**DE (2015) [111]**
1: Aggregation of usage data (viewing habits)	2.70 (σ=1.04)	2.82 (–)
2: Collection of usage data for personalized ads	3.09 (σ=1.09)	3.16 (σ=1.41)
3: Collection of usage data for unclear purpose	3.42 (σ=1.13)	3.64 (σ=1.28)
4: Collection of usage data for TV broadcasters	2.88 (σ=1.12)	3.22 (σ=1.44)
5: Shopping personalized ads	3.63 (σ=1.19)	N/A
6: Shopping personalized ads with insecure handling	3.97 (σ=1.15)	N/A
7: Data aggregation for personalized content	3.05 (σ=1.06)	N/A
8: Data aggregation for personalized ads	3.49 (σ=1.05)	N/A

HbbTV Statistics. 77 participants (58%) reported having seen HbbTV notifications while using their Smart TVs, showing an increased user percentage from the 2015 survey. However, only 15 (11%) correctly mentioned that such a protocol combines standard broadcast signal and broadband communication to deliver Internet-based content.

Risky Scenarios. When presented with broadcasters being able to store and analyze usage habits (scenario 4), participants assigned an average risk value of 2.88. All it took was adding that the information is used to show personalized advertising (scenario 2), to raise this value to 3.09. In addition, respondents gave a risk score of 3.49 to broadcasters who might aggregate data from other services and sell information to third parties (scenario 8). This highlights how users are concerned about how their data is used for profiling, but there is little awareness. The responsibility and duty are then in the hands of broadcasters to ensure the consensual handling and collection of their users' data. For the complete risk scoring assigned to each scenario, refer to Table 5.7. Finally, we compare values assigned to the first four scenarios in the 2015 survey. We notice a slight decrease in participants' risk scores. Still, values are comparable, and security seems to be a major concern.

Compared to a study by Malkin et al. in 2016 in the U.S. [177], Italian participants seem to be more concerned about personalized content. Out of the 591 respondents in the U.S., most found acceptable that viewing history is shared with broadcasters to improve personalized suggestions (as long as such information is not repurposed). Furthermore, when asked if they thought data could be repurposed (e.g., for ads), only 37% of the U.S. participants listed this as a possible scenario.

Privacy Policies. Of our 132 participants, 90 (68%) stated that they never or rarely read the privacy policy presented when accessing a digital service for the first time, and 123 (93%) indicated that they did not read such privacy policy presented while watching a TV channel (or were not able to answer). Only 47 respondents (35%) mentioned at least one type of data that could potentially be collected while using an Internet-connected Smart TV. The most mentioned collected type of data is "viewing times and preferences" (36 times), while the second most is "personal information," e.g., email addresses and

birth date (10 times). Only 4 (3%) participants mentioned that broadcasters could collect their geographical location.

Preferred Usage of Smart TVs. Lastly, we asked participants how they would like to connect their Smart TV to the Internet, considering security aspects, functionality, required effort, and costs. 45 participants (34%) preferred the cheaper solution that requires some configuration to secure the Internet communication of the Smart TV. Only 23 participants (17%) voted for connecting the device without further security measures, while 80 respondents (61%) would be willing to adopt some solution to improve security. The most mentioned features to consider when designing a tool to protect Smart TVs are "ease of use" (28 times) and "highly customizable" (25 times), with eventually two different options for both expert and non-expert users. Conversely, Malkin et al. showed their participants would likely be willing to give up (at least theoretically) some extra features to prevent data sharing [177].

5.7 HbbTV Blocker

To mitigate the privacy issues described in Section 5.2.2 and Section 5.5 and to protect users against broadcasters that are not privacy-compliant and/or misconfigured, we designed and developed a prototype for an HbbTV Blocker. HbbTV Blocker demonstrates how a simple yet effective denylist-approach similar to adblockers in the web and mobile domain can be deployed in this domain. Our implementation consists of a gateway that intercepts traffic to and from the Smart TV and domain filter lists on a per-TV-channel basis. We present the architecture of the HbbTV Blocker in Figure 5.3

Gateway vs. Proxy. In our proof-of-concept deployment, we use a Raspberry Pi [244] as the gateway and connect it via Ethernet to Internet router while we connect the Smart TV to the WiFi hotspot of the gateway. In such a way, all the traffic directed to the Smart TV passes through the gateway. We designed the prototype to work on nine Italian channels studied in Section 5.5. Expanding the tool to other channels is straightforward but would require us some manual effort. We plan to automate this procedure further to extract HbbTV URLs and ease the inclusion of new TV channels.

The reason why we chose a gateway over the proxy used by Ghiglieri and Tews in their Privacy Protector [110] is twofold. On one side, they adopted mitmproxy as a transparent proxy which requires its CA to be installed in the Smart TV to capture HTTPS traffic. Root access is required to install such certificates. Unfortunately, gaining root privileges on a Smart TV is not easy since no documentation is available, and every different model has its custom procedure (if any). Their approach worked fine a few years ago when primarily only HTTP was used in HbbTV communication, but it requires more effort with the increased adoption of HTTPS.

Additionally, even with simpler proxies that do not act as MITMs but collect traffic headers, there is a problem when setting those on Smart TVs. Android proxy settings apply only to browser traffic. Other applications' traffic, including HbbTV, even though

technically it is also rendered in a browser, does not pass through the proxy for security reasons. Root access is required to bypass this limitation [194]. Thus, we deemed the gateway approach the best in terms of universality and adaptability to different models, brands, and operating systems of Smart TVs considering the requirement of "ease of use."

Channel Identification. As a first step, HbbTV Blocker identifies the TV channel a user is currently watching. To do so, a Python script intercepts DNS queries using the pyshark library [154] (a wrapper for tshark), and filters them. If the contacted domain matches against specific string patterns defined for each of the nine channels (see Table 5.2), we set the current channel to the matched channel.

Per-Channel Denylist. We create a denylist of domains for each channel based on the tracking and analytics domains identified in Section 5.5. We use *iptables* [172] to enforce those lists. We add a new rule to block this specific traffic for each entry in the denylist. The format of the rule is the following `iptables -A INPUT -m string -string "domain" -algo bm -to 65535 -j DROP` with *domain* being replaced with the current entry.

We preferred the approach of denylists over allowlists: the latter would require defining a specific set of allowed domains for each application used by the user. Thus, we would risk unintentionally blocking this traffic whenever a user installs or uses a new application on their Smart TV. Additionally, defining allowlists for services like Google or Amazon AWS is a non-trivial task considering the number of domains and subdomains they use. Thus, we deem the denylist approach more scalable if new applications are installed and easier to manage. Other widely used tools, for example, Pi-hole [128], also adopt this solution to block unwanted traffic but do not offer the same granularity as HbbTV Blocker.

For each channel, we individually test the domains we flagged as tracking to evaluate if blocking them causes some breakage of the HbbTV application's intended functionalities. If by blocking such the application stops working, we exclude them from the channel denylist as this would mean losing the extra interactions offered by the broadcasters.

In addition, we decided to check for new tracking and advertisement domains by matching incoming traffic to the most recent versions of Pi-hole and EasyList lists. The users will then be displayed with the flagged domains and have the option to block them. We consider this necessary as such lists also block domains necessary for the correct functioning of the HbbTV application as hbbtv.mediaset.net.

HTTP Blocking and Upgrade. HbbTV Blocker additionally allows blocking all HTTP traffic via the following iptables rule *iptables -I FORWARD 1 -p tcp –destination-port 80 -j DROP*. Although receiving only encrypted traffic might be a good security practice, blocking all HTTP traffic might hinder the HbbTV application's functionalities. A better solution would be upgrading all HTTP traffic to HTTPS (if the remote endpoint supports it, similar to Desktop browsers), which we leave for future work.

Configuration. We designed our approach with ease of use and customization in mind based on users' preferences expressed in our survey in Section 5.6. We provide the user

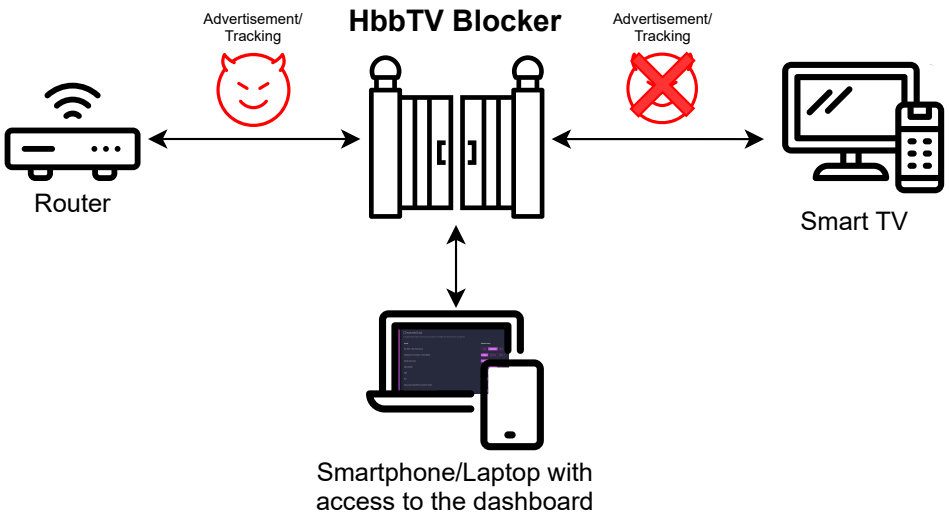


Figure 5.3: Overview of HbbTV Blocker in our testing environment.

with a graphical dashboard where different options are available; they can decide each channel's traffic behavior, block all HTTP traffic, get an overview of the number of blocked requests per channel, and upload customized denylists. In particular, the user can select three modes of operation:

- **Allow all**: all the traffic, including tracking and profiling, passes via the gateway.
- **Block tracking**: enforces the denylists designed to block tracking and analytics.
- **Block all**: blocks all traffic.

The *Block all* modality might hinder user experience since it also blocks traffic unrelated to HbbTV. We present the user with an informative alert when turning on this option.

We acknowledge that HbbTV blocker shifts the responsibility to the users of the Smart TV: they have to maintain the tracking denylists to ensure they are updated and complete. However, we provide the user with a trade-off between deciding to block all traffic and potentially impeding functionality or only selectively blocking certain domains. Again, we took Pi-hole as inspiration since most of its available denylists are written by the community of users and shared across them. Users can easily upload the updated denylist via the graphic interface they have access to.

Usability Testing. To verify that we respected the criteria of "ease of use" and "customization" highlighted by the user survey in Section 5.6 we conducted an in-person moderated study on a sample of five participants. Each interview lasted for 20 minutes, and we asked participants to install and interact with our tool (we gave them a Raspberry

Pi with Raspian already installed for time reasons and being its installation out of scope for our usability study). We informed our participants what data we were gathering and how we used it in the context of this study. Finally, we asked for their consent before and after the interview to respect ethical guidelines.

Participants generally did not report any difficulties in interacting with the HbbTV Blocker. Nevertheless, we received suggestions such as having a dedicated page for un-experienced users with a recommended configuration, i.e., that blocks tracking traffic without breaking functionalities. In addition, participants reported in some cases that options needed an adequate/simple description to understand them fully. We fixed the issues highlighted by our respondents.

Performance Evaluation. We used a Raspberry Pi 4 with 8GB of RAM and collected the number of packets exchanged per second and the number of bytes per second, CPU, and RAM usage. Such measures are collected by a script every 10 seconds to avoid the observer effect. As described above, we connect the Raspberry Pi through the Ethernet interface to the router and the Smart TV to its WiFi hotspot. Then for 30 minutes, we use the TV to navigate the channels and utilize their HbbTV applications. We report the results in Table 5.8 for when the Raspberry Pi is idle, i.e., HbbTV Blocker was not running, and when it was active. Despite increasing resource usage, our results indicate that the tool is not highly demanding. Additionally, the relatively low number of exchanged packets for HbbTV applications makes the tool suitable for its intended task of blocking requests without interruptions of the viewing experience.

In addition, we compute the latency introduced by our architecture. We run *curl* 100 times on three different HbbTV URLs (hbbtv.rds.radio, ht.la7.it/index.php, discovery.castoola.tv/realtime) with the Smart TV directly connected to the router or through the Raspberry Pi and extract the request time. We see an increased time in handling the requests; for the first URL, the average time in seconds is 0.012 and 0.28 without and with the Raspberry Pi. Nevertheless, a higher standard deviation in the second case (0.99) shows that a few requests took a long time, making the average not a trustworthy measure for latency. The median, instead, is 0.021 seconds, showing a reasonable overhead.

Comparison with Existing Tools. Most of the solutions we discussed in Section 5.3 are dependent on TV manufacturers, i.e., they block tracking carried out by the latter. As we investigated, the denylists proposed are insufficient to protect users against the heavy tracking of HbbTV-enabled channels, although currently representing one of the best approaches to defend against privacy risks. Solutions, such as the Security Manager proposed by Matejka [183], only focus on the security and privacy of independent apps; instead, HbbTV runs as a web application in the TV's built-in browser. Thus, we deem HbbTV Blocker a necessary solution to protect users against the significant tracking carried out via the HbbTV protocol.

Table 5.8: Performances Tests Results of HbbTV Blocker.

	Idle	**Running**
Average CPU usage (%)	3.70%	35.66%
Average RAM usage (%)	11.14%	14.66%
Average Packets per second	0	256 (max 7,433)
Average KBytes per second	0	176 (max 7,600)

5.8 Discussion

Both the results presented in Section 5.6 and Section 5.5 highlight a problematic immaturity in the context of HbbTV adoption on both broadcasters' and users' sides. The results of our technical analysis show the negligent behavior of broadcasters offering HbbTV applications. Fair treatment of users' data is not always guaranteed, as we demonstrated with the connection to tracking services before their consent. Remarkably, we find several violations of GDPR. Tracking before the user has expressed their consent contradicts the "Conditions for consent" of the European Regulation and the guidelines on cookies by the Italian GPDP. Additionally, withdrawing consent, with the consequent deletion of data, should be possible and as easy as giving it, according to the same statement of GDPR. We show this is not the case for channels that do not allow consent revocation or ask the user to contact their dedicated office directly. Even channels allowing consent revocation only do so partially, or in one case, do not honor the users' consent revocation. The absence of the privacy notice when accessing the HbbTV application for the first time, as for TV8, signals a violation of transparent information communication and the provision of correct information to the data subject. As for ORF, the "hidden" policy in a sub-menu of the app violates the principles of transparency. The incorrectness, incompleteness, and non-transparency of the privacy policies presented by the different channels do not help users understand what information is collected and how it is processed. This violates the principles of processing personal data, providing correct information, and transparency.

Users seem unaware of the potentially privacy-invasive appliance they have in their homes. This likely comes from a past mental model where TVs were just "secure" and their communication channel uni-directional, i.e., users were mere consumers of TV content. When we asked about the risks associated with Smart TVs or HbbTV, only few participants could mention at least one. Despite this lack of awareness, when confronted with potentially risky scenarios, users seem to be highly concerned about their data and personal information, e.g., viewing preferences, even more than what was reported by Ghiglieri in his survey on German users [111] in 2015. This highlights profound technical illiteracy, possibly leading to privacy problems for unaware users.

The consequences of having such a "relaxed" approach towards privacy in the Smart TV context, and more specifically towards the HbbTV protocol, could lead to several issues. In particular, additional issues arise from adopting dynamic and targeted advertising, now being pushed as part of "Addressable TV" by several players in this ecosystem [54, 89,

163]. With their persuasive power, algorithms can "nudge the behavior of data subjects and human decision-makers by filtering information" [192]. In the HbbTV context, users are encouraged to buy certain suitable products. Additionally, users' data can be aggregated and used for better profiling without their awareness. However, the risk does not necessarily stop there: we argue that while users have become more accustomed to targeted advertisement when surfing the web and using mobile apps, they are still used to the "linear TV" experience where the same content is provided to all viewers. Additionally, they might put particular trust into certain TV broadcasters, such as public and well-established news channels. However, with this paradigm shift, content, including (potentially political) advertisement can be highly personalized and targeted—by both private and public TV broadcasters and channels.

In terms of security risks, HbbTV content is not always encrypted, thus potentially leading to MITM attacks that can replace the HbbTV app's URL to load a different content or overlay it. This could be abused to spread misinformation through fake news, or for phishing attacks. Users can directly interact with interactive ads to make online purchases. As no sanitization or additional checks on the URL's source is performed, an ill-intentioned party could replace legit ads with malicious ones; thus, tricking the user into inserting sensitive data in a fake checkout page.

Finally, the advent of online shopping apps via HbbTV is already a reality in Germany and will not take long to reach other European countries. The user must insert sensitive data such as credit card information, billing address, and name to purchase online. The incorrect handling of this data might lead to severe security issues, such as theft of personal information and credentials. As reported in Section 5.5, the use of plain HTTP with no encryption of data when logging in to such services poses a serious threat to the user. Even more worryingly, in the case of HSE, this issue has been known since 2014, but still persists. To protect users from misconfigurations on the broadcasters' side, browsers on Smart TV's should catch up with their desktop counterparts, which nowadays upgrade connections to HTTPS by default. To further protect connections from eavesdropping, certificate pinning could be an option, but comes with maintenance overhead and again the risk of misconfiguration [239].

5.8.1 Key Takeaways

Following the discussion of our findings, we highlight our main takeaways:

- All the 36 TV channels contact at least one tracking domain; further, 26 communicate with trackers before the user has expressed their consent. Seven channels do not present any privacy policy when accessing them for the first time.
- 20 of the 36 TV channels (56%) adopt the invisible "tracking pixel" to profile users.
- Commonly used tracking denylists only block at maximum 44% in 2021 and 81% in 2022 of the domains in our traffic captures and marked as tracking.

- We found HTTP communication in most of our traffic captures; such traffic contained sensitive information such as device IDs, visitor IDs, country codes, and ISP information.
- The shopping channel HSE allows users to create accounts and log in over plain HTTP, thus exposing sensitive information to potential attackers (e.g., credentials and credit card number).
- Out of the 132 participants in the Smart TV and HbbTV awareness survey, 68% could not mention any security or privacy risk, and 68% stated they never read privacy policies presented by digital services.
- When confronted with risky scenarios, respondents' average risk score (from 1, low, to 5, high) ranges from 2.70 to 3.97; thus, users are highly concerned with their security and privacy but unaware of the risks.

5.9 Limitations and Future Work

For our study of TV broadcasters' privacy posture, we only selected a subset of channels that offer HbbTV functionalities based on different parameters, e.g., audience share. To complement our study, we plan to include further channels to have a complete overview of the HbbTV landscape in Europe.

Our user survey relies on self-reporting; therefore, we cannot verify whether the participants' claims correspond to their actual behavior when interacting with a Smart TV. Several studies have highlighted this discrepancy between self-professed privacy attitudes and actual behavior [147, 242, 248]. The understanding of how attitudes and behaviors diverge represents an orthogonal research challenge.

We intend HbbTV Blocker as an initial step in its development, which should be considered a prototype. We do not consider the performance results reported in Section 5.7 as complete, but they only glance at the resources' usage. We would further require usability testing to verify whether further customization options are needed and whether it fulfills the "ease of use" requirement. Furthermore, the *Block all* feature might impede a Smart TV's functionality; there is no straightforward way to detect when the user exits the "standard" TV channels and switches to a different app, such as YouTube, which might lead to unintended blocking. Still, we provide HbbTV Blocker's code as open source and argue it could be expanded with modules blocking tracking traffic on other smart home appliances, e.g., smart refrigerators.

In general, security and privacy issues through the integration of web content in other types of applications has been well documented, and still persist, most notably for different kind of WebView implementations on Android [38, 60, 200]. Studies have also shown differences in the security and privacy posture of different mobile browsers for the Android OS—in particular compared to desktop versions [173, 238]. Whether the

same issues exist in – potentially vendor-customized – browsers that ship pre-installed on (Android-based) Smart TVs is an interesting direction for future work.

5.10 Conclusions

Our study shows that users are exposed to severe privacy issues in the context of HbbTV-enabled devices. We analyzed the HbbTV traffic of selected European TV broadcasters and observed their adoption of invasive profiling and tracking (third-party) services. Our results show the TV broadcasters' negligence in handling users' data and, in particular, compliance issues with current regulations regarding user consent. Compared to previous studies in Germany, the situation seems not to have evolved over the past five years.

We then performed a user study in Italy to gain insights into their security and privacy awareness. Users seem to have worryingly low awareness of the risks linked to Smart TVs and HbbTV. However, they show great concern when confronted with potential issues—highlighting the need for tools enhancing the security level of the TV viewing experience while also improving the understanding of risks.

As a first step, we propose a solution to mitigate the privacy issues arising from the (essentially unregulated) adoption of HbbTV: HbbTV Blocker. It considers users' need for customization and ease of use by selectively blocking known ad and tracking domains.

CHAPTER 6

Mind over Matter: How Device Functionality Affects Smart Home Security Behaviors

Internet of Things (IoT) devices are increasingly present in households, offering convenience while also raising concerns about privacy and security. Prior research has documented technical vulnerabilities and examined the willingness to pay for better protection. Still, it remains unclear how people distinguish between device types and how these distinctions influence their security practices. This matters because users may perceive risks differently depending on how they understand the device's functionality. To investigate this, we conducted an expert card sorting study with ten researchers and practitioners from HCI, security, and IoT, identifying five categories of domestic IoT devices. We then utilized this categorization in an online survey (N = 213), measuring trust, perceived risk, and willingness to use across device categories using validated scales. We found that *Physical Access Devices* elicit significantly different security behavior than other categories. We contribute a grounded categorization of IoT devices and demonstrate their potential value for design, regulation, and user protection.

6.1 Introduction

Internet of Things (IoT) devices are essential in our current society. They have entered our houses and help us with everyday tasks, ranging from floor vacuuming to smart alarm systems. Consumer spending on smart home products and services is projected to reach $150 billion in 2024, highlighting their pervasiveness in our society [273]. Smart home devices collect, process, and store a conspicuous amount of data about the surrounding environment and the users themselves [247, 269], while attacks on IoT devices are far from unprecedented. In 2023, a popular user device vendor, Anker, admitted that their

Eufy smart camera streams were not always encrypted [129]. An attacker could access the recordings and potentially plan a robbery, knowing

While previous research has extensively explored the security vulnerabilities of IoT devices [230, 285, 301], there remains a gap in the literature about how users perceive and decide to acquire smart home devices. Prior studies have shown, for example, that users are willing to pay more to purchase smart devices with improved security and privacy and that different users' roles (e.g., primary user or bystander) impact their perception of a device [31, 88, 106, 180]. These works indicate that users value security and privacy, but they stop examining how such perceptions may differ depending on the IoT device category. Thus, it remains challenging for HCI to determine if and how functionality affects users' perception and behavior. Mathis et al. [184] argue that the community must build stronger bridges between technical and human-centered methods, validating that investigating the HCI and Security & Privacy intersection is necessary.

We examine how various categories of devices impact users' attitudes, particularly in terms of trust and perceived security. To our knowledge, no prior work has systematically defined IoT device categories to study user security perceptions. As a result, our first step was to identify representative clusters of IoT devices, which we then used to investigate whether and how user trust and willingness to adopt vary across these categories.

In this paper, we make the following contributions:

- We propose a novel categorization of IoT devices, derived from an expert card-sorting study with ten domain experts in security, privacy, IoT, and HCI. The resulting five categories, *Physical Access Devices*, *Surveillance & Assistance Devices*, *Media & Entertainment Devices*, *Household Convenience Devices*, and *Environment Monitoring & Control Devices*, offer the first empirically grounded framework for classifying domestic IoT devices.

- We empirically examine how user trust, perceived risk, and adoption intentions vary across categories through an online survey ($N = 213$). We show that *Physical Access Devices* stand out as uniquely risky and less trustworthy, while *Media & Entertainment Devices* are perceived as less risky and more attractive for adoption.

6.2 Related Work

The IoT ecosystem offers many benefits but faces major security and privacy challenges, often due to low-cost devices with poor firmware, lack of standardized protocols, and the rapid proliferation of connected products [302]. Many rely on weak default credentials, lack encryption, and expose insecure APIs, making them vulnerable to attack [301]. At the same time, IoT devices frequently collect sensitive health, biometric, and location data, underscoring the need to empower users to make more informed decisions.

Prior research highlights a gap between user trust and actual security. Mandal et al. show that nine out of 11 IOXT [1]-certified companion apps contain security flaws, yet users overwhelmingly trust them since certified [178]. Naser et al. find that security, privacy, and familiarity increase trust in IoT healthcare, strengthening adoption attitudes and reducing perceived risk [9]. Frik et al. demonstrate that perceptions of data acceptability vary by context (e.g., smart home owners vs. bystanders, as Airbnb guests), data type, and recipient, with audio data from smart speakers rated least acceptable [106]. Similarly, Marky et al. highlight that guests in smart homes often lack awareness or control over data collection, leading to tensions around consent and trust in shared spaces and asymmetries in mental models and control over data [180, 181, 182]. Parallel work underscores the challenges of usable authentication in domestic settings, where solutions must balance convenience, physical access, and shared household dynamics. Prange et al.[240, 241] stress that authentication must accommodate multiple users and contexts, while Mecke et al.[186] find that perceptions of physical, biometric, and behavioral methods vary widely, with no universal fit. These studies show that privacy and authentication design cannot be treated as purely technical problems but require HCI perspectives, tightening the link between these two research domains. Further work explores adoption trade-offs: Reitinger et al. find that urgency often outweighs privacy when outsourcing IoT repairs [23, 252], while Williams et al. confirm the privacy paradox, as users still purchase devices despite recognizing risks [320]. Pierce frames smart cameras as "creepy" and identifies three design factors, digital leakage, implicit functions, and "foot-in-the-door" devices, that shape acceptability [236]. Oser et al. [227] developed the *SAFER* framework and demonstrated that its application increased users' awareness of security issues when deployed in a large multinational organization. Borgert et al. [43] proposed a scale specifically for the smart home context to evaluate cybersecurity self-efficacy.

Existing work, however, typically isolates single domains (e.g., healthcare, access control) or device features (e.g., audio capture, biometrics), making it difficult to generalize how functionality shapes risk and trust. What is missing is an empirically grounded framework showing how users differentiate between device categories and how these distinctions influence trust, risk, and adoption. Our work addresses this gap by developing and validating such a categorization for domestic IoT devices and showing how categories systematically influence perceptions of trust, risk, and adoption.

6.3 Method

Prior research has not examined whether user security perceptions vary across different categories of IoT devices, nor has any standardized categorization been established. To operationalize functionality, we first needed a representative clustering of IoT devices. We therefore designed a two-study investigation: (1) an expert study, in which domain experts generated device categories based on IoT cards, and (2) a user study, in which a

[1] https://www.ioxtalliance.org/

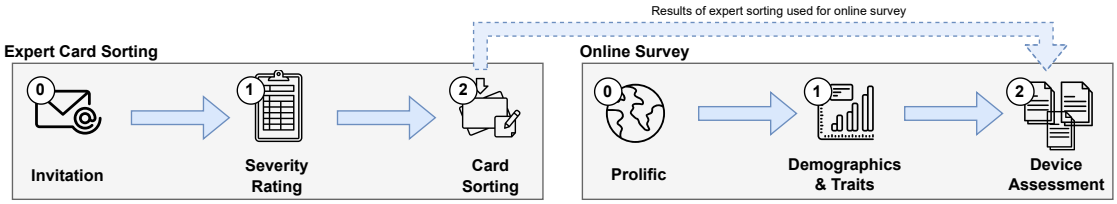


Figure 6.1: Procedure of the two conducted studies.

representative sample of Prolific[2] participants evaluated their perceptions of the devices. We outline the studies' flow in Figure 6.1: we first invited experts to participate in our card sorting study via email. They initially completed a severity rating of the cards, followed by the actual card sorting task. For the subsequent online survey, we recruited participants through Prolific. These participants provided demographic information and completed trait scales before assessing devices within the categories derived from the expert card sorting.

6.4 Expert Card Sorting

We first conducted a card sorting study, where experts (1) assigned severity ratings to IoT device cards and (2) clustered them into categories.

6.4.1 Study Design

We conducted pilot runs before each study to ensure robustness and minimize bias. A mock-up survey with three non-experts revealed some key issues. One participant categorized devices by room (e.g., kitchen, living room), not generating meaningful insights for our goals. Others defaulted to binary adoption categories (e.g., "yes" or "no"), which were too broad to capture subtle perceptions. Another participant raised concerns about how brand reputation and assumed security protections might influence their responses, highlighting a tradeoff between providing participants with sufficient guidance and avoiding priming effects. These reflections helped us refine our instructions and survey, guiding participants toward more analytically relevant categorizations.

We developed 29 cards representing IoT devices, selected from top-selling products on major retail platforms (e.g., Amazon, Google Shopping). To avoid bias, we instructed ChatGPT [3] to remove vendor and manufacturer identifiers and excluded health-related devices (e.g., glucose monitors, fitness trackers), as these raise additional safety and regulatory concerns beyond our scope. To further strengthen the validity of our cards, three researchers independently reviewed all of them to ensure consistency, clarity, and neutrality. In addition, we adopted ChatGPT to generate information on what kinds of

[2] https://prolific.com/
[3] https://chatgpt.com/

Vacuum Cleaner – IoT Device Profile	
Short Description	IoT vacuum cleaner controlled remotely via the mobile app. Automatically maps and cleans floors on a set schedule or on demand. Users can start, stop, or monitor cleaning sessions from anywhere. It avoids obstacles, and returns to its charging base automatically.
Collected Information	• Cleaning activity logs (timestamps, duration) • Generated environmental maps (floorplan) • Device geolocation metadata (inferred via IP) • User identifiers (username, email, ID) • Interaction telemetry (app commands) • Network parameters (SSID, IP, MAC address)
Connectivity & Sensors	• Visual camera for spatial mapping • WiFi + Bluetooth module • Infrared sensor for night vision and obstacles detection • Acoustic or optical dust detection module

Figure 6.2: One of the 29 IoT device cards – a vacuum cleaner.

data these devices typically collect and which sensors they are commonly equipped with. All cards followed a consistent four-part structure (see Figure 6.2).

6.4.2 Expert Participants

We recruited ten domain experts from academia and industry via our networks, a number consistent with prior pre-studies [220, 311]. Nine identified as male and one as female; seven disclosed their age (26-50 years, $M = 32.1$, $SD = 8.5$). Occupations included professors, PhD students, researchers, and engineers, spanning expertise in HCI (smart homes, ubiquitous computing), IoT security, mobile application security, cryptography, network measurement, cybersecurity, and hardware fabrication.

6.4.3 Procedure

For the expert assessment phase overview, see Figure 6.3. Each expert received an email invitation (Step 0) describing the study as an investigation of a potential classification of IoT devices in terms of functionality. To minimize bias, the invitation intentionally avoided detailed information about the study's objectives. The invitation included an informed consent form that we required experts to sign. We conducted the study online and asynchronously, allowing experts to complete the tasks conveniently.

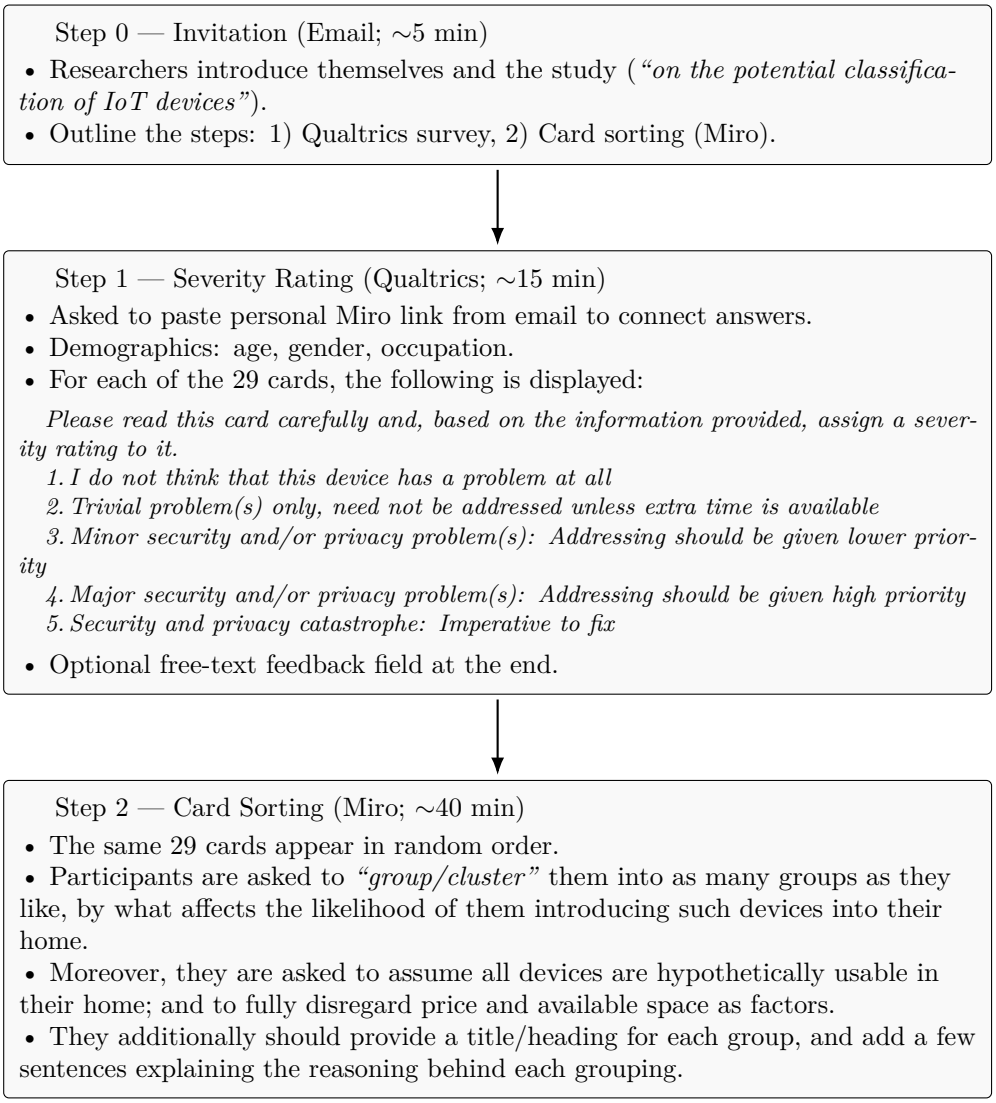


Figure 6.3: Expert card sorting procedure: invitation, severity rating in Qualtrics, and card sorting in Miro.

Step 1: Severity Rating

Following, experts evaluated the 29 IoT device cards based on their perceived severity (Step 1), such as how seriously they considered the potential security and/or privacy risks. To this end, the severity rating served as a sanity check to ensure that experts did not cluster devices based on security levels. We adapt Nielsen's severity scale, but shift the scope from usability problems to security ones [205]. This adaptation enabled us to leverage an established, widely recognized severity framework. We randomly shuffled the device cards to minimize order effects and unintentional associations. We gathered

ratings in an anonymous survey using the five-point scale presented in Figure 6.3.

Additionally, experts provided demographic information, including age, gender, and occupation. We also asked them to enter the link for Step 2 they had received with their invitation, allowing us to match their ratings with clustering data. Finally, an open-text field invited additional comments.

Step 2: Card Sorting

For Step 2, we asked experts to cluster the IoT device cards according to potential associations they might find among them and to name and justify the groups. We chose an open card sorting task, following the approach described by Spencer [283], as it enables participants to reveal their own categorization logic, thereby uncovering mental models and criteria that might not be captured through predefined taxonomies.

We reminded experts that they had previously assigned severity ratings to the devices to be clustered in Step 1. We told experts they could „group/cluster“ the devices into as many categories as they deemed appropriate, based on factors influencing their likelihood of introducing such devices into their home. We instructed them to assume all devices could hypothetically be used in their home, disregarding price and space (e.g., a pet treat dispenser was to be considered even if they did not own a pet).

The task lasted about one hour (5 minutes introduction, 15 minutes severity ratings, 40 minutes card sorting).

6.4.4 Apparatus

We collected severity ratings via Qualtrics[4], while experts performed card clustering on a personalized Miro[5] board. On the Miro board, we provided a brief example demonstrating how to construct clusters with mock devices and labels. We instructed the experts to use “virtual” sticky notes to (a) give each group a „title/heading“ and (b) „add some extra sentences“ explaining their reasoning, thus giving us more insights into their decision process. We provide detailed instructions below.

> Dear Participant!
> Thank you for taking part in our study. You have already completed the first task of assigning Severity Scores to 29 Internet of Things devices. The following are the instructions for the second phase of our analysis. To start, here is a little information about Miro. Imagine Miro as a whiteboard where you can sketch, write down ideas, and rearrange items. You can magnify the board and read even the smallest text by scrolling with your mouse or pinching your laptop’s touchpad.
> On the board, there are the same 29 devices you previously saw. Now we ask you to group/cluster them into as many groups as you like, based on what affects the

[4] https://qualtrics.com/
[5] https://miro.com/

likelihood of you introducing such devices into your home. Assume all listed devices are hypothetically usable in your home environment (e.g., even if you don't have a pet, consider the potential value or implications of a pet treat dispenser). Entirely disregard price and available space as factors.
To do so, drag the IoT device card and move it close to the others you want to cluster together. We show an example with the sticky notes below with two groups. When you are satisfied with your clustering, please use the sticky notes (fourth icon on the left) to give a title/heading to the groups. This way, we can understand your reasoning for adopting such categorization.
Finally, in addition to the headings, please add some extra sentences with an additional sticky note (or similar) explaining your reasoning. Additionally, please name other possible themes these cards could be clustered by that you can come up with.
Now, zoom out (to find all the 29 devices) and have fun! Thank you for your time!

6.4.5 Results

Severity Ratings

We converted the qualitative severity scale into a five-point numerical scale, where 1 represented the lowest perceived risk ("I do not think that this device has a problem at all") and 5 represented the highest ("Security and privacy catastrophe: Imperative to fix"). This allowed us to aggregate and compare expert assessments. The average severity score was 3.1 ($SD = 0.6$), indicating that devices presented moderate security concerns. Devices varied considerably: the soundbar scored lowest (1.6), while the baby monitor scored highest (4.3). This variation underscores that experts did not apply a uniform risk perception across all IoT devices, but instead differentiated their evaluations based on device category and functionality. For instance, devices perceived as collecting sensitive data (e.g., through microphones or cameras) were rated higher. In contrast, utility-focused devices with limited capabilities scored lower.

Card Categorization

The ten experts generated 43 distinct categories, with three to six categories per expert (M=4.3, SD=1.1). We report all the categories in Table 6.1. The list size of IoT devices in the categorization varied considerably. Across all 43 categories, the minimum number of devices was one, while the maximum was 15 (M=7.0, SD=4.2), indicating substantial variability in the number of devices across categories.

Using coder consensus distance between each device pair, we created a similarity matrix normalized by the number of experts who observed both devices. Two devices receive higher similarity scores when multiple experts consistently group them. We then transformed the similarity matrix into a distance matrix and applied hierarchical agglomerative clustering with average linkage. We produced the dendrogram in Figure 6.4, which visualizes the relative proximity of devices regarding consensus among experts. After exploring

Table 6.1: Expert categorizations and perceptions of IoT devices with rows as experts (E1-E10) and columns as category labels.

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6
E1	Ready to go, as is	I'd pay attention when configuring	Would look for alternative with different features	-	-	-
E2	Don't see value	Over-engineered, can be achieved also standalone without IoT	Benefit of connectivity, logs, analytics	-	-	-
E3	Home Appliances	Devices with camera and/or microphones	Devices that thieves and robbers love	Home theatre or office	Devices for convenience	-
E4	Will buy after serious consideration only if really necessary	Might buy	Will probably buy	-	-	-
E5	Not too bad	Telemetry data	Possible safety issues	Big nope	-	-
E6	Likely to use	Maybe	Unlikely, due to Functionality	Unlikely, due to Collected Data	-	-
E7	Security devices	Ambient conditions	Utility and Ambient	Utility devices	Entertainment	-
E8	Top tier	I'm likely to have these in my home, although I'd look for alternatives that collect less data	Unlikely to have	Only as a temporary, convenient solution	I wouldn't even consider buying those	Things I'd never put inside my house
E9	Never	I'd rather not	Maybe one day	There's a chance, maybe i already have it	Uncategorized	-
E10	Acceptable	Benign, but why do they collect data?	Can leak when I'm not at home	Has a microphone	Has a camera	-

alternative cluster solutions and assessing their interpretability, we identified five clusters of IoT devices by cutting the dendrogram at an empirically chosen level. These clusters provide a structured view of expert mental models and serve as the foundation for the user validation study.

To ensure meaningful and interpretable labels, four independent researchers examined the composition of each cluster and proposed potential labels. These are: *Physical Access Devices*, *Surveillance & Assistance Devices*, *Media & Entertainment Devices*, *Household Convenience Devices*, and *Environment Monitoring & Control Devices* as in Figure 6.4.

Several insights emerged. For example, the vacuum cleaner was placed under *Physical Access Devices*, due to their mobility and independent access to private spaces. The voice assistant was included within the *Surveillance & Assistance Devices* category, emphasizing

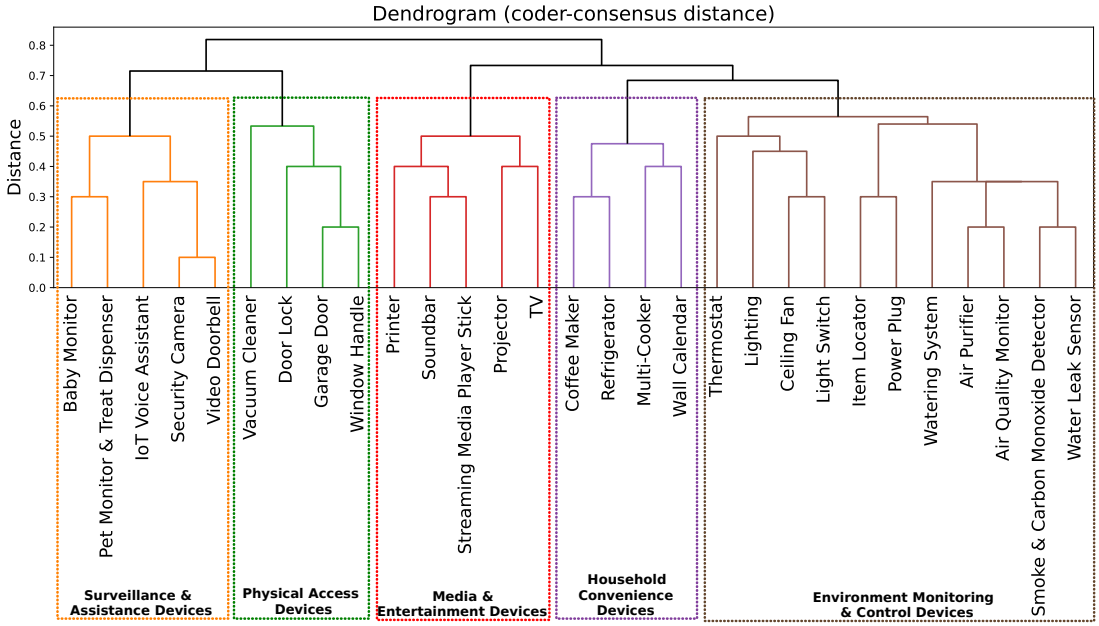


Figure 6.4: Hierarchical clustering dendrogram of IoT devices based on coder consensus distance. We generated the dendrogram using average-linkage hierarchical clustering on the normalized co-assignment similarity matrix. Colored boxes highlight the five clusters derived from expert categorizations, illustrating groups of devices that experts consistently perceived as closely related.

its dual role of assisting users while monitoring conversations and behaviors, raising privacy concerns. Devices in the *Media & Entertainment* cluster often rely primarily on output rather than input, suggesting that experts may have considered them less intrusive and worrisome. Finally, we highlight a difference between our labels "surveillance" and "monitoring": while both involve data collection, we assume surveillance as more invasive because it observes individuals directly, whereas monitoring typically focuses on specific environmental conditions or functionalities (e.g., temperature, leaks).

6.5 Online Survey

As our second study, we conducted an online survey to investigate the effect of IoT device categories on *Intention to Use*, *Perceived Risk*, *Reliability*, *Functionality*, and *Helpfulness*, using established scales.

6.5.1 Study Design

We employed a between-subjects study design with the following conditions.

Conditions

We assigned two cards from each category to verify users' perception of categories. We randomly drew two cards from each category to achieve a balanced representation. We randomized the card order within each participant's set to reduce order effects.

Measures

We used four scales: (1) the McKnight scale on trust in technology [185], adapted to IoT to assess general trust disposition; (2) the CySESH scale of cybersecurity self-efficacy [43], to measure confidence in managing IoT security-related tasks; (3) the Security Behavior Intentions Scale (SeBIS) [86], which evaluates secure practices like password use, updating, and cautious browsing; and (4) a technology acceptance scale [141] (using the constructs *Intention to Use* and *Perceived Risk*), capturing how risk perception influences adoption across categories. Collectively, these scales provide an understanding of users' trust, efficacy, behavioral intentions, and risk-based acceptance of IoT devices.

6.5.2 Participants

Participants had to be 18+ and have some IoT experience. We recruited participants on Prolific to ensure a broad and diverse participant pool, and we compensated them with 9£ per hour, according to Prolific guidelines. Of 221 initial participants, we retained only those passing at least two of three attention checks (in line with data quality standards) and completing the survey, yielding 213 valid responses. The average completion time was 20m53s (SD=8m40s).

Ages ranged from 20 to 71 ($M = 38.8$, $SD = 11.5$); 54.2% identified as male ($N = 116$), 44.9% as female ($N = 96$), and one preferred not to disclose. Occupations spanned IT, engineering, education, finance, healthcare, creative industries, management, and more. A considerable number of participants reported being students (n = 15), retired (n = 8), or unemployed (n = 6), while 27 participants opted not to disclose their occupation. Participants came from 19 countries, with most from the UK ($N = 88$), US ($N = 28$), and Canada ($N = 25$); others from Italy, Poland, Portugal, and elsewhere. English was dominant ($N = 128$), but over 25 languages were represented, including Polish, Italian, Portuguese, and Spanish, underscoring participant diversity.

Participants' Traits

CYSESH Scale. Scores indicated moderate confidence in IoT security management ($M = 4.40$, $SD = 1.07$), with participants confident in basic tasks such as disabling

devices in an attack (52 at "7", 43 at "6") or deleting cloud data (46 at "7", 51 at "6"). Confidence declined for administrative processes (e.g., contacting a data protection officer) and uncovering hidden practices (e.g., identifying third parties with data access), where scores clustered around "2–4." Similar drops were seen for detecting manipulative designs or tracking multi-device privacy risks, highlighting stronger capability in basic protective actions than in complex or institutional tasks.

McKnight Trait Scale. Most participants reported comfort and optimism toward IoT use (108 "somewhat agree," 72 "strongly agree"), trusting devices' reliability and functionality. Perceptions of institutional protections were more moderate, with many choosing "somewhat agree" (92 vendor protections, 111 legal structures) or "neither agree nor disagree" (66 and 51, respectively). The mean score was $M = 57.88$ ($SD = 8.26$). Although the distribution was non-normal ($W = 0.988$, $p < .001$), it was nearly symmetric (skewness $= -0.09$), indicating a generally positive but measured disposition toward IoT.

SEBIS (Security Behavior Intentions) Scale. Participants reported strong adherence to basic practices, with 187 "always" using mobile PINs and over 174 regularly securing laptops and tablets. Password diversification was also common, though behaviors such as screen-locking were variable (47 "always," 43 "sometimes," 42 "never"), and updating or link-checking remained inconsistent (34 "sometimes" opened unverified links). The average score was $M = 59.86$ ($SD = 8.44$). No major outliers were observed, suggesting representativeness of mainstream IoT users. The distribution significantly departed from normality ($W = 0.979$, $p < .001$), with a modest negative skew (-0.51) driven by participants with lower scores.

Summary. Across the three scales, distributions reflected diversity in security behaviors and attitudes. Central tendencies suggested moderate to strong engagement, but variability in CYSESH and a modest negative skew in SEBIS showed differences between confident, proactive users and those less consistent or uncertain. The absence of extreme outliers supports the sample as a broad cross-section of everyday IoT users rather than a specialized subgroup.

6.5.3 Procedure

After a briefing, participants provided demographics and completed the trait scales to capture their perceptions of trust, self-efficacy, and security behavior in the IoT context. They then evaluated two devices from each of the five categories. Participants rated them on *Intention to Use* (three items) and *Perceived Risk* (four items), following Im et al. [141]. Both constructs used 7-point Likert-type scales ranging from 1 = *strongly disagree* to 7 = *strongly agree*. In addition, participants evaluated *Reliability* (four items), *Functionality* (three items), and *Helpfulness* (four items), the three *Trusting Beliefs for Specific Technology* defined by McKnight et al. [185], each measured on 5-point Likert-type scales ranging from 1 = *strongly disagree* to 5 = *strongly agree*. We drew IoT cards at random from each category to ensure balanced representation. This meant no single device was over- or underrepresented, ensuring all devices were evaluated with comparable

frequency. For each participant, we randomized the order of cards to minimize order effects. Overall, we designed the study to take approximately 25 minutes per participant.

6.5.4 Apparatus

Participants were recruited via Prolific and directed to a Qualtrics survey, where their Prolific ID was recorded automatically. The survey included three attention checks; we screened out participants who failed two or more. Qualtrics handled both IoT card balancing and randomization.

6.5.5 Results

Trust in Specific Technology. We analyzed participants' perceptions of trust using the framework proposed by McKnight et al. [185], employing one-way analyses of covariance (ANCOVA) to account for individual differences in general trust in technology. We derived this trait of general trust from the scale consisting of *Faith in General Technology* and *Trusting Stance—General Technology* by McKnight et al. [185]. We selected ANCOVA over a standard ANOVA because it allows us to control for traits, reduce potential confounding, and increase statistical power when comparing trust across device categories. Using this one-way ANCOVA, controlling for the general trust trait, we compared the effect of our established device categories on the Trusting Belief in a Specific Technology, consisting of *Reliability*, *Functionality*, and *Helpfulness* [185]. We report Means and Standard Deviation in Table 6.2. We observed a significant effect of device category on this Trusting Belief ($F(4, 2099) = 7.18, p < .001$). The covariate, i.e., the general trust trait, was significantly related to Trusting Belief in a Specific Technology ($F(1, 2099) = 353.39, p < .001$). This indicates that participants' pre-existing disposition to trust technology strongly predicted their trust in the specific devices studied. As we show in Figure 6.5, post-hoc analysis using Tukey's HSD [304] revealed that participants perceived *Physical Access Devices* as significantly less trustworthy than any other categor: *Media & Entertainment Devices* ($p < .001$), *Environment Monitoring & Control Devices* ($p < .01$), *Surveillance & Assistance Devices* ($p < .01$), and *Household Convenience Devices* ($p < .05$).

To investigate the components of trust further, we conducted ANCOVAs on the individual measures of the Trusting Belief: *Reliability*, *Functionality*, and *Helpfulness*. While we found no significant effects regarding *Functionality*, they were present for the other two measures. Concerning *Reliability*, there was a significant effect of the device categories ($F(4, 2099) = 8.15, p < .001$). Post-hoc tests indicated that *Physical Access Devices* were rated as significantly less reliable than *Environment Monitoring & Control Devices* ($p < .001$), *Media & Entertainment Devices* ($p < .001$), *Household Convenience Devices* ($p < .01$), and *Surveillance & Assistance Devices* ($p < .01$). We also identified a significant effect of device category on *Helpfulness* ($F(4, 2099) = 3.50, p < .01$). Post-hoc analysis showed that when compared to *Physical Access Devices*, both *Media & Entertainment Devices* ($p < .05$) and *Surveillance & Assistance Devices* ($p < .05$) were perceived as significantly more helpful.

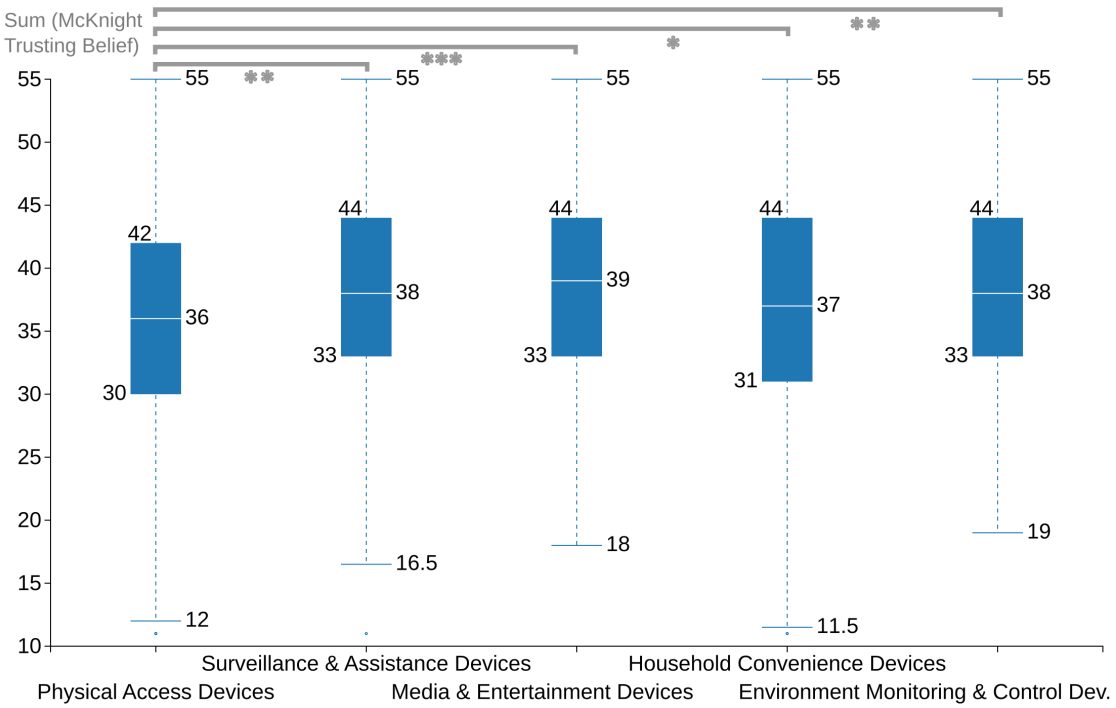


Figure 6.5: Box plots of Trusting Belief scores across device categories, showing lower perceived trust for *Physical Access Devices* compared to other categories.

Table 6.2: Trusting Beliefs (Reliability, Functionality, Helpfulness) and the Overall McKnight Device score across the five device categories. Entries show Mean (SD).

Category	Reliability	Functionality	Helpfulness	Overall
Environment Monitoring & Control Devices	13.77 (3.12)	11.16 (3.28)	13.06 (3.22)	37.99 (7.90)
Household Convenience Devices	13.56 (3.38)	10.73 (3.42)	13.39 (3.42)	37.68 (8.63)
Media & Entertainment Devices	14.01 (3.12)	11.50 (2.98)	13.45 (3.22)	38.96 (7.52)
Physical Access Devices	12.88 (3.46)	10.49 (3.65)	12.84 (3.33)	36.21 (8.79)
Surveillance & Assistance Devices	13.56 (3.29)	10.93 (3.39)	13.48 (3.19)	37.97 (8.17)

Technology Acceptance of IoT Devices. We analyzed perceptions related to the Technology Acceptance Model, specifically *Intention to Use* and *Perceived Risk* as described by Im et al. [141], using one-way repeated measures analyses of variance (ANOVA). We report Means and Standard Deviation in Table 6.3. Repeated measures ANOVA is appropriate as it accounts for the non-independence of observations from the same participant and increases sensitivity to detect differences across conditions. We conducted post-hoc comparisons using Tukey's HSD [304] for all significant main effects.

The ANOVA analysis revealed a significant effect of our device categories on participants' *Intention to Use* specific IoT devices ($F(4, 2100) = 6.40, p < .001$). As can be seen in Figure 6.6, post-hoc comparisons indicated that participants were significantly less willing to use *Media & Entertainment Devices* than *Household Convenience Devices* ($p < .01$).

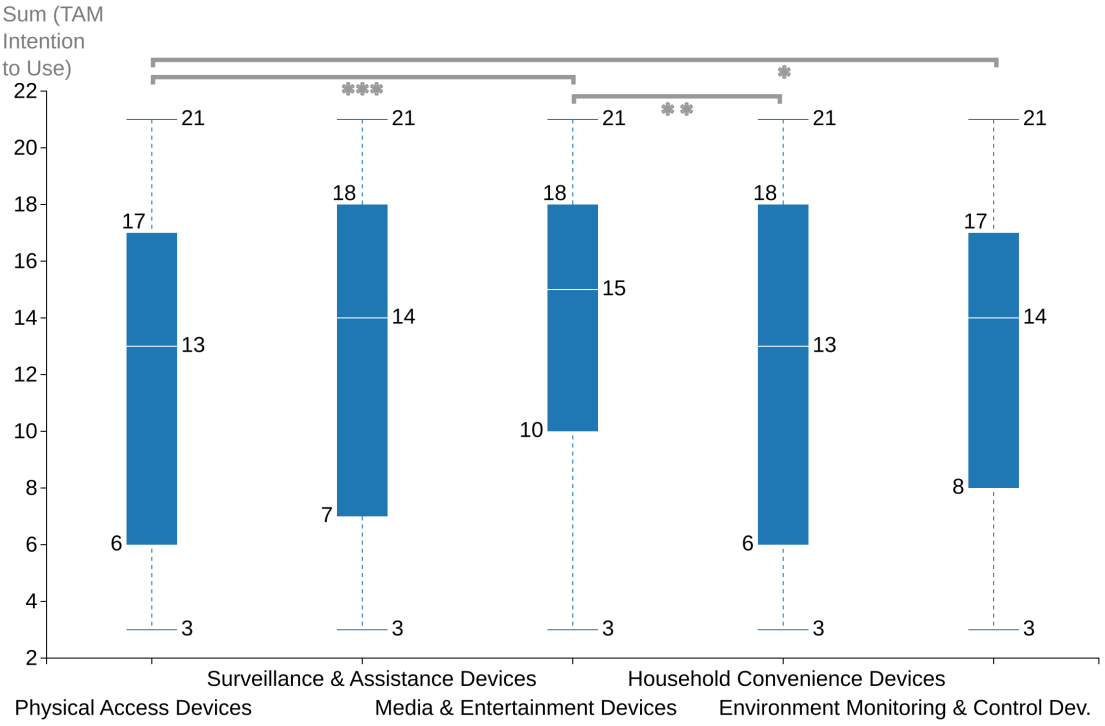


Figure 6.6: Box plots of *Intention to Use* across device categories, illustrating how participants' willingness to adopt devices varies between categories.

They were also less willing to use *Physical Access Devices* than *Media & Entertainment Devices* ($p < .001$) and *Environment Monitoring & Control Devices* ($p < .05$).

A second one-way repeated measures ANOVA examined differences in the factor of *Perceived Risk* across the five categories. This analysis also revealed a significant main effect ($F(4, 2100) = 7.86, p < .001$). As can be seen in Figure 6.7, post-hoc tests with Tukey's HSD highlighted several notable contrasts: *Physical Access Devices* were rated as significantly riskier than Media & Entertainment Devices ($p < .001$), *Surveillance & Assistance Devices* ($p < .001$), and *Environment Monitoring & Control Devices* ($p < .001$). *Household Convenience Devices* were rated as significantly riskier than *Media & Entertainment Devices* ($p < .05$) and *Surveillance & Assistance Devices* ($p < .05$).

6.6 Discussion

Category-Specific Risks. Our analysis shows that *Physical Access Devices* form a unique category: their ability to move through or actuate within private spaces raised perceived risk and lowered trust (lower McKnight's "Trusting Belief" overall; lower *Reliability* and *Helpfulness*) as well as *Intention to Use* relative to other categories. This aligns with security research of IoT ecosystems, as exposed backends increase the risks

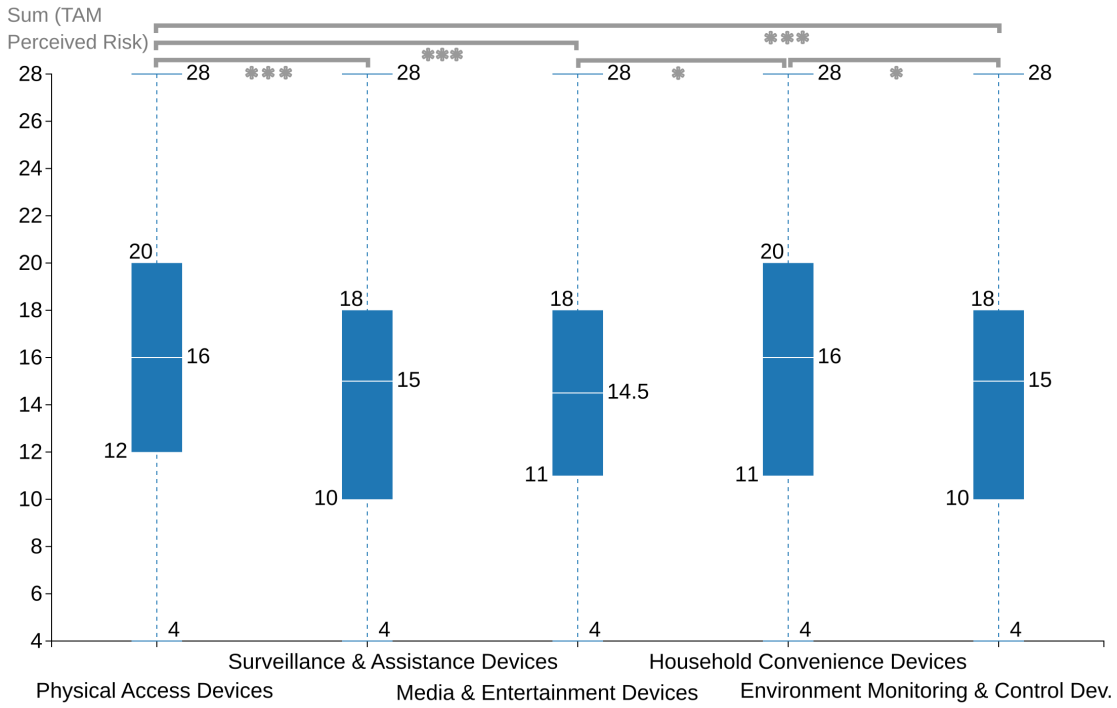


Figure 6.7: Box plots of *Perceived Risk* across device categories, highlighting differences in how risky participants considered each type of device.

Table 6.3: *Intention to Use* and *Perceived Risk* across the five device categories. Entries show Mean (SD).

Category	**Intention to Use**	**Perceived Risk**
Environment Monitoring & Control Devices	13.09 (5.42)	14.45 (5.57)
Household Convenience Devices	12.28 (6.01)	15.49 (5.88)
Media & Entertainment Devices	13.78 (5.35)	14.29 (5.01)
Physical Access Devices	11.98 (5.94)	15.99 (5.83)
Surveillance & Assistance Devices	12.83 (5.92)	14.40 (5.77)

when devices can manipulate the physical environment [301, 302]. It also resonates with HCI's "creepiness" in sensed domestic spaces (e.g., cameras), where surveillance and control raise concern [236]. Conversely, we observed partial evidence that *Media & Entertainment Devices* are perceived as less risky and more adoptable, consistent with their largely output-oriented functionality and with prior findings that some data types (e.g., audio) are less sensitive [106].

Impact of Functionality on Perceived Risk. Functionality played a central role across categories: participants often traded stricter privacy/security for valued utility,

evaluating risk in light of perceived benefits. This echoes the privacy paradox in IoT, users proceed despite risks when utility is compelling [320], and supports evidence that urgency and convenience can win over privacy during repair decisions [23, 252]. Our CySESH results ($M = 4.40$ ($SD = 1.07$)) further nuance this picture: participants felt confident with basic actions (e.g., turning off a device, deleting cloud data) but were less certain about complex, institutional, or opaque tasks (e.g., discovering third-party access, contacting a DPO), indicating where designs should help users [43]. SEBIS scores ($M = 59.86$ ($SD = 8.44$)) similarly suggest strong basics (PINs/passwords) but inconsistent proactive behaviors (screen locking, updating, link checking).

Trust & Adoption. Trust was closely tied to adoption: even after accounting for participants' general trust in technology, device-category differences remained, and higher trust in a specific device predicted stronger *Intention to Use*. This reflects the mixed signals users get from institutions/certifications, creating confidence even when issues remain (e.g., IOXT-certified apps with vulnerabilities) [178]. Our results extend prior work by showing that trust is not uniform across categories but changes with device functionality and physical capabilities.

Roles & Categories. Our results tie with HCI research on roles and context. Prior work shows that owners, guests, and bystanders have different privacy expectations and control levels [106, 180, 181, 182]. Our categorization adds to this by showing that device functionality itself shapes perceptions of risk, trust, and adoption. For example, *Physical Access Devices* raise challenges for authentication and shared household access, where no single method works for all users [186, 240, 241].

Our Contribution. Overall, our five-category framework helps HCI and SecPriv researchers focus on where perceptions of risk, trust, and usefulness differ most. It suggests prioritizing strong safeguards and transparency for *Physical Access Devices*, clearer consent and visibility for high-sensing devices, and better support for everyday security habits where users are confident but inconsistent. This category-based view goes beyond single-device studies by linking functionality to clear differences in trust, risk, and adoption across the smart home.

6.6.1 Limitations

In the following, we discuss limitations. First, the number of experts was relatively small, which may limit the diversity of perspectives. Larger and more heterogeneous samples could further validate the robustness of our categories. Second, the Prolific-based survey relied on self-reports of trust and self-efficacy, which may not reflect real-world behaviors and are subject to social desirability bias or limited device experience. This relates to the "Privacy Paradox," where people report more careful privacy practices than they actually follow [37]. Third, our clustering was constrained by the predefined set of 29 devices, which cannot capture the full IoT landscape. Finally, we did not account for contextual factors such as cultural background, prior experiences, or household composition, which may shape perceptions of security and trust.

6.6.2 Future Work

Future work should involve broader and more diverse expert and user samples, including cross-cultural studies to test whether our categorization generalizes across regions. Extending the framework to emerging domains, such as industrial or health IoT, could provide insights into higher-stakes contexts. Behavioral experiments (e.g., observing device configuration in controlled settings) would strengthen evidence on how trust and self-efficacy translate into practice. Finally, longitudinal studies could capture how perceptions evolve as IoT adoption grows and new security incidents occur.

6.7 Conclusion

In this paper, we examined how users perceive IoT devices' risk, reliability, functionality, and helpfulness, as well as their intention to use them, across categories derived from an expert clustering of 29 devices. The resulting categories were *Physical Access Devices*, *Surveillance & Assistance Devices*, *Media & Entertainment Devices*, *Household Convenience Devices*, and *Environment Monitoring & Control Devices*. We validated these categories through a Prolific study (N = 213 after attention checks). Results showed that *Physical Access Devices* were perceived as particularly risky, while *Media & Entertainment Devices* were seen as less risky and more attractive for adoption. Trust closely aligns with purchase intentions, underscoring its role in IoT uptake. Moreover, our results indicated strong confidence in basic security tasks and consistent use of PINs and passwords, but weaker self-efficacy and behaviors in more complex or proactive practices. These findings demonstrate that risk and trust perceptions are category-dependent, offering a foundation for future IoT security research and design.

CHAPTER 7

Key Takeaways

1. **Hard-coded secrets and open endpoints remain pervasive.** Static analysis of nearly 10,000 Android companion apps, along with Internet-wide scans, revealed thousands of MQTT, CoAP, and XMPP backends accessible with default or embedded credentials, wildcard topics, and outdated TLS. *Mitigation:* (1) avoid plaintext keys, (2) enforce broker/backend authentication by default, and (3) provide one-click "secure" cloud deployment templates.

2. **Legacy protocol versions are still widely adopted.** Two-thirds of TLS-enabled backends still accept TLS v1/v1.1; many remain vulnerable to attacks such as BEAST, Sweet32, or Logjam. *Mitigation:* (1) enforce minimum TLS 1.2 (preferably 1.3) in SDKs, (2) flag weak cipher negotiation, and (3) provide migration guides for resource-constrained devices.

3. **Disclosure pipelines do not scale.** Of 15,046 vulnerable addresses, fewer than 2% were fully remediated after a coordinated campaign; 60% of the emails bounced or failed to deliver. *Mitigation:* (1) require machine-readable `security.txt` endpoints for IoT clouds, (2) provide domain-level abuse contacts with disclosure guidelines, and (3) empower CERTs to act when operators are unresponsive.

4. **Smart-TV broadcast protocols leak before consent.** In 5 EU countries, 72% of HbbTV channels contacted trackers before consent, and 66% still used HTTP. *Mitigation:* (1) integrate HTTPS-only in HbbTV deployments, (2) provide broadcaster tooling that denies third-party pixels, and (3) allow home routers to enforce DNS-level HbbTV blocking until compliance is verified.

5. **User perception skews toward the visible threat, ignoring the ordinary.** A survey of 213 users ranked physical-access devices as the most risky yet underestimated type of entertainment device, reflecting our findings on TVs. *Mitigation:* (1)

introduce standardized "privacy labels," and (2) hub dashboards that show silent but profiling-heavy devices.

6. **Securing IoT needs aligned motivations across the ecosystem.** Findings confirm that vulnerabilities originate in code, propagate through cloud misconfigurations, and ultimately harm unaware end users. *Protection:* (1) secure-by-default libraries, (2) continuous, automated measurement with open data, (3) enforceable liability for cloud operators, and (4) user-centric tooling that converts risks into actionable choices.

CHAPTER 8

Conclusion and Future Work

We investigated insecurity across the IoT ecosystem, from mobile companion applications and backend cloud services to broadcast Smart TV protocols, and connected these technical insights with user perception studies. We revealed several key weaknesses in current IoT practices. First, our large-scale analysis of IoT companion apps uncovered hard-coded credentials and endpoints in 9,889 Android applications (Chapter 2), a pervasive, insecure development practice that exposes sensitive secrets. Second, by leveraging these app findings to inspect IoT backends, we identified 58,175 publicly reachable servers (across protocols such as MQTT, CoAP, and XMPP) that either leaked data, allowed unauthorized or were exposed to DoS threats, underscoring widespread cloud- and infrastructure-level vulnerabilities (Chapter 3 and Chapter 4). Third, in the smart TV domain, we found that 26 of 36 European HbbTV-enabled channels initiated tracking connections *before* obtaining user consent, revealing serious privacy violations in broadcast IoT services (Chapter 5). Finally, a survey of 213 users highlighted a persistent gap between perceived and actual IoT risks: participants consistently underestimated the threats posed by entertainment devices such as smart TVs (Chapter 6). Collectively, these findings illustrate that IoT security shortcomings originate at multiple layers of the ecosystem, from development and deployment through to end use, and ultimately impact end users, who often lack the awareness or means to protect themselves.

These results highlight that IoT insecurity is a *systemic* problem spanning technical and human dimensions. Weaknesses introduced during development and configuration (e.g., hard-coded credentials or open endpoints) propagate through cloud services and devices, eventually compromising user security and privacy at scale. Moreover, the misalignment between actual threats and user perceptions of risk means that the ecosystem suffers from technical flaws and fails to inform consumers. The impact is an IoT landscape where security issues are widespread, user data and privacy are consistently at risk, and existing regulatory frameworks struggle to keep up with the expanding attack surface.

Addressing these challenges requires a comprehensive approach. Investigating emerging IoT standards and platforms for potential vulnerabilities is an important direction on the technical front. For instance, the new *Matter* standard for smart-home interoperability [66] and the upcoming *DVB-I* specification for combined broadcast-broadband TV are expected to reshape their domains. We need security and privacy evaluations of these technologies to ensure they do not repeat past errors or introduce new risks. Reviewing these standards, as Matter adoption grows and DVB-I enables more personalized, addressable TV, helps build stronger protections from the start. In parallel, we need better coordinated vulnerability disclosure (CVD) to address the issues we found. The large number of insecure endpoints across thousands of apps and devices shows that ad-hoc, personalized disclosures do not scale. Future work should develop automated, scalable ways to notify vendors, track fixes over time, and hold parties accountable. Creating standardized CVD procedures, supported by industry groups and regulators, would ensure that discovered IoT vulnerabilities are reported and resolved quickly and consistently across a diverse vendor ecosystem.

Another important research direction is to help users through better design and education, so we close the current awareness gap. Our results show that many people either do not know about IoT risks or underestimate them, especially for devices that seem harmless. To address this, future work should build user-centered tools that make security and privacy easier to see and control. For example, designers can add clear setup prompts that explain key permissions, defaults, and data practices at first use, and provide privacy dashboards that show data flows and permissions across devices in one place.

Finally, because IoT insecurity spans many fields, we need broad collaboration. Improvements will only come if technical experts, human-computer interaction (HCI) researchers, industry, and policymakers work together. Engineers and security researchers should partner with usability experts so that protections are strong and easy to use. At the same time, regulators set clear rules and incentives for security and privacy by design. We can grow IoT without sacrificing fundamental security and privacy rights by aligning solid technical safeguards, user-centric design, and effective policy.

Overview of Generative AI Tools Used

AI-based coding assistants, particularly GitHub Copilot integrated within Visual Studio Code (VSCode), were used to assist with error handling, debugging, and generating graphs relevant to the research. These tools proved helpful during the experimental phases of the study. Additionally, generative AI tools for document editing, such as Grammarly and ChatGPT, were used to perform grammar checks, enhance formatting, and citation management throughout the manuscript. All generative tools were used critically, with outputs carefully reviewed to ensure accuracy, originality, and adherence to academic integrity standards.

Bibliography

[1] @igorsushko. 2022. TV systems have been hacked for May 9th. (May 9, 2022). Retrieved May 11, 2022 from https://twitter.com/igorsushko/status/1523543444127248385.

[2] Y. Aafer, W. You, Y. Sun, Y. Shi, X. Zhang, and H. Yin. 2021. Android SmartTVs Vulnerability Discovery via Log-Guided Fuzzing. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security)*. (Aug. 2021).

[3] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac. 2020. Peek-a-Boo: I See Your Smart Home Activities, Even Encrypted! In *Proceedings of the 13th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WISEC)*. (July 2020). DOI: 10.1145/3395351.3399421.

[4] Adintime. [n. d.] TV Audiences 2021. Retrieved July 28, 2022 from https://adintime.com/en/blog/tv-audiences-2021-n128.

[5] A. A. Al Alsadi, K. Sameshima, J. Bleier, K. Yoshioka, M. Lindorfer, M. Van Eeten, and C. H. Gañán. 2022. No Spring Chicken: Quantifying the Lifespan of Exploits in IoT Malware Using Static and Dynamic Analysis. In *Proceedings of the 17th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*. (May 2022). DOI: 10.1145/3488932.3517408.

[6] N. Alexopoulos, M. Brack, J. P. Wagner, T. Grube, and M. Mühlhäuser. 2022. How Long Do Vulnerabilities Live in the Code? A Large-Scale Empirical Measurement Study on FOSS Vulnerability Lifetimes. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security)*. (Aug. 2022).

[7] Alibaba. 2023. Whitelist DTS IP ranges for your user-created database. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://www.alibabacloud.com/help/en/data-transmission-service/latest/whitelist-dts-ip-ranges-for-your-user-created-database.

[8] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR)*. (May 2016). DOI: 10.1145/2901739.2903508.

[9] M. N. Alraja, M. M. J. Farooque, and B. Khashab. 2019. The effect of security, privacy, familiarity, and trust on users' attitudes toward the use of the iot-based healthcare: the mediation role of risk perception. *IEEE Access*, 7. DOI: 10.1109/ACCESS.2019.2904006.

[10] O. Alrawi, C. Lever, K. Valakuzhy, R. Court, K. Z. Snow, F. Monrose, and M. Antonakakis. 2021. The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security)*. (Aug. 2021).

[11] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *Proceedings of the 40th IEEE Symposium on Security & Privacy (S&P)*. (May 2019). DOI: 10.1109/SP.2019.00013.

[12] Amazon. 2023. AWS IP ranges. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://ip-ranges.amazonaws.com/ip-ranges.json.

[13] C. Amsüss. 2023. aiocoap. (Aug. 31, 2023). Retrieved Oct. 14, 2023 from https://github.com/chrysn/aiocoap.

[14] Android Developers. 2021. Security with network protocols. (Oct. 27, 2021). Retrieved Oct. 7, 2022 from https://developer.android.com/training/articles/security-ssl.

[15] Android Developers. 2022. Android Keystore system. (Oct. 7, 2022). Retrieved July 25, 2023 from https://developer.android.com/training/articles/keystore.

[16] Android Developers. 2023. Best practices for unique identifiers. (May 22, 2023). Retrieved July 25, 2023 from https://developer.android.com/training/articles/user-data-ids.

[17] Android Developers. 2023. Bluetooth permissions. (July 28, 2023). Retrieved July 28, 2023 from https://developer.android.com/guide/topics/connectivity/bluetooth/permissions.

[18] Android Developers. 2023. Version your app. (Apr. 12, 2023). Retrieved Apr. 27, 2023 from https://developer.android.com/studio/publish/versioning.

[19] Android Developers. 2023. Work with data more securely. (July 12, 2023). Retrieved July 25, 2023 from https://developer.android.com/topic/security/data.

[20] Android Developers. [n. d.] TelephonyManager. Retrieved July 13, 2022 from https://developer.android.com/reference/android/telephony/TelephonyManager.

[21] Android Developers. [n. d.] UI/Application Exerciser Monkey. Retrieved Mar. 30, 2022 from https://developer.android.com/studio/test/other-testing-tools/monkey.

[22] S. Andy, B. Rahardjo, and B. Hanindhito. 2017. Attack Scenarios and Security Analysis of MQTT Communication Protocol in IoT System. In *Proceedings of the 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. (Sept. 2017). DOI: 10.1109/EECSI.2017.8239179.

[23] D. Anthony, C. A. Gunter, W. He, M. Khanafer, S. Landau, R. Mangar, and N. Reitinger. 2023. The HandyTech's Coming Between 1 and 4: Privacy Opportunities and Challenges for the IoT Handyperson. In *Proceedings of the 22nd Workshop on Privacy in the Electronic Society (WPES)*. (Nov. 26, 2023). DOI: 10.1145/3603216.3624976.

[24] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. 2017. Understanding the Mirai Botnet. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security)*. (Aug. 2017). Retrieved July 22, 2024 from.

[25] AppBrain. [n. d.] Android network libraries. Retrieved Sept. 19, 2021 from https://www.appbrain.com/stats/libraries/tag/network/android-network-libraries.

[26] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. 2014. FlowDroid: Precise Context, Flow, Field, Object-Sensitive and Lifecycle-Aware Taint Analysis for Android Apps. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. (June 2014). DOI: 10.1145/2594291.2594299.

[27] 2022. Auditel - Sintesi Mensile Ottobre 2022. Retrieved Nov. 23, 2022 from https://www.auditel.it/wp-content/uploads/2022/10/Sintesi-Mensile-Ottobre-2022-ts-cum-7.pdf.

[28] 2022. Auditel - TV Mediaset.it. Retrieved July 28, 2022 from http://www.mediaset.it/auditel/ascolti.shtml.

[29] L. Babun, Z. Berkay Celik, P. McDaniel, and S. Uluagac. 2021. Real-time Analysis of Privacy-(un)aware IoT Applications. In *Proceedings of the 21st Privacy Enhancing Technologies Symposium (PETS)*. (July 2021). DOI: 10.2478/popets-2021-0009.

[30] Y. Bachy, F. Basse, V. Nicomette, E. Alata, M. Kaâniche, J.-C. Courrège, and P. Lukjanenko. 2015. Smart-TV Security Analysis: Practical Experiments. In *Proceedings of the 45th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. (June 2015). DOI: 10.1109/DSN.2015.41.

[31] H. Badran. 2019. IoT Security and Consumer Trust. In *Proceedings of the 20th International Conference on Digital Government Research (dg.o)*. (June 2019). DOI: 10.1145/3325112.3325234.

[32] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan. 2012. The Menlo Report. *IEEE Security & Privacy*, 10, 2, (Mar. 2012). DOI: 10.1109/MSP.2012.52.

[33] M. Balossini and S. Arzt. 2023. GitHub FlowDroid Issue #578 – Flowdroid execution time. (Feb. 16, 2023). Retrieved Apr. 27, 2023 from https://github.com/secure-software-engineering/FlowDroid/issues/578.

[34] A. Banks, E. Briggs, K. Borgendale, and R. Gupta. 2020. MQTT Version 5.0. (Mar. 2020). Retrieved Aug. 22, 2022 from https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[35] A. Banks and R. Gupta. 2014. MQTT Version 3.1.1. (Oct. 2014). Retrieved Aug. 22, 2022 from http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[36] P. Barre, C. Kasmi, and T. Sabono. 2018. COMMSEC: Hacking into Broadband and Broadcast TV Systems. In *Hack in the Box Security Conference (HITBSecConf)*.

[37] S. Barth and M. D. de Jong. 2017. The privacy paradox - investigating discrepancies between expressed privacy concerns and actual online behavior – a systematic literature review. *Telematics and Informatics*, 34, 7. DOI: 10.1016/j.tele.2017.04.013.

[38] P. Beer, L. Veronese, M. Squarcina, and M. Lindorfer. 2022. The Bridge between Web Applications and Mobile Platforms is Still Broken. In *Workshop of Designing Security for the Web (SecWeb)*.

[39] E. Belinski. 2023. Android API Levels. (Sept. 4, 2023). Retrieved Sept. 8, 2023 from https://apilevels.com/.

[40] Better Software Group. 2019. HbbTV: What Is it and How Does it Work? Retrieved Jan. 21, 2021 from https://bsgroup.eu/hbbtv-what-is-it-and-how-does-it-work/.

[41] S. Black. 2022. StevenBlack/hosts. Retrieved May 23, 2022 from https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts.

[42] D. Boffey. 2019. EU Recalls Children's Smartwatch over Data Fears. (Feb. 5, 2019). Retrieved Jan. 30, 2022 from https://www.theguardian.com/technology/2019/feb/05/eu-recalls-childrens-smartwatch-over-data-fears.

[43] N. Borgert, O. D. Reithmaier, L. Jansen, L. Hillemann, I. Hussey, and M. Elson. 2023. Home Is Where the Smart Is: Development and Validation of the Cybersecurity Self-Efficacy in Smart Homes (CySESH) Scale. In *Proceedings of the 2023*. (Apr. 2023). DOI: 10.1145/3544548.3580860.

[44] K. Borgolte, T. Chattopadhyay, N. Feamster, M. Kshirsagar, J. Holland, A. Hounsel, and P. Schmitt. 2019. How DNS over HTTPS is Reshaping Privacy, Performance, and Policy in the Internet Ecosystem. In *Proceedings of the 47th Research Conference on Communications, Information and Internet Policy (TPRC)*. (Sept. 2019). DOI: 10.2139/ssrn.3427563.

[45] K. Borgolte, T. Fiebig, S. Hao, C. Kruegel, and G. Vigna. 2018. Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2018). DOI: 10.14722/ndss.2018.23327.

[46] K. Borgolte, S. Hao, T. Fiebig, and G. Vigna. 2018. Enumerating Active IPv6 Hosts for Large-scale Security Scans via DNSSEC-signed Reverse Zones. In *Proceedings of the 39th IEEE Symposium on Security & Privacy (S&P)*. (May 2018). DOI: 10.1109/SP.2018.00027.

[47] C. Bormann, Z. Shelby, and K. Hartke. 2014. The Constrained Application Protocol (CoAP). RFC 7252. (June 2014). https://www.rfc-editor.org/info/rfc7252.

[48] M. Bozza. 2019. Catch the wave - Hijack HbbTV. Codemotion. Retrieved Sept. 25, 2025 from https://www.youtube.com/watch?v=2yeahbhPu9o.

[49] I. Butun, P. Österberg, and H. Song. 2020. Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures. *IEEE Communications Surveys & Tutorials*, 22, (Jan. 2020), 1, (Jan. 2020). DOI: 10.1109/COMST.2019.2953364.

[50] P. Cabrera Camara. 2019. SDR Against Smart TVs: URL and Channel Injection Attacks. DEFCON 27. Retrieved May 21, 2022 from https://infocondb.org/con/def-con/def-con-27/sdr-against-smart-tvs-url-and-channel-injection-attacks.

[51] 2023. California Privacy Rights Act (CPRA). Retrieved Dec. 2, 2023 from https://cppa.ca.gov/regulations/.

[52] M. L. Carlotta Tagliaro Florian Hahn. 2022. Full survey (in English and Italian). Retrieved Nov. 29, 2022 from https://docs.google.com/document/d/1UIsXN0-ihWwqGl4Lju0lEwSbAHvhFG8aniCgLV4m484/.

[53] P. Carter, C. Mulliner, M. Lindorfer, W. Robertson, and E. Kirda. 2016. CuriousDroid: Automated User Interface Interaction for Android Application Analysis Sandboxes. In *International Conference on Financial Cryptography and Data Security (FC)*. (Feb. 2016). DOI: 10.1007/978-3-662-54970-4_13.

[54] Castoola. 2022. Castoola+ Adding value to TV ads. Castoola+. Retrieved May 23, 2022 from https://castoola.com/.

[55] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. McDaniel, and A. S. Uluagac. 2018. Sensitive Information Tracking in Commodity IoT. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security)*. (Aug. 2018). DOI: 10.5555/3277203.3277329.

[56] 2023. CERT.at - Computer Emergency Response Team Austria. Retrieved Jan. 16, 2023 from https://cert.at/en/.

[57] O. Çetin, C. Gañán, M. Korczyński, and M. van Eeten. 2017. Make Notifications Great Again: Learning How to Notify in the Age of Large-Scale Vulnerability Scanning. In *Proceedings of the 16th Workshop on the Economics of Information Security (WEIS)*. (June 2017).

[58] J. Chen, W. Diao, Q. Zhao, C. Zuo, Z. Lin, X. Wang, W. C. Lau, M. Sun, R. Yang, and K. Zhang. 2018. IoTFuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2018). DOI: 10.14722/ndss.2018.23159.

[59] T.-H. Chen, C. Tagliaro, M. Lindorfer, K. Borgolte, and J. van der Ham-de Vos. 2024. Are You Sure You Want To Do Coordinated Vulnerability Disclosure? In *Proceedings of the 9th International Workshop on Traffic Measurements for Cybersecurity (WTMC)*. (July 2024). DOI: 10.1109/EuroSPW61312.2024.00039.

[60] E. Chin and D. A. Wagner. 2013. Bifocals: Analyzing WebView Vulnerabilities in Android Applications. In *Proceedings of the 14th International Workshop on Information Security Applications (WISA)*. (Aug. 2013). DOI: 10.1007/978-3-319-05149-9_9.

[61] E. Chong. 2018. The Growing Trend of Coin Miner JavaScript Infection. (Feb. 8, 2018). Retrieved Feb. 5, 2021 from https://www.fortinet.com/blog/threat-research/the-growing-trend-of-coin-miner-javascript-infection.html.

[62] S. R. Choudhary, A. Gorla, and A. Orso. 2015. Automated Test Input Generation for Android: Are We There Yet? In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. (Nov. 2015). DOI: 10.1109/ASE.2015.89.

[63] G. Chu, N. Apthorpe, and N. Feamster. 2019. Security and Privacy Analyses of Internet of Things Children's Toys. *IEEE Internet of Things Journal*, 6, 1, (Aug. 2019). DOI: 10.1109/JIOT.2018.2866423.

[64] K. Chung. 2023. Taking over a Dead IoT Company. (Jan. 9, 2023). Retrieved Feb. 5, 2023 from https://blog.kchung.co/taking-over-a-dead-iot-company/.

[65] Cloudflare. 2023. IP Ranges. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://www.cloudflare.com/ips/.

[66] Connectivity Standards Alliance. 2025. Matter | Connectivity Standards Alliance. Retrieved Oct. 3, 2025 from https://csa-iot.org/all-solutions/matter/.

[67] A. Continella, Y. Fratantonio, M. Lindorfer, A. Puccetti, A. Zand, C. Kruegel, and G. Vigna. 2017. Obfuscation-Resilient Privacy Leak Detection for Mobile Apps Through Differential Analysis. In *Proceedings of the 24th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2017). DOI: 10.14722/ndss.2017.23465.

[68] A. Continella, M. Polino, M. Pogliani, and S. Zanero. 2018. There's a Hole in that Bucket! A Large-scale Analysis of Misconfigured S3 Buckets. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC)*. (Dec. 2018). DOI: 10.1145/3274694.3274736.

[69] A. Cortesi, M. Hils, and raumfresser. 2023. mitmproxy. (Oct. 9, 2023). Retrieved Oct. 14, 2023 from https://mitmproxy.org/.

[70] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti. 2014. A Large-scale Analysis of the Security of Embedded Firmwares. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*. (Aug. 2014). DOI: 10.5555/2671225.2671232.

[71] Covertness. 2023. coap-rs. (Sept. 30, 2023). Retrieved Oct. 14, 2023 from https://github.com/Covertness/coap-rs.

[72] K. Cullen. 2021. FreeCoAP. (July 1, 2021). Retrieved Oct. 14, 2023 from https://github.com/keith-cullen/FreeCoAP.

[73] B. Custers, F. Dechesne, A. M. Sears, T. Tani, and S. van der Hof. 2018. A comparison of data protection legislation and policies across the eu. *Computer Law & Security Review*, 34, 2. DOI: 10.1016/j.clsr.2017.09.001.

[74] 2019. CVE-2019-19781. (Dec. 13, 2019). Retrieved May 19, 2024 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-19781.

[75] M. Dahlmanns, J. Lohmöller, J. Pennekamp, J. Bodenhausen, K. Wehrle, and M. Henze. 2022. Missed Opportunities: Measuring the Untapped TLS Support in the Industrial Internet of Things. In *Proceedings of the 17th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*. (May 2022). DOI: 10.1145/3488932.3497762.

[76] A. Davanian, M. Faloutsos, and M. Lindorfer. 2024. C2Miner: Tricking IoT Malware into Revealing Live Command & Control Servers. In *Proceedings of the 19th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*. (July 2024). DOI: 10.1145/3634737.3644992.

[77] M. J. Davino, L. Melo, H. Lu, M. d'Amorim, and A. Prakash. 2019. A Study of Vulnerability Analysis of Popular Smart Devices Through Their Companion Apps. In *Proceedings of the 3rd Workshop on the Internet of Safe Things (SafeThings)*. (May 23, 2019). DOI: 10.1109/SPW.2019.00042.

[78] A. Desnos, G. Gueguen, S. Bachmann, and contributors. 2020. GitHub – Androguard - Android Permissions. (July 3, 2020). Retrieved May 2, 2023 from https://github.com/androguard/androguard/tree/master/androguard/core/api_specific_resources/aosp_permissions.

[79] F. Dhia and M. Dacier. 2021. Zero Conf Protocols and Their Numerous Man in the Middle (MITM) Attacks. In *Proceedings of the 15th IEEE Workshop on Offensive Technologies (WOOT)*. (May 2021). DOI: 10.1109/SPW53761.2021.00060.

[80] C. Dietrich, K. Krombholz, K. Borgolte, and T. Fiebig. 2018. Investigating Operators' Perspective on Security Misconfigurations. In *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Oct. 2018). DOI: 10.1145/3243734.3243794.

[81] Digital Ocean. 2023. Digital Ocean Cloud IP ranges. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://digitalocean.com/geo/google.csv.

[82] A. Dilawer, D. Anupam, and Z. Fareed. 2022. Analyzing the Feasibility and Generalizability of Fingerprinting Internet of Things Devices. In *Proceedings of the 22nd Privacy Enhancing Technologies Symposium (PETS)*. (July 2022). DOI: 10.2478/popets-2022-0057.

[83] D. Dragomir, L. Gheorghe, S. Costea, and A. Radovici. 2016. A Survey on Secure Communication Protocols for IoT Systems. In *Proceedings of the 3rd International Workshop on Secure Internet of Things (SIoT)*. (Sept. 2016). DOI: 10.1109/SIoT.2016.012.

[84] EasyList Authors. 2022. EasyList. Retrieved May 23, 2022 from https://easylist.to/.

[85] Eclipse Foundation. 2020. IoT Developer Survey 2020. (Oct. 2020). Retrieved Nov. 15, 2021 from https://outreach.eclipse.foundation/eclipse-iot-developer-survey-2020.

[86] S. Egelman and E. Peer. 2015. Scaling the Security Wall: Developing a Security Behavior Intentions Scale (SeBIS). In *Proceedings of the 34th ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*. (Apr. 2015). DOI: 10.1145/2702123.2702249.

[87] P. Emami-Naeini, J. Dheenadhayalan, Y. Agarwal, and L. F. Cranor. 2021. Which Privacy and Security Attributes Most Impact Consumers' Risk Perception and Willingness to Purchase IoT Devices? In *Proceedings of the 42nd IEEE Symposium on Security & Privacy (S&P)*. (May 2021). DOI: 10.1109/SP40001.2021.00112.

[88] P. Emami-Naeini, J. Dheenadhayalan, Y. Agarwal, and L. F. Cranor. 2023. Are Consumers Willing to Pay for Security and Privacy of IoT Devices? In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security)*. (Aug. 2023).

[89] Equifax. 2022. Addressable TV - Business - Equifax. Retrieved May 23, 2022 from https://www.equifax.com/business/product/addressable-tv/.

[90] I. Erben. [n. d.] Country CIDR IP Ranges. Retrieved Jan. 25, 2022 from http://www.iwik.org/ipcountry/.

[91] ETSI. [n. d.] Consumer IoT security. Retrieved Feb. 1, 2023 from https://www.etsi.org/technologies/consumer-iot-security.

[92] European Audiovisual Observatory. 2016. IRIS Special - Smart TV and Data Protection. Retrieved May 23, 2022 from https://book.coe.int/en/european-audiovisual-observatory/6863-iris-special-smart-tv-and-data-protection.html.

[93] European Parliament and the Council of the European Union. 2016. General Data Protection Regulation (GDPR). *Official Journal of the European Union*, 59, (May 4, 2016), L119, (May 4, 2016). Retrieved July 5, 2021 from http://data.europa.eu/eli/reg/2016/679/oj.

[94] European Telecommunications Standards Institute (ETSI). 2018. ETSI TS 102 796 V1.5.1, Hybrid Broadcast Broadband TV. Standard. (Sept. 2018). https://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.05.01_60/ts_102796v010501p.pdf.

[95] European Union Agency for Fundamental Rights. 2020. Your rights matter: Data protection and privacy. Retrieved Sept. 25, 2025 from https://fra.europa.eu/en/publication/2020/fundamental-rights-survey-data-protection.

[96] Exodus. [n. d.] Trackers. Retrieved Feb. 23, 2022 from https://reports.exodus-privacy.eu.org/en/trackers/.

[97] A. A. Fadele, M. Othman, I. A. T. Hashem, and F. Alotaibi. 2017. Internet of Things security: A survey. *Journal of Network and Computer Applications*, 88, (June 15, 2017). DOI: 10.1016/j.jnca.2017.04.002.

[98] X. Feng, X. Zhu, Q.-L. Han, W. Zhou, S. Wen, and Y. Xiang. 2023. Detecting Vulnerability on IoT Device Firmware: A Survey. *IEEE/CAA Journal of Automatica Sinica*, 10, 1. DOI: 10.1109/JAS.2022.105860.

[99] C. Fenton and contributors. 2020. GitHub – APKiD. (Nov. 12, 2020). Retrieved Jan. 29, 2022 from https://github.com/rednaga/APKiD.

[100] E. Fernandes, J. Jung, and A. Prakash. 2016. Security Analysis of Emerging Smart Home Applications. In *Proceedings of the 1st IEEE European Symposium on Security & Privacy (EuroS&P)*. (Mar. 2016). DOI: 10.1109/SP.2016.44.

[101] S. Fernandez, O. Hureau, A. Duda, and M. Korczynski. 2024. WHOIS Right? An Analysis of WHOIS and RDAP Consistency. In *Proceedings of the 25th Passive and Active Measurement Conference (PAM)*. (Mar. 2024). DOI: 10.1007/978-3-031-56249-5_9.

[102] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke. 2022. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. *IEEE Access*, 10, (July 12, 2022). DOI: 10.1109/ACCESS.2022.3165809.

[103] Forum of Incident Response and Security Teams (FIRST). 2020. Guidelines and Practices for Multi-Party Vulnerability Coordination and Disclosure. (June 19, 2020). Retrieved May 19, 2024 from https://www.first.org/global/sigs/vulnerability-coordination/multiparty/guidelines-v1.1.

[104] I. Fouad, N. Bielova, A. Legout, and N. Sarafijanovic-Djukic. 2020. Missed by Filter Lists: Detecting Unknown Third-Party Trackers with Invisible Pixels. In *Proceedings of the 20th Privacy Enhancing Technologies Symposium (PETS)*. (July 2020). DOI: 10.2478/popets-2020-0038.

[105] E. Foudil and Y. Shafranovich. 2022. A File Format to Aid in Security Vulnerability Disclosure. Tech. rep. 9116. Request for Comments (RFC), (Apr. 2022). DOI: 10.17487/RFC9116.

[106] A. Frik, X. Zhan, N. Abdi, and J. Bernd. 2025. Who Cares? Contextual Privacy Judgments from Owner and Bystander Perspectives in Different Smart Home Situations. In *Proceedings of the 25th Privacy Enhancing Technologies Symposium (PETS)*. (July 2025). DOI: 10.56553/popets-2025-0091.

[107] P. Gadient, M. Ghafari, M.-A. Tarnutzer, and O. Nierstrasz. 2020. Web APIs in Android through the Lens of Security. In *Proceedings of the 27th IEEE International Conference on Software*

Analysis, Evolution and Reengineering (SANER). (Feb. 2020). DOI: 10.1109/SANER48275.2020.9054850.

[108] 2022. Garante per la Protezione dei Dati Personali. Retrieved May 30, 2022 from https://www.garanteprivacy.it/.

[109] M. Ghiglieri. 2014. I Know What You Watched Last Sunday - A New Survey Of Privacy In HbbTV. In *Proceedings of the IEEE Web 2.0 Security & Privacy Workshop (W2SP)*.

[110] M. Ghiglieri and E. Tews. 2014. A Privacy Protection System for HbbTV in Smart TVs. In *Proceedings of the 11th IEEE Consumer Communications and Networking Conference (CCNC)*. (Jan. 2014). DOI: 10.1109/CCNC.2014.6866595.

[111] M. Ghiglieri, M. Volkamer, and K. Renaud. 2017. Exploring Consumers' Attitudes of Smart TV Related Privacy Risks. In *Proceedings of the 5th International Conference on Human Aspects of Information Security, Privacy and Trust (HAS)*. (July 2017). DOI: 10.1007/978-3-319-58460-7_45.

[112] M. Ghiglieri and M. Waidner. 2016. HbbTV Security and Privacy: Issues and Challenges. *IEEE Security & Privacy*, 14, 3. DOI: 10.1109/MSP.2016.54.

[113] R. S. Girons. 2016. HbbTV Country Review. *HbbTV Symposium*. https://www.hbbtv.org/wp-content/uploads/2017/01/Regis-Saint-Girons-Country-Review-HbbTV-Symposium-2016.pdf.

[114] Google. 2023. Google Cloud IP ranges. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://www.gstatic.com/ipranges/cloud.json.

[115] Google. [n. d.] User Data. Retrieved July 25, 2023 from https://support.google.com/googleplay/android-developer/answer/10144311.

[116] J. Greig. 2022. Microsoft attributes alleged Chinese attack on Indian power grid to 'Boa' IoT vulnerability. (Nov. 25, 2022). Retrieved Nov. 20, 2023 from https://therecord.media/microsoft-attributes-alleged-chinese-attack-on-indian-power-grid-to-boa-iot-vulnerability.

[117] D. Gultsch and R. Raj. 2023. XMPP Compliance Tester. (July 22, 2023). Retrieved Oct. 14, 2023 from https://web.archive.org/web/20230417083716/https://github.com/iNPUTmice/caas.

[118] C. Han, I. Reyes, Á. Feal, J. Reardon, P. Wijesekera, N. Vallina-Rodriguez, A. Elazari, K. A. Bamberger, and S. Egelman. 2020. The Price is (Not) Right: Comparing Privacy in Free and Paid Apps. In *Proceedings of the 20th Privacy Enhancing Technologies Symposium (PETS)*. (July 2020). DOI: 10.2478/popets-2020-0050.

[119] haxrob. 2023. Twitter - HaxRob - Lightbulb App. (July 5, 2023). Retrieved July 25, 2023 from https://twitter.com/haxrob/status/1676424071452708864.

[120] HbbTV Association. 2020. HbbTV 2.0.3 Specification. Standard. (Oct. 2020). https://www.hbbtv.org/wp-content/uploads/2020/10/HbbTV-SPEC-00525-HbbTV-SPEC-00515-008-hbbtv203_2020_10_14.pdf.

[121] HbbTV Association. 2025. Deployments | HbbTV. (June 23, 2025). Retrieved June 23, 2025 from https://www.hbbtv.org/deployments/.

[122] HbbTV Association. 2025. HbbTV Overview. (June 23, 2025). Retrieved June 23, 2025 from https://www.hbbtv.org/overview/.

[123] Y. He, L. Zhang, Z. Yang, Y. Cao, K. Lian, S. Li, W. Yang, Z. Zhang, M. Yang, Y. Zhang, and H. Duan. 2020. TextExerciser: Feedback-driven Text Input Exercising for Android Applications. In *Proceedings of the 41st IEEE Symposium on Security & Privacy (S&P)*. (May 2020). DOI: 10.1109/SP40000.2020.00071.

[124] M. Herfurt. 2013. Security concerns with HbbTV. (June 2013). Retrieved Sept. 25, 2025 from https://www.researchgate.net/publication/277007241_Security_concerns_with_HbbTV.

[125] 2022. HiDes UT-100c. Retrieved Nov. 29, 2022 from http://www.hides.com.tw/product_cg74469_eng.html.

[126] Hkamran80. 2021. Pi-hole Blocklist for Smart TVs. Retrieved May 23, 2022 from https://gist.github.com/hkamran80/779019103fcd306979411d44c8d38459.

[127] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner. 2016. Smart Locks: Lessons for Securing Commodity Internet of Things Devices. In *Proceedings of the 11th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*. (Apr. 2016). DOI: 10.1145/2897845.2897886.

[128] Pi-hole Project Developers. 2022. Pi-hole. Retrieved May 30, 2022 from https://docs.pi-hole.net/.

[129] S. Hollister. 2023. Anker Finally Comes Clean about its Eufy Security Cameras. (Jan. 2023). Retrieved Aug. 22, 2024 from https://www.theverge.com/23573362/anker-eufy-security-camera-answers-encryption.

[130] A. Holst. 2021. IoT connected devices worldwide 2019-2030. (Oct. 19, 2021). Retrieved Jan. 15, 2022 from https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[131] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster. 2020. Comparing the Effects of DNS, DoT, and DoH on Web Performance. In *Proceedings of the 29th The Web Conference (WebConf; previously WWW)*. (Apr. 2020). DOI: 10.1145/3366423.3380139.

[132] A. D. Householder and J. Spring. 2022. Are We Skillful or Just Lucky? Interpreting the Possible Histories of Vulnerability Disclosures. *Digital Threats: Research and Practice*, 3, 4, (Feb. 7, 2022). DOI: 10.1145/3477431.

[133] D. Huang, N. Apthorpe, G. Acar, F. Li, and N. Feamster. 2020. IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale. In *Proceedings of the ACM International Joint Conference on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT/Ubicomp)*. (June 2020). DOI: 10.1145/3397333.

[134] J. Huggler. 2017. Germany bans internet-connected dolls over fears hackers could target children. (Feb. 17, 2017). Retrieved Nov. 20, 2023 from https://www.telegraph.co.uk/news/2017/02/17/germany-bans-internet-connected-dolls-fears-hackers-could-target.

[135] T. Hunt. 2015. When children are breached – inside the massive VTech hack. (Nov. 18, 2015). Retrieved Nov. 20, 2023 from https://www.troyhunt.com/when-children-are-breached-inside/.

[136] T. Hunt. 2017. Data from connected CloudPets teddy bears leaked and ransomed, exposing kids' voice messages. (Feb. 28, 2017). Retrieved Nov. 20, 2023 from https://www.troyhunt.com/data-from-connected-cloudpets-teddy-bears-leaked-and-ransomed-exposing-kids-voice-messages/.

[137] IBM. 2023. CRAM-MD5 and DIGEST-MD5 authentication. Retrieved Jan. 23, 2023 from https://www.ibm.com/docs/en/zos/2.1.0?topic=use-cram-md5-digest-md5-authentication.

[138] IBM. 2023. IBM Cloud IP ranges. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://cloud.ibm.com/docs/cloud-infrastructure?topic=cloud-infrastructure-ibm-cloud-ip-ranges.

[139] M. Ibrahim, A. Continella, and A. Bianchi. 2023. AoT - Attack on Things: A security analysis of IoT firmware updates. In *Proceedings of the 8th IEEE European Symposium on Security & Privacy (EuroS&P)*. (July 2023). DOI: 10.1109/EuroSP57164.2023.00065.

[140] IFTTT. [n. d.] If This Then That – Connect Your Apps. Retrieved July 24, 2023 from https://ifttt.com/.

[141] I. Im, Y. Kim, and H.-J. Han. 2008. The effects of perceived risk and technology type on users' acceptance of technologies. *Information & Management*, 45, 1. DOI: 10.1016/j.im.2007.03.005.

[142] Internet Assigned Numbers Authority (IANA). 2022. Uniform Resource Identifier (URI) Schemes. (Jan. 28, 2022). Retrieved Jan. 30, 2022 from https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml.

[143] Internet Engineering Task Force (IETF). [n. d.] Authentication and Authorization for Constrained Environments (ACE). Retrieved Oct. 21, 2023 from https://datatracker.ietf.org/wg/ace/about/.

[144] K. Irion and N. Helberger. 2017. Smart tv and the online media sector: user privacy in view of changing market realities. *Telecommunications Policy*, 41, 3. DOI: 10.1016/j.telpol.2016.12.013.

[145] ISO/IEC JTC 1/SC 27. 2018. ISO/IEC 29147:2018 Information technology – Security techniques – Vulnerability disclosure. Tech. rep. 29147. International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), (Oct. 2018). Retrieved May 19, 2024 from https://www.iso.org/standard/72311.html.

[146] ISO/IEC JTC 1/SC 27. 2019. ISO/IEC 30111:2019 Information technology - Security techniques - Vulnerability disclosure. Tech. rep. 30111. International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), (Oct. 2019). Retrieved May 19, 2024 from https://www.iso.org/standard/69725.html.

[147] C. Jensen, C. Potts, and C. Jensen. 2005. Privacy practices of Internet Users: Self-reports versus Observed Behavior. *International Journal of Human-Computer Studies*, 63, 1. DOI: 10.1016/j.ijhcs.2005.04.019.

[148] Y. Jia, L. Xing, Y. Mao, D. Zhao, X. Wang, S. Zhao, and Y. Zhang. 2020. Burglars' IoT Paradise: Understanding and Mitigating Security Risks of General Messaging Protocols on IoT Clouds. In *Proceedings of the 41st IEEE Symposium on Security & Privacy (S&P)*. (May 2020). DOI: 10.1109/SP40000.2020.00051.

[149] X. Jin, S. Manandhar, K. Kafle, Z. Lin, and A. Nadkarni. 2022. Understanding IoT Security from a Market-Scale Perspective. In *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Nov. 2022). DOI: 10.1145/3548606.3560640.

[150] Z. Jin, L. Xing, Y. Fang, Y. Jia, B. Yuan, and Q. Liu. 2022. P-Verifier: Understanding and Mitigating Security Risks in Cloud-Based IoT Access Policies. In *Proceedings of the 29th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Nov. 2022). DOI: 10.1145/3548606.3560680.

[151] A. S. John. 2022. AWS, Azure and Google together account for 66% of cloud market. (Oct. 31, 2022). Retrieved Feb. 2, 2023 from https://wire19.com/amazon-microsoft-and-google-cloud-infrastructure-market/.

[152] P. Kijewski. 2023. Threat Activity and Vulnerabilities in Indonesia, Malaysia, Philippines, and Thailand. (June 15, 2023). Retrieved Nov. 15, 2023 from https://blog.apnic.net/2023/06/15/threat-activity-and-vulnerabilities-in-indonesia-malaysia-philippines-and-thailand/.

[153] J. Kim, H. Choi, H. Namkung, W. Choi, B. Choi, H. Hong, Y. Kim, J. Lee, and D. Han. 2016. Enabling Automatic Protocol Behavior Analysis for Android Applications. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. (Dec. 2016). DOI: 10.1145/2999572.2999596.

[154] Kiminewt. 2022. Pyshark. Retrieved Nov. 29, 2022 from https://kiminewt.github.io/pyshark/.

[155] R. K. Konoth, E. Vineti, V. Moonsamy, M. Lindorfer, C. Kruegel, H. Bos, and G. Vigna. 2018. MineSweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense. In *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Oct. 2018). DOI: 10.1145/3243734.3243858.

[156] D. Kuchhal and F. Li. 2021. Knock and Talk: Investigating Local Network Communications on Websites. In *Proceedings of the 21st Internet Measurement Conference (IMC)*. (Nov. 2021). DOI: 10.1145/3487552.3487857.

[157] V. Kulikov. [n. d.] xmpp-info. Retrieved Apr. 20, 2022 from https://nmap.org/nsedoc/scripts/xmpp-info.html.

[158] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, and Z. Durumeric. 2019. All Things Considered: An Analysis of IoT Devices on Home Networks. In *Proceedings of the*

28th USENIX Security Symposium (USENIX Security). (Aug. 2019). DOI: 10.5555/3361338.3361419.

[159] P. Lamkin. 2019. Report Claims Ring Employees Had Unfettered Access To Security Camera Footage. (Jan. 11, 2019). Retrieved July 30, 2021 from https://www.forbes.com/sites/paullamkin/2019/01/11/report-claims-ring-employees-had-unfettered-access-to-security-camera-footage/.

[160] S. Larson. 2017. FDA confirms that St. Jude's cardiac devices can be hacked. (Jan. 9, 2017). Retrieved Jan. 23, 2023 from https://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/.

[161] M. Laurer, W. van Atteveldt, A. Casas, and K. Welbers. Building Efficient Universal Classifiers with Natural Language Inference. (Dec. 29, 2023). arXiv: 2312.17543 [cs.CL].

[162] E. Lear, R. Droms, and D. Romascanu. 2019. Manufacturer Usage Description Specification. RFC 8520. (Mar. 2019). https://datatracker.ietf.org/doc/rfc8520.

[163] S. Leffel, K. Muller, and J. Kappel. 2020. Addressable TV AAdvertising. https://smartclip.tv/addressable-tv-advertising-white-paper/.

[164] T. Lelégard. 2022. TSDuck. Retrieved Nov. 29, 2022 from https://tsduck.io/.

[165] F. Li, Z. Durumeric, J. Czyz, M. Karami, M. Bailey, D. McCoy, S. Savage, and V. Paxson. 2016. You've Got Vulnerability: Exploring Effective Vulnerability Notifications. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security)*. (Aug. 2016).

[166] F. Li and V. Paxson. 2017. A Large-Scale Empirical Study of Security Patches. In *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Oct. 2017). DOI: 10.1145/3133956.3134072.

[167] L. Li, A. Bartel, T. F. Bissyandé, J. Klein, Y. Le Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, and P. McDaniel. 2015. IccTA: Detecting Inter-Component Privacy Leaks in Android Apps. In *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering (ICSE)*. (May 2015). DOI: 10.1109/ICSE.2015.48.

[168] R. Light. 2021. Paho MQTT. v1.6.1. (Oct. 21, 2021). Retrieved Oct. 14, 2023 from https://pypi.org/project/paho-mqtt/.

[169] R. Light. [n. d.] mosquitto. Retrieved Jan. 7, 2023 from https://mosquitto.org/.

[170] M. Lindorfer, M. Neugschwandtner, and C. Platzer. 2015. Marvin: Efficient and Comprehensive Mobile App Classification Through Static and Dynamic Analysis. In *Annual International Computers, Software & Applications Conference (COMPSAC)*. (July 2015). DOI: 10.1109/COMPSAC.2015.103.

[171] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. van der Veen, and C. Platzer. 2014. Andrubis - 1,000,000 Apps Later: A View on Current Android Malware Behaviors. In *International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. (Sept. 2014). DOI: 10.1109/BADGERS.2014.7.

[172] Linux man-pages project. 2022. iptables. Retrieved Nov. 29, 2022 from https://linux.die.net/man/8/iptables.

[173] M. Luo, P. Laperdrix, N. Honarmand, and N. Nikiforakis. 2019. Time Does Not Heal All Wounds: A Longitudinal Analysis of Security-Mechanism Support in Mobile Browsers. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2019). DOI: 10.14722/ndss.2019.23149.

[174] G. Lyon. 2019. Nmap: the Network Mapper. v7.80. (Aug. 10, 2019). Retrieved Mar. 30, 2022 from https://nmap.org/.

[175] K. Macnish and J. van der Ham. 2020. Ethics in cybersecurity research and practice. *Technology in Society*, 63, (Oct. 2020). DOI: 10.1016/j.techsoc.2020.101382.

[176] F. Maggi, R. Vossler, and D. Quarta. 2018. The Fragility of Industrial IoT's Data Backbone. Security and Privacy Issues in MQTT and CoAP Protocols. (Dec. 4, 2018). Retrieved Aug. 21, 2021 from https://documents.trendmicro.com/assets/white_papers/wp-the-fragility-of-industrial-IoTs-data-backbone.pdf.

[177] N. Malkin, J. Bernd, M. Johnson, and S. Egelman. 2018. "What Can't Data Be Used For?": Privacy Expectations about Smart TVs in the U.S. In *Proceedings of the 3rd European Workshop on Usable Security (EuroUSEC)*. (Apr. 2018). DOI: 10.14722/eurousec.2018.23016.

[178] P. Mandal, A. S. Ami, V. Olaiya, S. H. Razmjo, and A. Nadkarni. 2024. "Belt and suspenders" or "just red tape"?: Investigating Early Artifacts and User Perceptions of IoT App Security Certification. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security)*. (Aug. 2024).

[179] A. M. Mandalari, D. J. Dubois, R. Kolcun, M. T. Paracha, H. Haddadi, and D. Choffnes. 2021. Blocking Without Breaking: Identification and Mitigation of Non-Essential IoT Traffic. In *Proceedings of the 21st Privacy Enhancing Technologies Symposium (PETS)*. (July 2021). DOI: 10.2478/popets-2021-0075.

[180] K. Marky, N. Gerber, M. G. Pelzer, M. Khamis, and M. Mühlhäuser. 2022. "You offer privacy like you offer tea": Investigating Mechanisms for Improving Guest Privacy in IoT-Equipped Households. In *Proceedings of the 22nd Privacy Enhancing Technologies Symposium (PETS)*. (July 2022). DOI: 10.56553/popets-2022-0115.

[181] K. Marky, S. Prange, F. Krell, M. Mühlhäuser, and F. Alt. 2020. "You just can't know about everything": Privacy Perceptions of Smart Home Visitors. In *Proceedings of the 19th International Conference on Mobile and Ubiquitous Multimedia (MUM)*. (Nov. 2020). DOI: 10.1145/3428361.3428464.

[182] K. Marky, S. Prange, M. Mühlhäuser, and F. Alt. 2022. Roles Matter! Understanding Differences in the Privacy Mental Models of Smart Home Visitors and Residents. In *Proceedings of the 21st International Conference on Mobile and Ubiquitous Multimedia (MUM)*. (Nov. 2022). DOI: 10.1145/3490632.3490664.

[183] J. Matejka, P. Podhradský, and J. Londák. 2016. Security Manager for Hybrid Broadcast Broadband Architecture Evolution. In *Proceedings of the 58th International Symposium ELMAR*. (Sept. 2016). DOI: 10.1109/ELMAR.2016.7731754.

[184] F. Mathis, K. Vaniea, and M. Khamis. 2022. Prototyping Usable Privacy and Security Systems: Insights from Experts. *International Journal of Human–Computer Interaction*, 38, 5. DOI: 10.1080/10447318.2021.1949134.

[185] D. H. Mcknight, M. Carter, J. B. Thatcher, and P. F. Clay. 2011. Trust in a specific technology: An investigation of its components and measures. *ACM Transactions on Management Information Systems (TMIS)*, 2, 2. DOI: 10.1145/1985347.1985353.

[186] L. Mecke, K. Pfeuffer, S. Prange, and F. Alt. 2018. Open Sesame! User Perception of Physical, Biometric, and Behavioural Authentication Concepts to Open Doors. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia (MUM)*. (Nov. 2018). DOI: 10.1145/3282894.3282923.

[187] W. Meers. 2015. Hello Barbie, Goodbye Privacy? Hacker Raises Security Concerns. (Nov. 30, 2015). Retrieved Nov. 20, 2023 from https://www.huffpost.com/entry/hello-barbie-security-concerns_n_565c4921e4b072e9d1c24d22.

[188] B. Michéle. 2015. Broadcast. In *Smart TV Security: Media Playback and Digital Video Broadcast*. SpringerBriefs in Computer Science. DOI: 10.1007/978-3-319-20994-4.

[189] Micrsoft. 2023. Azure IP Ranges and Service Tags – Public Cloud. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://www.microsoft.com/en-us/download/details.aspx?id=56519.

[190] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma. 2017. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems (ICDCS)*. (June 2017). DOI: 10.1109/ICDCS.2017.283.

[191] M. Mimoso. 2017. Children's Voice Messages Leaked in CloudPets Database Breach. (Feb. 8, 2017). Retrieved Nov. 20, 2023 from https://threatpost.com/childrens-voice-messages-leaked-in-cloudpets-database-breach/123956/.

[192] B. D. Mittelstadt, P. Allo, M. Taddeo, S. Wachter, and L. Floridi. 2016. The Ethics of Algorithms: Mapping the Debate. *Big Data & Society*, 3, 2. DOI: 10.1177/2053951716679679.

[193] H. Mohajeri Moghaddam, G. Acar, B. Burgess, A. Mathur, D. Y. Huang, N. Feamster, E. W. Felten, P. Mittal, and A. Narayanan. 2019. Watching You Watch: The Tracking Ecosystem of Over-the-Top TV Streaming Devices. In *Proceedings of the 26th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Nov. 2019). DOI: 10.1145/3319535.3354198.

[194] C. Morgan. 2022. Do Android proxy settings apply to all apps on the device? Retrieved July 27, 2022 from http://copyprogramming.com/howto/do-android-proxy-settings-apply-to-all-apps-on-the-device.

[195] L. Morgese Zangrandi, T. van Ede, T. Booij, S. Sciancalepore, L. Allodi, and A. Continella. 2022. Stepping out of the MUD: Contextual threat information for IoT devices with manufacturer-provided behaviour profiles. In *Proceedings of the 38th Annual Computer Security Applications Conference (ACSAC)*. (Dec. 2022). DOI: 10.1145/3564625.3564644.

[196] A. Nakajima, T. Watanabe, E. Shioji, M. Akiyama, and M. Woo. 2019. A Pilot Study on Consumer IoT Device Vulnerability Disclosure and Patch Release in Japan and the United States. In *Proceedings of the 14th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*. (July 2019). DOI: 10.1145/3321705.3329849.

[197] Y. Nan, X. Wang, L. Xing, X. Liao, R. Wu, J. Wu, Y. Zhang, and X. Wang. 2023. Are You Spying on Me? Large-Scale Analysis on IoT Data Exposure through Companion Apps. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security)*. (Aug. 2023).

[198] National Cyber Security Centre of The Netherlands (NCSC-NL). 2018. Coordinated Vulnerability Disclosure: the Guideline. (Oct. 2, 2018). Retrieved May 19, 2024 from https://english.ncsc.nl/publications/publications/2019/juni/01/coordinated-vulnerability-disclosure-the-guideline.

[199] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani. 2019. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Communications Surveys & Tutorials*, 21, (July 2019), 3, (July 2019). DOI: 10.1109/COMST.2019.2910750.

[200] M. Neugschwandtner, M. Lindorfer, and C. Platzer. 2013. A View to a Kill: WebView Exploitation. In *Proceedings of the 6th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*. (Aug. 2013). DOI: 10.5555/3241085.3241092.

[201] S. Neupane, F. Tazi, U. Paudel, F. V. Baez, M. Adamjee, L. De Carli, S. Das, and I. Ray. 2022. On the Data Privacy, Security, and Risk Postures of IoT Mobile Companion Apps. In *Proceedings of the 36th IFIP Annual Conference on Data and Applications Security and Privacy (DBSec)*. (July 2022). DOI: 10.1007/978-3-031-10684-2_10.

[202] J. L. Newcomb, S. Chandra, J.-B. Jeannin, C. Schlesinger, and M. Sridharan. 2017. IOTA: a calculus for internet of things automation. In *Proceedings of the ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward)*. (Oct. 2017). DOI: 10.1145/3133850.3133860.

[203] D. T. Nguyen, C. Song, Z. Qian, S. V. Krishnamurthy, E. J. M. Colbert, and P. McDaniel. 2018. IotSan: Fortifying the Safety of IoT Systems. In *Proceedings of the 14th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. (Dec. 2018). DOI: 10.1145/3281411.3281440.

[204] NicolasFNino and S. Arzt. 2023. GitHub FlowDroid Issue #601 – FlowDroid with EPICC. (Apr. 5, 2023). Retrieved Apr. 27, 2023 from https://github.com/secure-software-engineering/FlowDroid/issues/601.

[205] J. Nielsen. 1994. Severity Ratings for Usability Problems. (Nov. 1994). https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/.

[206] NIST. 2011. CVE-2011-3389. (Sept. 6, 2011). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/cve-2011-3389.

[207] NIST. 2015. CVE-2015-4000. (May 26, 2015). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2015-4000.

[208] NIST. 2015. CVE-2015-6409. (Dec. 26, 2015). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2015-6409.

[209] NIST. 2016. CVE-2016-2183. (Aug. 31, 2016). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2016-2183.

[210] NIST. 2017. CVE-2017-8866. (Dec. 11, 2017). Retrieved July 26, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2017-8866.

[211] NIST. 2018. CVE-2018-19417. (Nov. 21, 2018). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2018-19417.

[212] NIST. 2019. CVE-2017-7655. (Mar. 27, 2019). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2017-7655.

[213] NIST. 2019. CVE-2018-12550. (Mar. 27, 2019). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2018-12550.

[214] NIST. 2019. CVE-2018-12551. (Mar. 27, 2019). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2018-12551.

[215] NIST. 2019. CVE-2018-12679. (Apr. 2, 2019). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2018-12679.

[216] NIST. 2019. CVE-2019-16732. (Dec. 13, 2019). Retrieved July 26, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2019-16732.

[217] NIST. 2019. CVE-2019-9749. (Mar. 13, 2019). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2019-9749.

[218] NIST. 2019. CVE-2019-9750. (Mar. 13, 2019). Retrieved Jan. 5, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2019-9750.

[219] NIST. 2022. CVE-2022-30271. (Dec. 13, 2022). Retrieved July 26, 2023 from https://nvd.nist.gov/vuln/detail/CVE-2022-30271.

[220] J. O'Hagan, M. Khamis, M. McGill, and J. R. Williamson. 2022. Exploring Attitudes Towards Increasing User Awareness of Reality From Within Virtual Reality. In *Proceedings of the 2022 ACM International Conference on Interactive Media Experiences (IMX)*. (June 2022). DOI: 10.1145/3505284.3529971.

[221] T. OConnor, D. Jessee, and D. Campos. 2021. Through the Spyglass: Towards IoT Companion App Man-in-the-Middle Attacks. In *Proceedings of the Cyber Security Experimentation and Test Workshop (CSET)*. (Sept. 2021). DOI: 10.1145/3474718.3474729.

[222] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi. 2019. HomeSnitch: Behavior Transparency and Control for Smart Home IoT Devices. In *Proceedings of the 12th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WISEC)*. (May 2019). DOI: 10.1145/3317549.3323409.

[223] OECD. 2021. Understanding the digital security of products. *OECD Digital Economy Papers*, (Feb. 9, 2021). DOI: 10.1787/abea0b69-en.

[224] F. R. Olivera. [n. d.] Maven Repository. Retrieved Apr. 5, 2023 from https://mvnrepository.com/.

[225] Oracle. 2023. Oracle Cloud Infrastructe Documentation – IP Address Ranges. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://docs.oracle.com/en-us/iaas/Content/General/Concepts/addressranges.htm.

[226] Y. Oren and A. D. Keromytis. 2014. From the Aether to the Ethernet - Attacking the Internet using Broadcast Digital Television. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*. (Aug. 2014).

[227] P. Oser, S. Feger, P. W. Woźniak, J. Karolus, D. Spagnuelo, A. Gupta, S. Lüders, A. Schmidt, and F. Kargl. 2020. SAFER: Development and Evaluation of an IoT Device Risk Assessment Framework in a Multinational Organization. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4, 3. DOI: 10.1145/3414173.

[228] A. Palmieri, P. Prem, S. Ranise, U. Morelli, and T. Ahmad. 2019. MQTTSA: A Tool for Automatically Assisting the Secure Deployments of MQTT Brokers. In *Proceedings of the 12th IEEE World Congress on Services (SERVICES)*. (July 2019). DOI: 10.1109/SERVICES.2019.00023.

[229] B. Pan. 2021. GitHub – Dex2jar. v2.1. (Oct. 29, 2021). Retrieved Jan. 15, 2022 from https://github.com/pxb1988/dex2jar.

[230] M. T. Paracha, D. J. Dubois, N. Vallina-Rodriguez, and D. Choffnes. 2021. IoTLS: Understanding TLS Usage in Consumer IoT Devices. In *Proceedings of the 21st Internet Measurement Conference (IMC)*. (Nov. 2021). DOI: 10.1145/3487552.3487830.

[231] E. Pariwono, D. Chiba, M. Akiyama, and T. Mori. 2018. Don't Throw Me Away: Threats Caused by the Abandoned Internet Resources Used by Android Apps. In *Proceedings of the 13th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*. (June 2018). DOI: 10.1145/3196494.3196554.

[232] C. Partridge and M. Allman. 2016. Ethical Considerations in Network Measurement Papers. *Communications of the ACM*, 59, 10. DOI: 10.1145/2896816.

[233] E. Pauley and P. McDaniel. 2023. Understanding the Ethical Frameworks of Internet Measurement Studies. In *Proceedings of the 2nd IEEE International Workshop on Ethics in Computer Security (ETHICS)*. (Feb. 2023). DOI: 10.14722/ethics.2023.239547.

[234] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis. 2020. IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis. In *Proceedings of the 5th IEEE European Symposium on Security & Privacy (EuroS&P)*. (Sept. 2020). DOI: 10.1109/EuroSP48549.2020.00037.

[235] Perflyst. 2021. Smart-TV Blocklist for Pi-hole. Retrieved May 23, 2022 from https://perflyst.github.io/PiHoleBlocklist/SmartTV.txt.

[236] J. Pierce. 2019. Smart Home Security Cameras and Shifting Lines of Creepiness: A Design-Led Inquiry. In *Proceedings of the 2019*. (May 2019). DOI: 10.1145/3290605.3300275.

[237] S. Pletinckx, K. Borgolte, and T. Fiebig. 2021. Out of Sight, Out of Mind: Detecting Orphaned Web Pages at Internet-Scale. In *Proceedings of the 28th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Nov. 2021). DOI: 10.1145/3460120.3485367.

[238] A. Pradeep, A. Feal, J. Gamba, A. Rao, M. Lindorfer, N. Vallina-Rodriguez, and D. Choffnes. 2023. Not Your Average App: A Large-scale Privacy Analysis of Android Browsers. In *Proceedings of the 23rd Privacy Enhancing Technologies Symposium (PETS)*. (July 2023). DOI: 10.56553/popets-2023-0003.

[239] A. Pradeep, T. M. Paracha, P. Bhomwick, A. Davanian, A. Razaghpanah, T. Chung, M. Lindorfer, N. Vallina-Rodriguez, D. Levin, and D. Choffnes. 2022. A Comparative Analysis of Certificate Pinning in Android & iOS. In *Proceedings of the 22nd Internet Measurement Conference (IMC)*. (Oct. 2022). DOI: 10.1145/3517745.3561439.

[240] S. Prange, C. George, and F. Alt. 2021. Design Considerations for Usable Authentication in Smart Homes. In *Proceedings of the 2021 Mensch und Computer (MuC)*. (Sept. 2021). DOI: 10.1145/3473856.3473878.

[241] S. Prange, E. von Zezschwitz, and F. Alt. 2019. Vision: Exploring Challenges and Opportunities for Usable Authentication in the Smart Home. In *Proceedings of the 4th IEEE European Symposium on Security & Privacy Workshops (EuroS&PW)*. (June 2019). DOI: 10.1109/EuroSPW.2019.00024.

[242] S. Preibusch. 2013. Guide to Measuring Privacy Concern: Review of Survey and Observational Instruments. *International Journal of Human-Computer Studies*, 71, 12. DOI: 10.1016/j.ijhcs.2013.09.002.

[243] M. Rapoport, P. Suter, E. Wittern, O. Lhotak, and J. Dolby. 2017. Who You Gonna Call? Analyzing Web Requests in Android Applications. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR)*. (May 2017). DOI: 10.1109/MSR.2017.11.

[244] Raspberry Pi Foundation. 2022. Raspberry Pi. Retrieved Nov. 29, 2022 from https://www.raspberrypi.org/.

[245] O. A. V. Ravnås. [n. d.] Frida. Retrieved Aug. 24, 2022 from https://frida.re/.

[246] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill. 2018. Apps, Trackers, Privacy, and Regulators: A Global Study of the Mobile Tracking

Ecosystem. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2018). DOI: 10.14722/ndss.2018.23009.

[247] N. Redini, A. Continella, D. Das, G. De Pasquale, N. Spahn, A. Machiry, A. Bianchi, C. Kruegel, and G. Vigna. 2019. DIANE: Identifying Fuzzing Triggers in Apps to Generate Under-constrained Inputs for IoT Devices. In *Proceedings of the 40th IEEE Symposium on Security & Privacy (S&P)*. (May 2019). DOI: 10.1109/SP40001.2021.00066.

[248] E. M. Redmiles, Z. Zhu, S. Kross, D. Kuchhal, T. Dumitras, and M. L. Mazurek. 2018. Asking for a Friend: Evaluating Response Biases in Security User Studies. In *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Oct. 2018). DOI: 10.1145/3243734.3243740.

[249] RedOrbit GmbH. 2022. RedOrbit HbbTV Emulator. Retrieved Nov. 29, 2022 from https://chrome.google.com/webstore/detail/redorbit-hbbtv-emulator/mmgfafehampkahlmoahbjcjcmgmkppab?hl=en.

[250] D. Reidsma, J. van der Ham, and A. Continella. 2023. Operationalizing Cybersecurity Research Ethics Review: From Principles and Guidelines to Practice. In *Proceedings of the 2nd IEEE International Workshop on Ethics in Computer Security (ETHICS)*. (Feb. 2023). DOI: 10.14722/ethics.2023.237352.

[251] D. Reidsma, J. van der Ham, and A. Continella. 2023. Operationalizing Cybersecurity Research Ethics Review: From Principles and Guidelines to Practice. In *Proceedings of the 2nd IEEE International Workshop on Ethics in Computer Security (ETHICS)*. (Feb. 2023). DOI: 10.14722/ethics.2023.237352.

[252] N. Reitinger, W. He, C. Bruno, S. Landau, C. A. Gunter, M. Khanafer, R. Mangar, and D. L. Anthony. 2025. Help Me Help You: Privacy Considerations for Third Party IoT Device Repair. In *Proceedings of the 25th Privacy Enhancing Technologies Symposium (PETS)*. (July 2025). DOI: 10.56553/popets-2025-0128.

[253] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi. 2019. Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proceedings of the 19th Internet Measurement Conference (IMC)*. (Oct. 2019). DOI: 10.1145/3355369.3355577.

[254] J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez. 2018. Bug Fixes, Improvements, ... and Privacy Leaks - A Longitudinal Study of PII Leaks Across Android App Versions. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2018). DOI: 10.14722/ndss.2018.23143.

[255] ReVuln. 2013. The TV is watching you: Samsung 0-day. Last accessed: 2021-01-18. https://vimeo.com/55174958.

[256] L. P. Rondon, L. Babun, K. Akkaya, and A. S. Uluagac. 2019. HDMI-Walk: Attacking HDMI Distribution Networks via Consumer Electronic Control Protocol. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*. (Dec. 2019). DOI: 10.1145/3359789.3359841.

[257] L. P. Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac. 2022. Survey on Enterprise Internet-of-Things systems (E-IoT): A security perspective. *Ad Hoc Networks*, 125. DOI: 10.1016/j.adhoc.2021.102728.

[258] 2022. RTL 102.5 TV (Page Version ID: 128120211). Retrieved July 28, 2022 from https://it.wikipedia.org/w/index.php?title=RTL_102.5_TV&oldid=128120211.

[259] J. Rüth, T. Zimmermann, K. Wolsing, and O. Hohlfeld. 2018. Digging into Browser-Based Crypto Mining. In *Proceedings of the 18th Internet Measurement Conference (IMC)*. (Nov. 2018). DOI: 10.1145/3278532.3278539.

[260] S. Sabetan. 2023. The Uninvited Guest: IDORs, Garage Doors, and Stolen Secrets. (Apr. 4, 2023). Retrieved Nov. 20, 2023 from https://medium.com/@samsabetan/the-uninvited-guest-idors-garage-doors-and-stolen-secrets-e4b49e02dadc.

[261] S. J. Saidi, S. Matic, O. Gasser, G. Smaragdakis, and A. Feldmann. 2022. Deep Dive into the IoT Backend Ecosystem. In *Proceedings of the 22nd Internet Measurement Conference (IMC)*. (Oct. 2022). DOI: 10.1145/3517745.3561431.

[262] P. Saint-Andre. 2011. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120. (Mar. 2011). https://www.rfc-editor.org/info/rfc6120.

[263] P. Saint-Andre. 2011. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120. (Mar. 2011). https://www.rfc-editor.org/info/rfc6120.

[264] P. Saint-Andre. 2011. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. RFC 6121. (Mar. 2011). https://www.rfc-editor.org/info/rfc6121.

[265] P. Saint-Andre. 2012. In-Band Registration. XEP 0077. Version 2.4. XMPP Standards Foundation. https://xmpp.org/extensions/xep-0077.html.

[266] P. Saint-Andre and D. Cridland. 2012. Roster Versioning. XEP 0237. Version 1.3. XMPP Standards Foundation. https://xmpp.org/extensions/xep-0237.html.

[267] Salesforce. 2023. Salesforce Core Services – IP Addresses and Domains to Allow. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://help.salesforce.com/s/articleView?id=000384438&type=1.

[268] Samsung. 2021. Cotopaxi. v1.7.0. (Aug. 5, 2021). Retrieved Aug. 22, 2022 from https://github.com/Samsung/cotopaxi.

[269] D. Schmidt, C. Tagliaro, K. Borgolte, and M. Lindorfer. 2023. IoTFlow: Inferring IoT Device Behavior at Scale through Static Mobile Companion App Analysis. In *Proceedings of the 30th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Nov. 2023). DOI: 10.1145/3576915.3623211.

[270] T. Seals. 2022. Critical MQTT-Related Bugs Open Industrial Networks to RCE Via Moxa. (Feb. 11, 2022). Retrieved Nov. 20, 2023 from https://threatpost.com/critical-mqtt-bugs-industrial-rce-moxa/178399/.

[271] L. SeungJin and K. Seungjoo. 2013. Smart TV Security - #1984 in 21st Century. https://www.slideshare.net/skim71/smart-tv-security-1984-in-21st-century.

[272] Shadowserver Foundation. 2020. Accessible CoAP Report – Exposed Constrained Application Protocol Services on the Internet. (June 24, 2020). Retrieved Nov. 15, 2023 from https://www.shadowserver.org/news/accessible-coap-report-scanning-for-exposed-constrained-application-protocol-services/.

[273] A. Sharif. 2024. Consumer Spending on Smart Home Products and Services Worldwide from 2015 to 2025. (Aug. 2024). Retrieved Aug. 22, 2024 from https://www.statista.com/statistics/693303/smart-home-consumer-spending-worldwide/.

[274] E. Shiffman. 2018. HbbTV: How Addressable TV is Implemented in the EU. (July 19, 2018). Retrieved Jan. 31, 2021 from https://www.spotx.tv/resources/blog/product-pulse/hbbtv-how-addressable-tv-is-implemented-in-the-eu/.

[275] Shodan. 2023. Honeyscore - Honeypot Or Not? Retrieved Oct. 21, 2023 from https://honeyscore.shodan.io/.

[276] Shodan. [n. d.] Shodan – Search Engine for the Internet of Everything. Retrieved Oct. 14, 2023 from https://www.shodan.io.

[277] P. Sivakumaran and J. Blasco. 2019. A Study of the Feasibility of Co-located App Attacks against BLE and a Large-Scale Analysis of the Current Application-Layer Security Landscape. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*. (Aug. 2019). DOI: 10.5555/3361338.3361340.

[278] P. Sivakumaran, C. Zuo, Z. Lin, and J. Blasco. 2023. Uncovering Vulnerabilities of Bluetooth Low Energy IoT from Companion Mobile Apps with Ble-Guuide. In *Proceedings of the 18th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*. (July 2023). DOI: 10.1145/3579856.3595806.

[279] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. 2019. Classifying IoT Devices in Smart Environments Using Network Traffic

Characteristics. *IEEE Transactions on Mobile Computing*, 18, 8. DOI: 10.1109/TMC.2018.2866249.

[280] V. Sivaraman, D. Chan, D. Earl, and R. Boreli. 2016. Smart-Phones Attacking Smart-Homes. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WISEC)*. (July 2016). DOI: 10.1145/2939918.2939925.

[281] N. Sombatruang, T. Caulfield, I. Becker, A. Fujita, T. Kasama, K. Nakao, and D. Inoue. 2023. Internet Service Providers' and Individuals' Attitudes, Barriers, and Incentives to Secure IoT. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security)*. (Aug. 2023).

[282] SoSci Survey GmbH. 2022. SoSci Survey. Retrieved Nov. 29, 2022 from https://www.soscisurvey.de/.

[283] D. Spencer. 2009. *Card Sorting: Designing Usable Categories*. English. (1st ed.). Rosenfeld Media, (Apr. 2009). ISBN: 978-1933820026.

[284] D. Springall, Z. Durumeric, and J. A. Halderman. 2016. FTP: The Forgotten Cloud. In *Proceedings of the 46th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. (June 2016). DOI: 10.1109/DSN.2016.52.

[285] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis. 2021. Open for hire: attack trends and misconfiguration pitfalls of IoT devices. In *Proceedings of the 21st Internet Measurement Conference (IMC)*. (Nov. 2021). DOI: 10.1145/3487552.3487833.

[286] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis. 2021. RIoTPot: A Modular Hybrid-interaction IoT/OT Honeypot. In *Proceedings of the 26th European Symposium on Research in Computer Security (ESORICS)*. (Oct. 2021). DOI: 10.1007/978-3-030-88428-4.

[287] Statista. 2022. Arte: Audience Market Share Germany 2021. Retrieved July 28, 2022 from https://www.statista.com/statistics/416431/kabel-eins-audience-market-share-germany/.

[288] Statista. 2022. Austria: TV channels by audience share october 2022. Retrieved Nov. 28, 2022 from https://de.statista.com/statistik/daten/studie/303727/umfrage/marktanteile-der-fernsehsender-in-oesterreich-monatlich/.

[289] Statista. 2022. Finland: TV channels by audience share 2021. Retrieved Nov. 28, 2022 from https://www.statista.com/statistics/633922/tv-channels-in-finland-by-audience-share/.

[290] Statista. 2022. Sport1: audience share 2021. Retrieved Nov. 28, 2022 from https://de.statista.com/statistik/daten/studie/285796/umfrage/marktanteil-von-sport-1/.

[291] Statista. 2022. ZDF: Market Share in Germany 2021. Retrieved July 28, 2022 from https://www.statista.com/statistics/410906/zdf-market-share-germany/.

[292] B. Stock, G. Pellegrino, F. Li, M. Backes, and C. Rossow. 2018. Didn't You Hear Me? - Towards More Successful Web Vulnerability Notifications. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2018). DOI: 10.14722/ndss.2018.23171.

[293] B. Stock, G. Pellegrino, C. Rossow, M. Johns, and M. Backes. 2016. Hey, You Have a Problem: On the Feasibility of Large-Scale Web Vulnerability Notification. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security)*. (Aug. 2016).

[294] J. Stoll. 2025. Average daily time spent watching TV in the United States from 2019 to 2023. (June 23, 2025). Retrieved June 23, 2025 from https://www.statista.com/statistics/186833/average-television-use-per-person-in-the-us-since-2002/.

[295] J. Stoll. 2025. Daily TV viewing time in European countries. (June 23, 2025). Retrieved June 23, 2025 from https://www.statista.com/statistics/422719/tv-daily-viewing-time-europe/.

[296] J. Stoll. 2025. Number of TV households worldwide from 2010 to 2026. (June 23, 2025). Retrieved Sept. 25, 2022 from https://www.statista.com/statistics/268695/number-of-tv-households-worldwide/.

[297] F. Streibelt, M. Lindorfer, S. Gürses, C. H. Gañán, and T. Fiebig. 2023. Back-to-the-Future Whois: An IP Address Attribution Service for Working with Historic Datasets. In *Proceedings of*

the 24th Passive and Active Measurement Conference (PAM). (Mar. 2023). DOI: 10.1007/978-3-031-28486-1_10.

[298] L. Sujay Vailshery. 2023. Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030. (July 27, 2023). Retrieved Oct. 13, 2023 from https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[299] C. Tagliaro, F. Hahn, R. Sepe, A. Aceti, and M. Lindorfer. 2023. I Still Know What You Watched Last Sunday: Security and Privacy of the HbbTV Protocol in the European Smart TV Landscape. In *Proceedings of the 30th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2023). DOI: 10.14722/ndss.2023.24102.

[300] C. Tagliaro, F. Hahn, R. Sepe, A. Aceti, and M. Lindorfer. 2023. Investigating HbbTV Privacy Invasiveness Across European Countries. In *Proceedings of the 12th Workshop on Learning from Authoritative Security Experiment Results (LASER)*. (Feb. 2023). DOI: 10.14722/laser-ndss.2023.24102.

[301] C. Tagliaro, M. Komsic, A. Continella, K. Borgolte, and M. Lindorfer. 2024. Large-Scale Security Analysis of Real-World Backend Deployments Speaking IoT-Focused Protocols. In *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. (Sept. 2024). DOI: 10.1145/3678890.3678899.

[302] O. Traynor. 2025. IoT Cybersecurity: Connecting the Dots. Retrieved Aug. 25, 2025 from https://cybelangel.com/iot_cybersecurity/.

[303] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky. 2020. Packet-Level Signatures for Smart Home Devices. In *Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2020). DOI: 10.14722/ndss.2020.24097.

[304] J. W. Tukey. 1949. Comparing individual means in the analysis of variance. *Biometrics*, 5, 2. DOI: 10.2307/3001913.

[305] R. Vallée-Rai, E. Gagnon, L. Hendren, P. Lam, P. Pominville, and V. Sundaresan. 2000. Optimizing Java Bytecode Using the Soot Framework: Is It Feasible? In *Proceedings of the 9th Compiler Construction*. (Jan. 2000). DOI: 10.1007/3-540-46423-9_2.

[306] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, K. Papagiannaki, H. Haddadi, and J. Crowcroft. 2012. Breaking for Commercials: Characterizing Mobile Advertising. In *Proceedings of the 12th Internet Measurement Conference (IMC)*. (Nov. 2012). DOI: 10.1145/2398776.2398812.

[307] J. van der Ham and R. van Rijswijk-Deij. 2017. Ethics and Internet Measurements. *Journal of Cyber Security and Mobility*, 5, (Oct. 2017), 4, (Oct. 2017). http://dx.doi.org/10.13052/jcsm2245-1439.543.

[308] J. van der Ham, A. Continella, P. de Willigen, and D. Reidsma. 2023. University of Twente Policy for Coordinated Vulnerability Disclosure in Research. (Mar. 22, 2023). Retrieved May 19, 2024 from https://www.utwente.nl/en/service-portal/research-support/procedures-facilities/coordinated-vulnerability-disclosure-policy-for-research.

[309] M. van der Horst. 2023. Global Vulnerability Vigilance: Timely Disaster Notification using Internet-Scale Coordinated Vulnerability Disclosure. (July 11, 2023). Retrieved May 19, 2024 from https://scripties.uba.uva.nl/search?id=record_54279.

[310] J. Varmarken, H. Le, A. Shuba, A. Markopoulou, and Z. Shafiq. 2020. The TV is Smart and Full of Trackers: Measuring Smart TV Advertising and Tracking. In *Proceedings of the 20th Privacy Enhancing Technologies Symposium (PETS)*. (July 2020). DOI: 10.2478/popets-2020-0021.

[311] S. Viswanathan, B. Omidvar-Tehrani, and J.-M. Renders. 2022. What Is Your Current Mindset? Categories for a Satisficing Exploration of Mobile Point-of-Interest Recommendations. In *Proceedings of the 2022*. (Apr. 2022). DOI: 10.1145/3491102.3501912.

[312] T. Walshe and A. C. Simpson. 2022. Coordinated Vulnerability Disclosure programme effectiveness: Issues and recommendations. *Computers & Security*, 123, (Dec. 2022). DOI: 10.1016/j.cose.2022.102936.

[313] J. Walton, J. Steven, J. Manico, K. Wall, R. Iramar, and contributors. [n. d.] Certificate and Public Key Pinning. Retrieved Oct. 7, 2022 from https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning.

[314] Q. Wang, W. Ul Hassan, A. Bates, and C. A. Gunter. 2018. Fear and Logging in the Internet of Things. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2018). DOI: 10.14722/ndss.2018.23282.

[315] X. Wang, Y. Sun, S. Nanda, and X. Wang. 2019. Looking from the Mirror: Evaluating IoT Device Security through Mobile Companion Apps. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*. (Aug. 2019). DOI: 10.5555/3361338.3361418.

[316] Wassname. 2020. Samsung Smart-TV Blocklist Adlist (for PiHole). Retrieved May 23, 2022 from https://gist.github.com/wassname/b594c63222f9e4c83ea23c818440901b.

[317] H. Wen, Q. Zhao, Q. A. Chen, and Z. Lin. 2020. Automated Cross-Platform Reverse Engineering of CAN Bus Commands from Mobile Apps. In *Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*. (Feb. 2020). DOI: 10.14722/ndss.2020.24231.

[318] Wetter, Dirk. 2023. testssl. v3.2rc2. Retrieved Jan. 23, 2023 from https://github.com/drwetter/testssl.sh.

[319] S. Whited and J. Schäfer. 2018. XMPP Compliance Suites. XEP 0387. Version 1.0.0. XMPP Standards Foundation. https://xmpp.org/extensions/xep-0387.html.

[320] M. Williams, J. R. C. Nurse, and S. Creese. 2017. "Privacy Is the Boring Bit": User Perceptions and Behaviour in the Internet-of-Things. In *Proceedings of the 15th International Conference on Privacy, Security and Trust (PST)*. (Aug. 2017). DOI: 10.1109/PST.2017.00029.

[321] Wireshark Foundation. 2022. TShark. Retrieved May 30, 2022 from https://www.wireshark.org/docs/man-pages/tshark.html.

[322] Wireshark Foundation. 2022. Wireshark. Retrieved May 30, 2022 from https://www.wireshark.org/.

[323] M. S. Wogalter, D. M. DeJoy, and K. R. Laughery. 1999. Organizing Theoretical Framework: A Consolidated Communication-Human Information Processing (C-HIP) Model. In *Warnings and Risk Communication*.

[324] D. Wood, N. Apthorpe, and N. Feamster. 2017. Cleartext Data Transmissions in Consumer IoT Medical Devices. In *Proceedings of the Workshop on Internet of Things Security and Privacy (IoT-S&P)*. (Nov. 2017). DOI: 10.1145/3139937.3139939.

[325] H. Xu, M. Yu, Y. Wang, Y. Liu, Q. Hou, Z. Ma, H. Duan, J. Zhuge, and B. Liu. 2022. Trampoline Over the Air: Breaking in IoT Devices Through MQTT Brokers. In *Proceedings of the 7th IEEE European Symposium on Security & Privacy (EuroS&P)*. (June 2022). DOI: 10.1109/EuroSP53844.2022.00019.

[326] R. Yaben, N. Lundsgaard, J. August, and E. Vasilomanolakis. 2024. Towards Identifying Neglected, Obsolete, and Abandoned IoT and OT Devices. In *Proceedings of the 8rd Network Traffic Measurement and Analysis Conference (TMA)*. (May 2024). DOI: 10.23919/TMA62044.2024.10558996.

[327] Yandex. 2023. Public IP address ranges for Yandex Cloud. (Jan. 23, 2023). Retrieved Jan. 23, 2023 from https://cloud.yandex.com/en/docs/vpc/concepts/ips.

[328] yokotayokota, S. Arzt, and J. Samhi. 2021. GitHub FlowDroid Issue #386 – Usage of IccTA in Flowdroid. (Sept. 13, 2021). Retrieved Apr. 27, 2023 from https://github.com/secure-software-engineering/FlowDroid/issues/386.

[329] S. Zamfir, T. C. Balan, I. Iliescu, and F. Sandu. 2016. A security analysis on standard IoT protocols. In *Proceedings of the 3rd International Conference on Applied and Theoretical Electricity (ICATE)*. (Oct. 2016). DOI: 10.1109/ICATE.2016.7754665.

[330] Q. Zhao, C. Zuo, J. Blasco, and Z. Lin. 2022. PeriScope: Comprehensive Vulnerability Analysis of Mobile App-defined Bluetooth Peripherals. In *Proceedings of the 17th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*. (May 2022). DOI: 10.1145/3488932.3517410.

[331] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, and Y. Zhang. 2019. Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*. (Aug. 2019). DOI: 10.5555/3361338.3361417.

[332] C. Zuo, Z. Lin, and Y. Zhang. 2019. Why Does Your Data Leak? Uncovering the Data Leakage in Cloud from Mobile Apps. In *Proceedings of the 40th IEEE Symposium on Security & Privacy (S&P)*. (May 2019). DOI: 10.1109/SP.2019.00009.

[333] C. Zuo, H. Wen, Z. Lin, and Y. Zhang. 2019. Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps. In *Proceedings of the 26th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. (Nov. 2019). DOI: 10.1145/3319535.3354240.