Security and Privacy for Connected Devices

A thesis presented in fulfillment of the requirements for the Habilitation to

Direct Research (HDR).

Dr. Daniele Antonioli

Reviewers: Prof. I. Verbauwhede (KUL), Prof. C. Castelluccia (INRIA), Prof. M. Cunche (INRIA)

Jury: Prof. I. Verbauwhede (KUL), Prof. C. Castelluccia (INRIA), Prof. M. Cunche (INRIA), Prof. H. Debar (Télécom SudParis), Prof. M. Payer (EPFL), Prof. K. Rasmussen (Oxford University), Dr. C. Maurice (INRIA)

Institutions: Institut Polytechnique Paris (IPP) École Polytechnique (EP), and EURECOM

May 12, 2025

Contents

List of Figures

List of Tables

Listings

Abstract

The thesis tackles the challenge of providing secure and private connected devices. Our definition of connected device includes information technology, operational technology, and Internet of Things devices and their related networks and communication technologies. Since we connect billions of devices and use them for pervasive critical services, like communication, positioning, and transportation, it is essential to protect connected devices from security, privacy, and even safety breaches.

Security and privacy of connected devices is a massive research area. We focus on four urgent research questions (RQ1–RQ4) related to the security and privacy of standard and proprietary technologies used by connected devices, risk assessment with threat modeling, and unauthorized device and user tracking. We answer the presented research questions with novel contributions from four research papers (BLUFFS, E-Spoofer, ADF, and FP-tracer) and discuss five more research papers related to the thesis (E-Trojans, CTRAPS, BreakMi, AutoBtSec, and BLURtooth). We conclude the thesis discussing future directions in connected system security and privacy, including the open-source hardware revolution.

Acknowledgements

I want to thank my PhD students, Marco, Soumaya, Tommaso, and Farzam, who are the lifeblood of my research group. Without them, I would not have been able to submit this HDR thesis. I also thank the many master's and PhD students I have advised and co-advised so far who enriched my perspectives.

I thank Nils, Kasper, and Mathias, who advised me during the calm and windy days. I also thank the ORSHIN team, which is reshaping security and privacy for open-source hardware and software and the ENCOPIA team who has provided practical and elegant solutions for end-to-end security and privacy assessment. Finally, I thank all the collaborators I met and will meet along the way.

Thanks to the HDR reviewers and jury members for their feedback and the HDR contact points at EP and IPP. Thanks to EURECOM, which supports my candidature and provides a welcoming and stimulating research and teaching environment.

Thanks to my family, Aurora, Sette, and friends for their support. We roll with it!

Chapter 1: Introduction

1.1 Scope

The thesis focuses on *security* and *privacy* for *connected devices*. Connected devices are pervasive, with an adoption rate exponentially increasing over the years. For example, we are connecting approximately 20 billion IoT devices and their number will double by 2033 [1]. These connections enable essential services for billions of people, such as sensing, monitoring, remote control, virtual/augmented reality, communication, tracking, positioning, transportation, data transfer, and health. It is paramount to protect these services to avoid security, privacy, and safety issues [2, 3, 4].

A device is a collection of *hardware* and *software* that work together to provide a service. The hardware includes the device processor, memory, and input/output chips. The software covers the bootloader, kernel, firmware, operating system, libraries, and applications. Typically, the software is updatable even remotely, while the hardware cannot be updated unless a device is recalled. The hardware design and implementation are usually proprietary, while the software counterparts can be open source.

Connected devices communicate using a plethora of *protocols*. There are *standard* protocols such as Bluetooth, Wi-Fi, and TLS. These protocols are used daily by billions of devices, and their main benefit is interoperability. But, there are also *proprietary* protocols developed by vendors for ad-hoc services like communication between an IoT device and a smartphone running a companion app. Proprietary protocols are problematic for researchers as their design and implementation are unavailable, and typically reverse-engineering is required to assess them. A protocol design is provided by a *specification*, which is usually in informal (in prose) but can also be formal (mathematical model). A protocol is implemented by following the specification and optionally a reference implementation. Some protocol aspects are implemented in software and others in hardware. A protocol implementation might span multiple subsystems and chips.

Connected devices must provide *security* and *privacy* guarantees. Security is typically defined using the CIA triad (confidentiality, integrity, and availability) and authenticity. These four properties guarantee that unauthorized parties cannot access, modify, or prevent the communication. Privacy is usually assured by confidentiality, authenticity, data protection, anonymity, and unlinkability.

Several national, European, and international *regulations* exist on the security and privacy of connected devices. *Common Criteria (CC)* [5] is the most popular software and

hardware security certification standard. CC provides functional and assurance requirements that product vendors can implement and certify in dedicated CC testing labs. The evaluation involves compliance with one or more protection profiles involving one or more security targets and is standardized in the Common Methodology for Information Technology Security Evaluation (CEM).

FIPS 140-3 [6] complements CC with cryptographic design and implementation requirements involving software and hardware. It is used to approve and standardize the usage of cryptographic mechanisms, like AES for encryption, and protocols, like TLS for data protection. As for CC, a vendor can test and qualify a device as FIPS compliant according to cryptographic requirements.

The *GDPR (General Data Protection Regulation)* [7] is a European law to safeguard people's data and protect their privacy. The regulation, among others, focus on lawful processing of personal data and guarantees that the purpose of the data collection is well defined and the related data collection and storage is limited accordingly. Moreover, it safeguards the rights to access, rectify, erase, restrict, and port personal data.

However, security and privacy regulations alone are insufficient to protect connected devices. They should be combined with effective methods and frameworks for assessing whether a regulation holds in practice. Moreover, we need future-proof techniques to uncover and fix security and privacy issues that are not covered or even known by a regulation.

Achieving both security *and* privacy (S&P) is challenging because the two requirements are *intertwined* and might even *compete*. It is known that confidentiality, i.e., data protection from unauthorized access, improves security and privacy. However, authenticity, i.e., assuring that a device is trustworthy, favors security but reduces privacy. For example, a device with a unique and cryptographically signed identity (e.g., a trusted certificate) is easy to authenticate (security gain), but it is also easy to track (privacy loss).

1.2 Challenges and Research Questions

Interconnecting our societies enables better services but also increases security and privacy risks. Today, we can control an industrial nuclear power plant using JavaScript and a tablet. We can monitor a fleet of millions of (electric) vehicles by querying a cloud server with our smartphone. We can automate the management of our house with sensors and actuators and control them remotely. We can chat with our smartphones using satellite communication. Breaking any of these communication links can result in catastrophic consequences.

Security and privacy researchers, often in separate and specialized communities, struggle to keep up with the rapid evolution of connected devices. For example, every smart device

has more and more software and hardware features to improve the user experience, but with potential security and privacy side effects. A recent example is Windows Recall, an AI assistant enabled by new software and hardware (CoPilot+ and NPU). After its release, Windows had to immediately disable and retire it because of its associated privacy risks [8].

The thesis's challenge is *providing security and privacy for connected devices*. Our definition of connected devices is broad and includes information technology (IT), Internet of Things (IoT), and operational technology (OT) devices and related networks and communication technologies. Connected devices use cases of interest include: mobile communication, health and object tracking, autonomous driving, smart mobility, critical infrastructures, and home automation.

Security and privacy of connected systems represent a massive research area. The thesis narrows the scope to *four urgent research questions (RQ)* that, if answered properly, would dramatically improve the S&P of real-world connected systems:

RQ1: Do standard and pervasive communication protocols, like Bluetooth, guarantee security and privacy?

RQ2: What is the security and privacy of pervasive proprietary connected device ecosystems, like Xiaomi e-scooters or Fitbit fitness trackers?

RQ3: Can we assess the S&P risks (threat model) of real-world connected systems, like digital cryptowallets?

RQ4: Is the connected society protected from user and device tracking, including web tracking based on browser fingerprinting?

1.3 Contributions

The thesis answers the four research questions presented in Section 1.2 with novel findings from *four research papers*: 1) **BLUFFS [BLU]** [9], 2) **E-Spoofer [ESP]** [10], 3) **Attack-Defense Framework [ADF]** [11], and 4) **FP-tracer [FPT]** [12]. Moreover, it discusses *real-world insights* and *future directions* in connected systems security and privacy, including the open-source hardware revolution.

Next, we motivate why we selected the four papers. The papers have been published in prestigious system security and privacy *journals* (i.e., PETS and ACM TECS) and *conferences* (i.e., ACM WiSec, ACM CCS). Daniele is the first or leading author of the papers. They are ordered chronologically from the least recent (2023–2024). They cover broad and

orthogonal security and privacy areas, including wireless, embedded, mobile, web, and cyber-physical systems, reverse engineering, vulnerability, exploitation, prevention, and threat modeling. They combine novel theoretical approaches with impactful and large-scale empirical assessments.

Moreover, the presented papers involve the *three PhD students* advised by Dr. Daniele Antonioli so far: i) *Dr. Marco Casagrande* who graduated in Dec 2024 (now a postdoc at KTH), ii) *Soumaya Boussaha* who is a third-year PhD student, iii) *Tommaso Sacchetti* who is a second-year PhD student. The works cover the *four research projects* where Daniele is or was lead, PI, or CoPi: ORSHIN, ENCOPIA, AFRL, and NF-HiSec.

The following five papers are also related to this dissertation but cannot be discussed in detail due to space constraints. We summarize them next and provide references for the interested reader.

- **E-Trojans [ETR]** [13] under minor revision at WiSec'25 and submitted to Black Hat USA 25. This is a follow-up to E-Spoofer's work. We focus on the security and privacy of the e-scooter internal architecture. We uncover critical flaws and present the first battery ransomware attack for electric scooters, which generalizes to other battery-powered embedded systems.

- **CTRAPS [CTR]** [14] to be presented at IEEE Euro S&P'25 and is under submission for DEF CON 2025. In this paper, we study the security and privacy of CTAP, a core protocol and component of the FIDO2 authentication standard. We uncover critical vulnerabilities and attacks, called API confusion and Client Impersonation, and provide an open-source tool to reproduce the attacks and related fixes.

- **BreakMi [BMI]** [15] presented at CHES'22 and Hardweario'23. In this paper, we analyze the proprietary communication protocols of FitBit (Google) and Xiaomi, the market's two most popular fitness tracker vendors. We uncover severe vulnerabilities and attacks that allow the impersonation of a trusted fitness tracker to a mobile app and vice-versa, as well as the mounting of a MitM attack. We release an open-source toolkit to reproduce the attacks and discuss effective countermeasures.

- **AutoBtSec [BTC]** [16] presented at IEEE WOOT'22. This paper presents the first evaluation of protocol-level Bluetooth security threats in the automotive industry. We focus on the car infotainment system and show that the vast majority of vehicles are still vulnerable to the KNOB+BIAS impersonation attack chain. An extension of this work was presented in the Bluetoolkit presentation at DEF CON 24 and is under submission at WOOT'25.

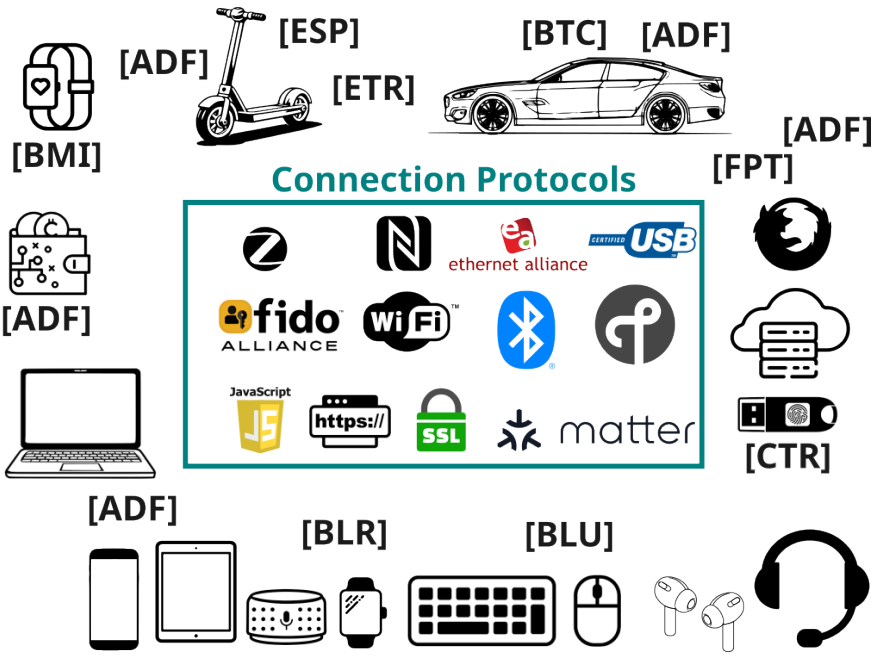

Figure 1.1: Our nine contributions (in square brackets) span the security and privacy of heterogeneous connected systems of devices and humans.

- **BLURtooth [BLR]** [17] presented at ACM AsiaCCS'22. This paper shows for the first time that Bluetooth Classic (BC) can be exploited from Bluetooth Low Energy (BLE) and vice-versa. The root cause of the exploitation is cross-transport key derivation (CTKD), a feature in the Bluetooth standard that was introduced to improve usability.

Figure 1.1 shows the devices and related technologies impacted by our nine research papers. The papers touch billions of heterogeneous devices from IT, IoT, and OT domains, including cars, e-scooters, fitness trackers, cloud servers, IoT devices, mobile devices, and so on. They cover wireless and wired technologies and protocols, which might be standard or proprietary, including Bluetooth Classic, Bluetooth Low Energy, Wi-FI, NFC, FIDO2, and HTTP. For a complete list of Daniele's publications, refer to https://francozappa.github.io/publication.

1.4 Chapter Organization

The following four chapters provide excerpts from their related research paper. Each chapter starts with references to the paper, the paper abstract, an introduction about the topic, including state-of-the-art and motivation, and presents the design of a solution to tackle the research gap explained in the motivation. Then, it discusses the implementation and evaluation of such a solution. The Conclusion chapter ends the thesis and discusses real-world insights and future directions in security and privacy for connected systems, including the open-source hardware revolution.

Chapter 2: BLUFFS

2.1 Reference

This chapter presents a conference paper from 2023 titled: *BLUFFS: Bluetooth Forward and Future Secrecy Attacks and Defenses* [9]. The paper was published in the *ACM Conference on Computer and Communications Security (CCS)*. For more information, have a look at the paper, slides, code, CVE-2023-24023, 37C3 slides. 37C3 talk, and THCON24 talk.

2.2 Abstract

Bluetooth is a pervasive technology for wireless communication. Billions of devices use it in sensitive applications and to exchange private data. The security of Bluetooth depends on the Bluetooth standard and its two security mechanisms: pairing and session establishment. No prior work, including the standard itself, analyzed the *future and forward secrecy* guarantees of these mechanisms, e.g., if Bluetooth pairing and session establishment defend past and future sessions when the adversary compromises the current. To address this gap, we present *six* novel attacks, defined as the *BLUFFS attacks*, breaking Bluetooth sessions' forward and future secrecy. Our attacks enable device impersonation and machine-in-the-middle *across* sessions by only compromising *one* session key. The attacks exploit *two* novel vulnerabilities that we uncover in the Bluetooth standard related to unilateral and repeatable session key derivation. As the attacks affect Bluetooth at the architectural level, they are effective regardless of the victim's hardware and software details (e.g., chip, stack, version, and security mode).

We also release `BLUFFS`, a low-cost toolkit to perform and automatically check the effectiveness of our attacks. The toolkit employs *seven* original patches to manipulate and monitor Bluetooth session key derivation by dynamically patching a closed-source Bluetooth firmware that we reverse-engineered. We show that our attacks have a *critical* and *large-scale* impact on the Bluetooth ecosystem by evaluating them on *seventeen* diverse Bluetooth chips (*eighteen* devices) from popular hardware and software vendors and supporting the most popular Bluetooth versions. Motivated by our empirical findings, we develop and successfully test an *enhanced* key derivation function for Bluetooth that stops *by-design* our six attacks and their four root causes. We show how to effectively integrate our fix into the Bluetooth standard and discuss alternative implementation-level mitigations. We *responsibly disclosed* our contributions to the Bluetooth SIG.

2.3 Introduction

Bluetooth is a *pervasive* technology for low-power wireless communication [18, 19, 20]. It provides two transports: Bluetooth Classic for high throughput and connection-oriented use cases and Bluetooth Low Energy (BLE) for connectionless and low throughput scenarios. This paper focuses on *Bluetooth Classic*, from now indicated as *Bluetooth*. As billions of devices, such as smartphones, laptops, speakers, headsets, and tablets, daily employ Bluetooth to exchange sensitive data and commands, Bluetooth must provide strong security and privacy guarantees, including confidentiality, integrity and authenticity.

Bluetooth's security and privacy depend on *pairing* and *session establishment*, two mechanisms specified in the *Bluetooth standard* (v5.3) [21]. Devices use pairing to agree upon a long-term secret called the pairing key. Pairing involves user interaction, such as pressing a button or confirming a numeric value on the screen. Paired devices use session establishment to create encrypted and integrity-protected connections, each protected by a *fresh* session key derived from the (static) pairing key and runtime parameters (key diversifiers). Session establishment, unlike pairing, does *not* require user interaction.

Pairing and session establishment have two security modes: (i) *Legacy Secure Connections (LSC)* using legacy cryptographic primitives and procedures, (ii) *Secure Connections (SC)* employing FIPS-compliant ones, such as ECDH, AES-CCM. Pairing and session establishment are *critical* attack surfaces as if they are vulnerable. An adversary can exploit such vulnerability on any (standard-compliant) Bluetooth device. This critical risk motivated extensive research on pairing [17, 22, 23, 24, 25, 26, 27] and session establishment [28, 29].

However, no prior work has investigated Bluetooth's *forward* and *future secrecy* guarantees and their relation with pairing and session establishment. Forward and future secrecy, which enable the defense of past and future messages from key compromise attacks, are *not* even discussed by the Bluetooth standard. We extrapolated these properties via a careful analysis of the standard. We inferred that Bluetooth should provide forward and future secrecy among sessions if the pairing key stays secret. Hence, an attacker compromising the current session key should not be able to decrypt data from past (i.e., forward secrecy) and future sessions (i.e., future secrecy). Then we questioned this assumption and uncovered that, instead, sessions' forward and future secrecy *can* be broken by stealthily attacking *session key derivation* at the protocol level, *without* knowing the pairing key or triggering a new (suspicious) pairing event.

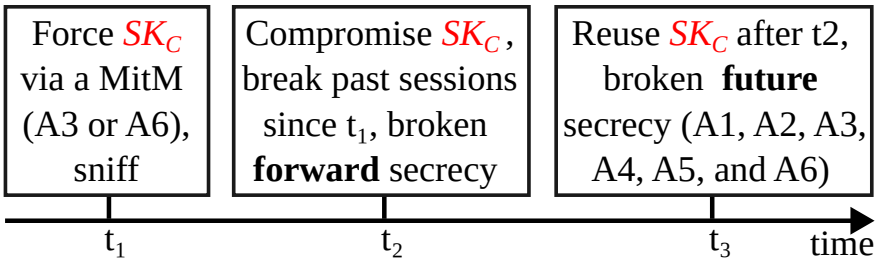

Figure 2.1: BLUFFS attack timeline.

2.4 Design

We present the **BLUFFS attacks**, six novel attacks breaking Bluetooth's forward and future secrecy by targeting session establishment. The attacks exploit an attack strategy forcing LSC session establishment and manipulating in novel ways its key derivation to *reuse* a key known to the attacker across sessions. The attacker first installs a weak session key, then spends some time brute-forcing it, and reuses it to impersonate or machine-in-the-middle (MitM) a victim in subsequent sessions (breaking future secrecy) and decrypt data from past sessions (breaking forward secrecy). We decline the attack strategy in six attack scenarios related to the victim's connection role (i.e., initiator or responder) and Bluetooth security mode (i.e., LSC or SC).

The BLUFFS attacks break Bluetooth's session forward and future *without* compromising prior (strong) SKs negotiated by the victims. We consider forward (future) secrecy broken if Charlie compromises past (future) sessions once SK_C is brute-forced (i.e., compromised). As shown by the timeline in Figure 2.1, the attacker at time t_1 mounts a MitM attack forcing SK_C (A3 or A6), captures the traffic on the current and subsequent sessions, and starts brute forcing SK_C. At $t_2 > t_1$ she brute forces (compromises) SK_C and decrypts all past messages exchanged since t_1 violating forward secrecy. At $t_3 > t_2$ she reuses SK_C to impersonate or MitM Alice and Bob across the next sessions (A1, A2, A3, A4, A5, and A6). Hence, she breaks future secrecy by violating the sessions' confidentiality, integrity, and authenticity from t_2 onwards.

2.5 Vulnerabilities

The BLUFFS attacks' root causes are *four* architectural vulnerabilities in the specification of Bluetooth session establishment (i.e., **RC1**, **RC2**, **RC3**, and **RC4**) [21]. **RC1** and **RC2** are *novel* as they are the first targeting SK derivation and allowing to derive the same SK across sessions (breaking their forward and future secrecy). While **RC3** and **RC4** have been

exploited to attack other session establishment phases. For instance, the BIAS attacks [29] employ them to bypass PK authentication, while the KNOB attacks [28] take advantage of them to downgrade the entropy of SK.

RC1: LSC SK diversification is unilateral (new). The Bluetooth LSC SKDF derives a SK using static inputs (i.e., PK, BA) and variable ones (i.e., AC, SE, SD). The variable inputs diversify SKs across sessions. One would expect that both the Central and the Peripheral would contribute to SK diversification. However, the standard allows the *Central* to set all the SK diversification values. Hence, an attacker impersonating a Central (or role switching to a Central when impersonating a Peripheral) can *unilaterally drive* SK diversification (across sessions). We note that the Peripheral's Bluetooth address is unusable as a variable input because Bluetooth (Classic) does *not* support randomized link-layer addresses.

RC2: LSC SK diversification does not use nonces (new). SK is diversified using random numbers (AC [21, p. 625] and SD [21, p. 637]) and a positive integer (SE in [21, p. 962]). As none of them is a nonce, they can be *reused* in past, present, and future sessions without violating the standard. Hence, an attacker who knows a triplet (AC_C, SE_C, SD_C) and the corresponding SK_C, can force the victims to derive the same *attacker-controlled* session key across sessions.

RC3: LSC SK diversifiers are not integrity protected. The variable inputs exchanged during SK derivation are sent without integrity protection. As a result, an attacker who is spoofing a device or performing MitM on a session can manipulate AC, SE, and SD, without being detected.

RC4: Downgrading SC to LSC does not require authentication. The negotiation of SC or LSC is not integrity-protected. Hence, an attacker can always downgrade a session to LSC, regardless of SC support from the victim, and trigger LSC key negotiation and KDF_{LSC}.

Root causes and attacks. Table 2.1 shows how the six BLUFFS attacks map to the root causes. All attacks take advantage of **RC1**, **RC2**, and **RC3** as they unilaterally derive a constant session key without using a nonce and manipulating the integrity of the session key diversifiers. **RC4** is exploited by the three BLUFFS attacks targeting SC to downgrade a session to LSC.

2.6 Implementation

We develop the `BLUFFS` toolkit to perform and detect the BLUFFS attacks automatically and with low effort. The toolkit provides an *attack device* module requiring open-source

Table 2.1: Mapping the six BLUFFS attacks to their four root causes.

BLUFFS attack	RC1	RC2	RC3	RC4
A1: Spoofing a LSC Central	✓	✓	✓	×
A2: Spoofing a LSC Peripheral	✓	✓	✓	×
A3: MitM LSC victims	✓	✓	✓	×
A4: Spoofing a SC Central	✓	✓	✓	✓
A5: Spoofing a SC Peripheral	✓	✓	✓	✓
A6: MitM SC victims	✓	✓	✓	✓

software, a Linux laptop, and a Cypress/Infineon CYW20819 board [30]. We provide *seven* new patches for the board's closed-source firmware enabling monitoring and tampering with Bluetooth session key derivation. Moreover, our *attack checker* module cleverly parses and analyzes session establishment messages, aka Link Manager Protocol (LMP) packets from a pcap file to automatically compute session keys and detect our attacks.

2.7 Evaluation

We demonstrate that the BLUFFS attacks are *effective on a large scale* by evaluating eighteen devices embedding seventeen unique Bluetooth chips. We successfully exploited a broad set of devices (e.g., laptops, smartphones, headsets, and speakers), operating systems (e.g., iOS, Android, Linux, Windows, and proprietary OSes), Bluetooth stacks (e.g., BlueZ, Gabeldorsche, Bluedroid, and proprietary ones), vendors (e.g., Intel, Broadcom, Cypress, Cambridge Silicon Radio, Infineon, Bestechnic, Apple, Murata, Universal Scientific Industrial, Samsung, Dell, Google, Bose, Logitech, Xiaomi, Lenovo, Jaybird, and Qualcomm), and Bluetooth versions (e.g., 5.2, 5.1, 5.0, 4.2, and 4.1).

Table 2.2 presents our evaluation results obtained by testing the six BLUFFS attacks on *eighteen* heterogeneous and popular devices (second column) embedding *seventeen* unique Bluetooth chips (first column) and employing the most popular Bluetooth versions (third column). The last six columns contain a ✓ if a device is vulnerable to an attack; otherwise, a ×. The fourth, fifth, and sixth columns show CI, PI, and MitM attacks when the spoofed victim supports LSC (i.e., A1, A2, and A3). The last three columns report CI, PI, and MitM attacks while impersonating an SC device (i.e., A4, A5, and A6).

LSC Victims. As shown by the first six rows in Table 2.2, *all* tested LSC chips and devices are vulnerable to the six attacks, with one exception. The Logitech BOOM 3 speaker is *not* vulnerable to the PI attacks (A2, A5), as it requires the Central to authenticate PK,

Table 2.2: BLUFFS attacks evaluation results. **Notes**: [1]ask to authenticate as a Central, [2]does not check authentication response (CR), [3]vulnerable SK downgrade with 1 byte of entropy, [4]does not allow LSC session establishment if paired with SC. **Acronyms**: USI stands for Universal Scientific Industrial, CYW for Cypress, BCM for Broadcom, and CSR for Cambridge Silicon Radio. A n/a in the Chip column indicates that the chip SoC model is unavailable from public sources.

Chip	**Device(s)**	**BTv**	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**
LSC Victims								
Bestechnic BES2300	Pixel Buds A-Series[3]	5.2	✓	✓	✓	✓	✓	✓
Apple H1	AirPods Pro	5.0	✓	✓	✓	✓	✓	✓
Cypress CYW20721	Jaybird Vista	5.0	✓	✓	✓	✓	✓	✓
CSR/Qualcomm BC57H687C	Bose SoundLink[1,2]	4.2	✓	✓	✓	✓	✓	✓
Intel Wireless 7265 (rev 59)	Thinkpad X1 3rd gen	4.2	✓	✓	✓	✓	✓	✓
CSR n/a	Logi BOOM 3[1]	4.2	✓	×	✓	✓	×	✓
SC Victims								
Infineon CYW20819	CYW920819EVB-02	5.0	✓	✓	✓	✓	✓	✓
Cypress CYW40707	Logi MEGABLAST	4.2	✓	✓	✓	✓	✓	✓
Qualcomm Snapdragon 865	Mi 10T[4]	5.2	✓	✓	✓	×	×	×
Apple/USI 339S00761	iPhones 12[4], 13[4]	5.2	✓	✓	✓	×	×	×
Intel AX201	Portege X30-C[4]	5.2	✓	✓	✓	×	×	×
Broadcom BCM4389	Pixel 6[4]	5.2	✓	✓	✓	×	×	×
Intel 9460/9560	Latitude 5400[4]	5.0	✓	✓	✓	×	×	×
Qualcomm Snapdragon 835	Pixel 2[4]	5.0	✓	✓	✓	×	×	×
Murata 339S00199	iPhone 7[4]	4.2	✓	✓	✓	×	×	×
Qualcomm Snapdragon 821	Pixel XL[4]	4.2	✓	✓	✓	×	×	×
Qualcomm Snapdragon 410	Galaxy J5[4]	4.1	✓	✓	✓	×	×	×

thus preventing the attacker from completing session establishment (despite eventually being able to reuse SK_C). The Bose SoundLink speaker also asks the Central to authenticate but is still vulnerable to A2 and A5 as it does *not* check the challenge response. The Google Pixel Buds A-Series (2021) are still vulnerable to the KNOB downgrade resulting in SK_C with *1 byte* of entropy; we reported this worrisome finding to Google and got a "will not fix" response.

SC Victims. The last *eleven* rows in Table 2.2 shows our findings about chips and devices supporting *SC*. If the spoofed victim supports LSC, all chips/devices are vulnerable to the CI, PI, and MitM attacks (A1, A2, A3). Hence, an attacker can impersonate any chip/device from the LSC block of rows to any chip/device in the SC set. If we impersonate an SC device, the CYW20819 and CYW40707 chips are vulnerable to A4, A5, and A6, demonstrating that the attacks are effective against SC. Instead, the other eight chips/de-

vices we tested are *not* vulnerable to A4, A5, and A6, as the chips enforce SC between pairing and session establishment, preventing the attacker from downgrading the session to LSC. But, they are still vulnerable to A1, A2, and A3 because of the vulnerabilities we uncover with LSC.

Evaluation impact. Driven by our empirical results shown in Table 2.2, we are convinced that the BLUFFS attacks are *practical* and have a *large-scale* impact on the Bluetooth ecosystem. In particular, they can target *SC and LSC* devices (e.g., laptops, smartphones, headsets, and speakers) supporting a wide range of *operating systems* (e.g., iOS, Android, Linux, Windows, and proprietary OS), *Bluetooth stacks* (e.g., BlueZ, Gabeldorsche, Bluedroid, and proprietary ones), *vendors* (e.g., Intel, Broadcom, Cypress, Cambridge Silicon Radio, Infineon, Bestechnic, Apple, Murata, Universal Scientific Industrial, Samsung, Dell, Google, Bose, Logitech, Xiaomi, Lenovo, Jaybird, and Qualcomm), and *Bluetooth versions* (e.g., 5.2, 5.1, 5.0, 4.2, and 4.1).

2.8 Countermeasures and Disclosure

Motivated by our evaluation results, we propose an *enhanced* Bluetooth session key derivation function that *stops by-design* our attacks and their root causes. Our countermeasure is *backward compatible* with the Bluetooth standard and adds minimal overheads. Specifically, it reuses standard-compliant crypto primitives (i.e., e_1 and e_3) and link-layer functions (i.e., LMP commands). It requires forty-eight (48) extra bytes over the air and three extra function calls. We successfully test the fix against the BLUFFS attacks at the protocol level and release the fix in our toolkit. We also discuss implementation-specific mitigations that vendors can use to mitigate some BLUFFS attacks.

We responsibly disclosed our findings and toolkit to the Bluetooth Special Interest Group (SIG) in October 2022. The Bluetooth SIG acknowledged our findings, coordinated the disclosure with the affected vendors, and reserved *CVE-2023-24023* for our report. We also reached out to Google, Intel, Apple, Qualcomm, and Logitech. Google scored our report with high severity, awarded us a bounty, and is working on a fix. Intel did the same but scored the report with medium severity. Apple and Logitech acknowledged the report and are working on fixes. Qualcomm has not replied yet. We *open-sourced* our toolkit according to responsible disclosure at https://github.com/francozappa/bluffs.

Chapter 3: E-Spoofer

3.1 Reference

This chapter presents a conference paper from 2023 titled: *E-Spoofer: Attacking and Defending Xiaomi Electric Scooter Ecosystem* [10]. The paper was published in the *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. For more information, have a look at the paper, slides, code, poster, demo, WiSec24 talk, and THCON24 talk.

3.2 Abstract

Xiaomi is the market leader in the electric scooter (e-scooter) segment, with millions of active users. It provides several e-scooter models and Mi Home, a mobile application for Android and iOS to manage and control an e-scooter. Mi Home and the e-scooter interact via Bluetooth Low Energy (BLE). No prior research evaluated the security of this communication channel, as it employs security protocols proprietary to Xiaomi. Exploiting these protocols results in severe security, privacy, and safety issues, e.g., an attacker could steal an e-scooter or prevent the owner from controlling it.

In this work, we fill this research gap by performing the first security evaluation on all proprietary wireless protocols deployed to Xiaomi e-scooters from 2016 to 2021. We identify and reverse-engineer *four* of them, each having ad-hoc Pairing and Session phases. We develop *four* attacks exploiting these protocols at the architectural level, and we call them Malicious Pairing (MP) and Session Downgrade (SD). Both attacks can be performed from proximity if the attacker's machine is within BLE range of the target e-scooter or remotely via a malicious application co-located with Mi Home. An adversary can utilize MP and SD to steal a password-protected and software-locked e-scooter, or to prevent a victim from accessing it via Mi Home. We isolate *six* attack root causes, including the lack of authentication while pairing and improperly enforcing the e-scooter password.

We open-source the ***E-Spoofer*** toolkit. Our toolkit automates the MP and SD attacks and includes a reverse-engineering module for future research. We empirically confirm the effectiveness of our attacks by exploiting three e-scooters (i.e., M365, Essential, and Mi 3), embedding five BLE subsystem boards and eight BLE firmware versions that support all four Xiaomi protocols. We design and evaluate *two* practical countermeasures that address our impactful attacks and their root causes, and we release them as part of `E-Spoofer`. We

Figure 3.1: E-Spoofer legitimate (top) and adversarial (bottom) scenarios.

responsibly disclosed our findings to Xiaomi.

3.3 Introduction

Xiaomi is leading the electric scooter (e-scooter) market [31]. Its ecosystem includes seven e-scooters released in the last seven years (i.e., M365, Pro 1, Pro 2, 1S, Essential, Mi 3, and Mi 4) and the *Mi Home* mobile application for Android [32] and iOS [33]. Mi Home enables a user to manage his e-scooter, e.g., wirelessly lock and unlock it or set a password. Mi Home and the e-scooter communicate via *proprietary application-layer* protocols developed by Xiaomi. These protocols are undocumented, not peer-reviewed, and built on a *Bluetooth Low Energy (BLE)* link layer.

Despite their associated security, privacy, and safety risks, no research work evaluated the security protocols used by Xiaomi to secure the interaction between its e-scooters and Mi Home. Instead, recent work focused on the privacy implications of e-scooter rental apps (including Xiaomi) [34] and on the security of Xiaomi's fitness tracking ecosystem [35]. In our work, we find that Xiaomi protocols can be exploited to (remotely) unlock and steal an e-scooter or permanently prevent its owner from managing it from Mi Home.

3.4 Design

This work presents the *first* security evaluation of the communication channel between Xiaomi's e-scooters and Mi Home. As shown at the top of Figure 3.1, the e-scooter acts as a BLE peripheral (connection responder), while Mi Home is the BLE central (connection initiator). The e-scooter periodically broadcasts BLE advertisement packets to be discovered. These packets contain the e-scooter name, model, security level, and pairing mode activation. Mi Home scans the BLE spectrum and lists all connectable Xiaomi e-scooters nearby. Once connected, the devices exchange data using BLE's Generic Attribute Profile (GATT). The e-scooter exposes a GATT server, which includes the Nordic UART Service and a custom Xiaomi service. On the other hand, Mi Home acts as a GATT client, sending read, write, and subscribe requests to the e-scooter's GATT server. To communicate, Mi Home and the e-scooter establish a BLE link-layer connection. Then, they use *proprietary* application-layer protocols and mechanisms.

As shown in the bottom of Figure 3.1, we focus on *proximity-based* and *remote* attackers. The proximity-based attacker targets the e-scooter with BLE signals and is in range of the target device. They have the following goals: (i) unlock and steal a (password-protected) e-scooter, and (ii) prevent the legitimate owner from accessing and controlling the e-scooter via Mi Home. The remote adversary attacks the e-scooter using a malicious application installed on the victim's smartphone. Thus, they require the victim's smartphone to be within BLE range of the e-scooter, but she can remotely activate the app. For example, the adversary can attack the e-scooter while the victim is parking the e-scooter and walking away from the parking lot. The remote attacker has the same goals as the proximity-based attacker.

We uncover and reverse-engineer all four e-scooter protocols used from 2016 to 2021. We label them as P1, P2, P3, and P4, and we dissect their custom Pairing (i.e., key agreement) and Session phases. As summarized in Table 3.1, we find that P1, P2, and P3 offer no security guarantees but *security through obscurity*. Instead, P4 provides some security properties (e.g., ECDH key agreement and AES-CCM authenticated encryption) but is vulnerable to *downgrade* attacks. Moreover, we find that Xiaomi decided *not* to use standard BLE link-layer security mechanisms (e.g., BLE pairing), despite their devices support them.

We present *four* novel attacks targeting the Xiaomi protocols' specifications. Two attacks enable a proximity-based or remote attacker to pair maliciously with an e-scooter and get authorized access to it *without* spoofing the victim's identity (i.e., MP). The other two attacks allow a proximity-based or remote attacker to downgrade the connection with an e-scooter to an insecure version and send arbitrary commands (i.e., SD). The proximity-based

Table 3.1: E-Spoofer reversed protocols summary.

ID	Name	Pairing	Session
P1	No security	None	None
P2	XOR obf.	Public XOR mask, no auth	XOR mask obf., no auth, no int
P3	AES-ECB and XOR obf.	AES-ECB key agr, no auth	XOR obf., implicit auth, no int
P4v1	ECDH and AES-CCM	ECDH, AES-CCM unil auth	HKDF, AES-CCM, mut auth
P4v2	ECDH and AES-CCM	ECDH, AES-CCM unil auth	P4v1 + downgrade protection

adversary must be in BLE range of the target e-scooter. Instead, the remote adversary must have installed a malicious app on the victim's smartphone. Our attacks achieve *impactful* goals, such as unlocking and stealing an e-scooter or preventing a victim from regaining control of the e-scooter via Mi Home. We isolate the *six* attacks' root causes, including the improper authentication and authorization mechanisms and the unprotected but privileged vendor-specific features of Xiaomi protocols.

3.5 Vulnerabilities

The four presented attacks presented are enabled by *six* vulnerabilities that we isolated in the Xiaomi protocols' specification:

V1: Unauthenticated Pairing. None of the Pairing phases require *device authentication* (e.g., via a certificate signed by Xiaomi). Hence, an attacker can pair with an e-scooter while spoofing an arbitrary Mi Home app without authenticating, regardless of the application-layer protocol used by the e-scooter (i.e., P1, P2, P3, or P4).

V2: Unintentional Pairing mode. Pressing the scooter's headlight button activates pairing mode for *seventeen seconds* without notifying the user. Hence, whenever the victim presses the headlight button, an attacker in the BLE range of the e-scooter can detect that the e-scooter is pairable from its BLE advertisements and pair. Alternatively, given physical access, the attacker can trigger pairing mode while the victim is away by simply pressing the headlight button (even if the e-scooter is software-locked).

V3: Improper e-scooter password enforcement. The e-scooter does *not* enforce the password set by the user via Mi Home. Only Mi Home checks it to prevent unauthorized access to the e-scooter from the victim's smartphone. Therefore, an attacker can tamper with a password-protected e-scooter without knowing the password. Moreover, Mi Home does not provide a way to *deactivate* the password, and the password does *not* change across factory resets. If the adversary changes the e-scooter password, she prevents the victim from controlling the e-scooter via Mi Home.

V4: Unprotected sensitive memory. Xiaomi custom protocols include an unauthenticated command to read and write *sensitive* memory regions. For example, the attacker can read and overwrite the victim's password from the e-scooter DRV subsystem memory. Moreover, she can tamper with the BLE subsystem memory to lock, unlock, reboot, and shut down the e-scooter.

V5: Downgradable and insecure Session. Xiaomi custom protocols include unauthenticated commands to downgrade the Session phase. For instance, the attacker can downgrade a P4v1 Session to a P3 Session and a P2 Session to a P1 Session. At the same time, the P1, P2, and P3 Session phases are insecure and do not guarantee confidentiality, authenticity, or integrity. P1 uses no key, P2 employs XOR-based obfuscation with a constant XOR mask, and P3 uses a slightly more complex yet predictable obfuscation based on AES-ECB and XOR operations.

V6: No BLE security despite device support. Xiaomi does not employ BLE security at the link layer despite device support but relies solely on its custom security mechanisms at the application layer. So, there is no in-depth defense, and the application layer is a single point of failure.

3.6 Implementation

We release `E-Spoofer`, a toolkit capable of performing our four attacks by reimplementing and abusing the four reversed Xiaomi protocols. The toolkit includes *three* extensible modules. Two dedicated modules implement the Malicious Pairing and Session Downgrade attacks. The reverse-engineering (RE) module offers protocol dissectors to decode and build custom Xiaomi packets (e.g., P1, P2, P3, and P4). and useful Frida hooks for Mi Home to intercept and modify the proprietary Xiaomi payloads dynamically.

3.7 Evaluation

We successfully evaluate the attacks in *eight* different attack scenarios covering P1, P2, P3, and P4. Our setup allows testing multiple e-scooter configurations by using *three* modded e-scooters (e.g., M365, Essential, and Mi 3) with *five* BLE subsystems and *eight* BLE firmware.

As shown in Table 3.2 our attacks are effective on all tested e-scooters. In all attack scenarios, we managed to unlock an e-scooter and steal it or lock it and change its password, preventing its legitimate owner from accessing it via Mi Home. These results lead to millions [36] of exploitable devices.

Table 3.2: E-Spoofer evaluation result.

					Proximity		Remote	
BLE Fw	**Proto**	**E-sco**	**BLE Board**	**BLE SoC**	**MP**	**SD**	**MP**	**SD**
BLE072	P1	M365	M365 (Original)	nRF51822 QFAA	✓	-	✓	-
BLE081	P2	M365	M365 (Original)	nRF51822 QFAA	✓	✓	✓	✓
BLE090	P3	M365	Pro 1 (Clone)	nRF51822 QFAA	✓	✗	✓	✗
BLE122	P4v1	M365	M365 (Original)	nRF51822 QFAA	✓	✓	✓	✓
BLE129	P4v1	M365	Pro 2 (Clone)	nRF51822 QFAC	✓	✓	✓	✓
BLE152	P4v1	Essential	Essential (Original)	nRF51822 QFAC	✓	✓	✓	✓
BLE153	P4v1	Mi 3	Mi 3 (Original)	nRF51822 QFAC	✓	✓	✓	✓
BLE157	P4v2	Mi 3	Mi 3 (Original)	nRF51822 QFAC	✓	✗	✓	✗

We identified and disclosed a *UI authentication* bug in Mi Home for Android and iOS during our experiments. From Mi Home v7.6.704 onwards, the user can lock or unlock a password-protected e-scooter *without* entering the password. The cause is a 1 second UI delay between the app wake-up and the password prompt. Since the password is only checked by Mi Home, due to the `improper e-scooter password enforcement`, the attacker can bypass app-based password protection, unlock the e-scooter, and steal it. Xiaomi acknowledged this bug, rewarding us with a bounty, but gave no information about a fix.

3.8 Countermeasures and Disclosure

To fix the four attacks and their six root causes, we developed and tested two usable and low-cost countermeasures and include them in our toolkit. First, we propose a backward-compatible pairing protocol with proper authentication and authorization mechanisms. Second, we provide a script to patch the session downgrade command from an e-scooter BLE firmware. We successfully tested our patch on the M65 and Pro 1 e-scooters, whose BLE firmware is no longer updated by Xiaomi.

We responsibly disclosed our findings multiple times with Xiaomi via their bug bounty program [37]. In October 2022, we reported a UI password bypass issue with Mi Home, Xiaomi acknowledged it and provided a bug bounty. In November 2022, we shared a technical report and the code to reproduce our findings. In December 2022, we provided them with a video of the attacks on actual devices. Xiaomi did not follow up. We conducted our experiments in a controlled environment without involving third-party users and services. We provide our open-source `E-Spoofer` toolkit at https://github.com/Skiti/ESpoofer.

Chapter 4: ADF

4.1 Reference

This chapter presents a journal paper from 2024 titled: *AttackDefense Framework (ADF): Enhancing IoT Devices and Lifecycles Threat Modeling* [11]. The paper was published in the *ACM Transactions on Embedded Computing Systems (ACM TECS)*. For more information, have a look at the paper, code, and tutorial.

4.2 Abstract

Threat modeling (TM) is essential to manage, prevent, and fix security and privacy issues in our society. TM requires a data model to represent threats and tools to exploit such data. Current TM data models and tools have significant limitations that prevent their use in real-world scenarios. For example, it is challenging to TM embedded devices with current data models and tools as they cannot model their hardware, firmware, and low-level software. Moreover, it is impossible to TM a device lifecycle or security-privacy tradeoffs as these data models and tools were developed for other use cases (e.g., software security or user privacy).

We fill this relevant gap by presenting the AttackDefense Framework (ADF), which provides a novel data model and related tools to augment TM. ADF's building block is the AD object that can represent heterogeneous and complex threats. Moreover, ADF provides automation to process a collection of AD objects, including ways to create sets, maps, chains, trees, and word clouds of AD objects. We present `ADF`, a toolkit implementing ADF composed of four modules (Catalog, Parse, Check, and Analyze).

We confirm that the data model and tools provided by ADF are useful by running an extensive set of experiments while threat modeling a crypto wallet and its lifecycle. Our experiments involved seven expert groups from academia and industry, each using the ADF on an orthogonal threat class. The evaluation generated 175 high-quality ADs covering ISA/IEC 62433-4-1 SecDev Lifecycle, side channels, fault injection, microarchitectural attacks, speculative execution, pre-silicon testing, invasive physical chip modifications, Bluetooth protocol and implementation threats, and FIDO2 authentication.

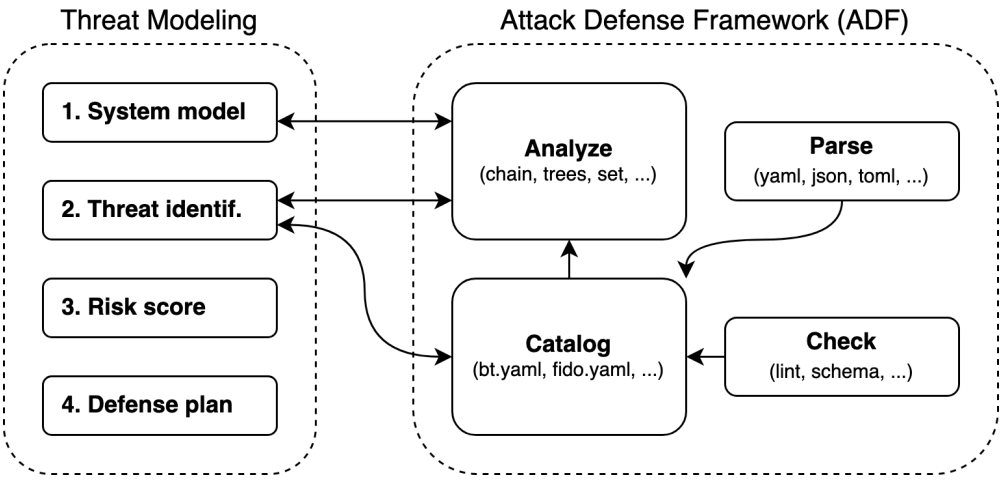

Figure 4.1: ADF high-Level Overview.

4.3 Introduction

Threat Modeling (TM) is essential to manage our society's security and privacy risks. In short, TM allows systematically listing, prioritizing, and addressing digital threats [38, 39, 40], thus providing tangible security benefits compared to other more common but potentially less effective practices, such as compliance with security standards [41, 42, 43]. TM includes data formats to represent threats, tools to process them, and methodologies to identify, prioritize, and address them.

Current TM data formats and tools cannot cover critical aspects of real-world (embedded) devices. For instance, there is no way to accurately represent and process threats related to *hardware* (e.g., invasive physical attacks), *firmware* (e.g., bootloader), *hardware-software interface* (e.g., speculative execution and microarchitectural issues), and *communication protocols* (e.g., protocol- or implementation-level threats). Moreover, they focus on security or privacy attacks on products, neglecting possible *security-privacy tradeoffs*. Other shortfalls are that they cannot model the *products' lifecycle* (e.g., supply chain attacks) and the *defenses* associated with the attacks. Finally, current TM data formats and tools try to be human and machine-friendly, but they usually lack the latter, not allowing for automation and optimization of TM exercises. Hence, the TM community tends to focus on specialized classes of threats (e.g., software security or user privacy) and has limited tooling available.

4.4 Design

We fill this gap by presenting the **AttackDefense Framework (ADF)**, a novel framework to enhance TM coverage, effectiveness, automation, and (re)usability. ADF's building block

Listing 4.1: AttackDefense (AD) Object written in YAML.

```yaml
ad_name:
  # Primary fields
  a: attack
  d:
    policy1: [mech1, mech2]
    policy2: [mech1, mech2]
    ...
  surf: [surf, subsurf, subsubsurf, ...]
  vect: [vector1, vector2, ...]
  model: [model1, model2, ...]
  tag: [tag1, tag2, ...]
  # Optional fields
  risk: [score1, score2, ...]
  year: 2023
  cve: ["123", "456", ...]
  cwe: ["123", "456", ...]
  capec: ["123", "456", ...]
  vref: ["vendor-ref1", ...]
  ...: ...
```

is the *AttackDefense Object (AD)*, a new data structure for representing threats.

Listing 4.1 shows how to write an AD using YAML, but note that other serialization languages, including JSON, TOML, or XML can be used. Each AD has a unique name (`ad_name`), six *primary* fields, and *optional* fields. A field is a key-value pair and supports various data types, including dictionaries, lists, strings, and integers. The AD has six primary fields:

- `a` contains a string describing an attack with an arbitrary level of abstraction (e.g., coarse-grained or fine-grained).
- `d` stores sub-dicts to model different defense strategies for an attack. In particular, each sub-dict encodes a *high-level policy* string (e.g., `policy1`) and a list of *concrete mechanisms* strings (e.g., `[mech1, mech2]`) satisfying such policy. The sub-dicts could be ordered according to some criteria (e.g., from the most effective to the least effective).
- `surf` is an ordered list of strings describing the attack surface (i.e., target). The list is ordered such that each element *narrows down* the attack surface from the broadest to the most specific.
- `vect` is a list of strings containing the attack vectors (i.e., techniques) related to the attack.
- `model` stores the adversary models capable of performing the attack in a list of strings.
- `tag` is a list of strings storing useful metadata, such as the AD type, security-privacy trade-offs, and other technicalities.

Additionally, there are some *optional* fields to enhance the AD:

- `risk` is a list of strings storing risk scores associated with the attack (e.g., CVSS).
- `year` is an int storing the year when the attack was first discovered.
- `cve`, `cwe`, and `capec` are lists of strings storing identifiers from those catalogs related to the attack.
- `vref` is a list of vendor reference strings associated with the attack, including security advisory identifiers from Linux [44] or Android [45].

The AD object satisfies seven requirements that we set based on the state of the art. In particular, it covers attacks and defenses, security and privacy, hardware and firmware, product and lifecycle, and fine- and coarse-grained threats.

R1: Attacks and Defenses. Current TM frameworks focus on the attacker. We want to focus on the attacker and the defender at the same time. This approach enables reasoning about fine-grained and coarse-grained mitigation strategies, identifying critical attacks with and without known defenses, exploring alternative defensive strategy for a specific attack, evaluating defense-in-depth options, and determining the minimum number of defenses required to address an attack.

R2: Security and Privacy. Existing TM frameworks treat security and privacy separately, leading to issues like overlooked security-privacy trade-offs. We want to consider security and privacy simultaneously to capture their unavoidable joint benefits and trade-offs. For instance, we could model a scenario to explore the competing goals of confidentiality and integrity (security) vs. repudiability and traceability (privacy).

R3: Hardware and Firmware. There is a need to broaden the scope of threat modeling to cover hardware and firmware threats, which are relevant but often neglected during TM. Besides traditional TM areas, we want to cover novel hardware and firmware threat classes, including invasive and non-invasive physical attacks like side channels, fault injection, and physical chip manipulations. Furthermore, we want to address threats at the intersection of hardware and software, such as microarchitectural and speculative execution attacks.

R4: Product and Lifecycle. Existing TM frameworks focus on analyzing devices or systems, not their development lifecycle. For instance, current frameworks cannot model hardware and software supply chain attacks. We want to include lifecycle threats in our framework to enable TM practitioners to manage critical process risks such as SolarWinds [46] and Supermicro [47].

R5: Fine- and Coarse-grained Threats. While current TM frameworks focus on generic classes of threats, we want to support threats at different levels of abstraction. For instance, we model coarse-grained threats (e.g., generic attack techniques), such as

buffer overflows, and fine-grained ones (e.g., real-world attack instances), like Heartbleed on OpenSSL. By doing so, we enhance our TM analysis capabilities. For example, we can automatically generate hierarchies of threats based on abstraction levels, including trees, chains, and graphs.

R6: Reusable and Updatable. Present TM frameworks are complex to combine, update, and automate. We want to prevent duplication of the same TM exercises by providing reusable data formats and tools. We wish to interoperate with existing TM methodologies such as STRIDE and LINDDUN. Additionally, the framework must be updated, allowing new attacks and defenses to be incorporated as they become available. We aim to consistently and incrementally add threats over time to cover situations where new threats are identified, or old threats are patched or reintroduced.

R7: Machine- and Human-friendly. Current TM frameworks are either machine- or human-friendly. We want a framework that minimizes friction between humans and machines. Users should be able to read, write, analyze, and share attack and defense strategies. The framework should accommodate users with varying expertise in TM, including developers and threat modeling experts. The framework should enable machines to automatically generate valuable TM outputs such as interactive and portable reports and visualizations. It should also facilitate intelligent storage of these outputs, leveraging techniques such as version control, CI/CD pipelines, and machine-checkable data formats.

The ADF is developed to be reusable and updatable (i.e., future-proof), friendly to machines and humans, and compatible with any TM methodology (e.g., STRIDE, LINDDUN, and ATree). An AD object can be written in any serialization language. We recommend using YAML or JSON. Moreover, we implement valuable automation on the AD objects, unlocking novel TM capabilities. For example, we show how to create *flat* AD sets or maps or *hierarchical* AD chains, trees, or word clouds. These capabilities significantly increase TM's effectiveness, coverage, and speed.

4.5 Implementation

We implement the ADF design in the `ADF` toolkit that contains *four* modules: Catalog, Check, Analyze, and Parse. Figure 4.1 shows a high-level overview of the ADF. Catalog contains the ADs that we develop in our case studies. Parse can extract ADs from YAML, JSON, TOML, and XML files and can be easily extended to parse other file types. Check automatically validates the syntax, semantics, and content of the ADs using a combination of checkers such as yamllint and Python schema. Analyze provides functions to automatically process ADs to generate, among others, ADs' sets, maps, trees, wordclouds, and chains. We

will open-source our toolkit with a permissive license to let other individuals take advantage of ADF and provide feedback.

4.6 Evaluation

We describe the results of *seven case studies* run by industrial and academic expert groups covering a broad spectrum of threats using a *crypto wallet* and its *life cycle* as a reference. In particular, we evaluate attacks and defenses related to ISA/IEC 62433-4-1 SecDev Lifecycle, side-channels, fault injection, microarchitectural attacks, speculative execution, pre-silicon testing, invasive physical chip modifications, Bluetooth protocol and implementation, and FIDO2 authentication. As a result of our evaluation, the seven experts created and used 175 ADs and provided invaluable feedback to improve the ADF. Our toolkit is open-source and available at https://github.com/francozappa/adf.

Chapter 5: FP-tracer

5.1 Reference

This chapter presents a journal paper from 2024 titled: *FP-tracer: Fine-grained Browser Fingerprinting Detection via Taint-tracking and Multi-level Entropy-based Thresholds* [12]. The paper was published in the *Privacy Enhancing Technologies Symposium (PETS)*. For more information, have a look at the paper, slides, code, and PETS24 talk.

5.2 Abstract

Browser fingerprinting is an effective technique to track web users by building a fingerprint from their browser attributes. It is also stealthy because the tracker uses legitimate JavaScript API calls offered by the browser engine, which can be obfuscated before they are sent to a (third-party) server. Current browser fingerprinting detection methodologies employ limited collection and classification techniques, such as binary classification of fingerprinters based on the number of non-obfuscated exfiltrated attributes. As a result, they produce inconsistent findings. Meanwhile, the privacy of millions of web users is at risk daily.

We address this gap by presenting FP-tracer, a novel methodology to detect and classify browser fingerprinters based on dynamic taint tracking and joint entropy classification. Our methodology enables detecting first- and third-party fingerprinters even when they use obfuscation by tainting attributes, propagating them, and logging when they are leaked (via 62 sources and 25 sinks). Moreover, it discriminates the invasiveness of fingerprinting activities, even from the same service, by measuring the joint entropy of the collected attributes and clustering them.

We implement FP-tracer by extending Foxhound, a privacy-oriented Firefox fork with numeric type tainting, more taint tracking sources and sinks, support for multiple sources, and better logging capabilities. We embed our implementation in our automated crawling infrastructure, which is capable of testing websites in parallel using programmable and reproducible logic. We will open-source our implementation.

We evaluate FP-tracer by performing a large-scale crawl over the Tranco Top 100K, and detect, amongst others, audio, canvas, and storage fingerprinting on the web. Among others, we find high fingerprinting activities in 8% of domains, with more moderate activity reaching 75%. Notably, third-party scripts are almost five times more likely to perform

Listing 5.1: JavaScript Browser fingerpriting.

```
// Collection
let height = screen.height;
let width = screen.width;?\tikzmark{width}?
let userAgent = navigator.userAgent;
// Aggregation
let resolution = width?\tikzmark{resw}? * height?\tikzmark{resh}?;
let fingerprint = userAgent + resolution.toString();
let userId = MD5hash(fingerprint);
// Exfiltration
let requestUrl = 'https://example.com?userId=' + userId;
fetch(requestUrl);
```

fingerprinting for high activity levels. In addition, we measure that the most severe category of fingerprinting obfuscates 46% of transmitted attributes, and 38% of fingerprinters involve two or more domains. Finally, we find that existing consent banners do not provide an effective defense against browser fingerprinting.

5.3 Introduction

Browser fingerprinting is a stealthy technique used to uniquely identify users across the web by crafting fingerprints from browser attributes accessible through its JavaScript APIs. These APIs give access to fingerprintable browsers' configurations, plug-ins, screen dimensions, and installed fonts. Browser fingerprinters are hard to detect because they use legitimate JavaScript API calls, and they generate web traffic that looks benign. On the other hand, they are violating the privacy of millions of Internet users [48, 49, 50]. For instance, the following listing shows a fingerprinting script collecting three attributes, aggregating them using math operations and hashing, and then exfiltrating the resulting fingerprint.

Several browser fingerprinting detection methodologies have been developed. But their experimental results are *inconsistent*, their collection strategy is *coarse-grained*, and their classification methods are *binary*. For instance, the state of the art has varying opinions on the amount of browser fingerprinting on popular websites, with reported rates ranging from 10% [51, 52] to 70% [49, 53]. Course-grained techniques based on browser API monitoring or code analysis [54, 55, 56, 57] cannot provide details on the destination of collected attributes. In addition, binary classification based on attribute counting [53] or machine learning [52] are either too restrictive or too liberal and only provide part of a broader picture.

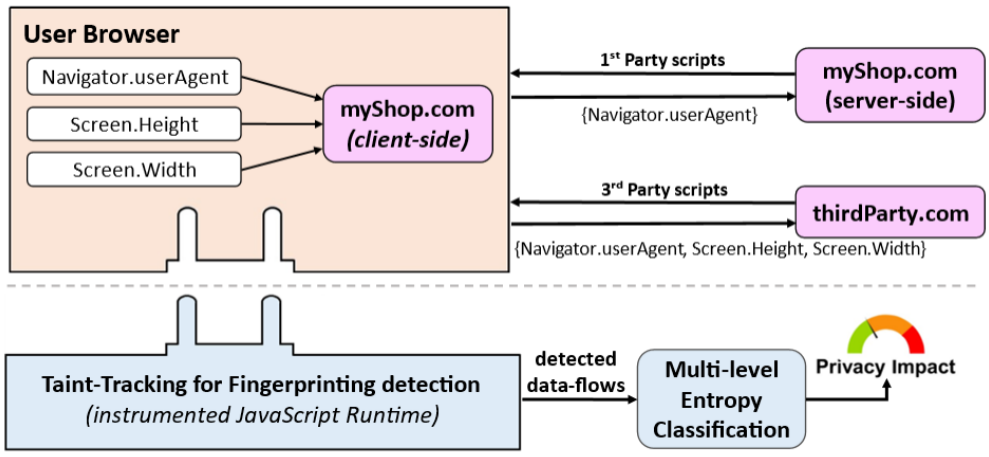

Figure 5.1: FP-tracer's high-level overview.

5.4 Design

We address these three relevant gaps by presenting *FP-tracer*, an innovative browser fingerprinting analysis and detection methodology based on *fine-grained taint-tracking* and *joint entropy thresholds*. As shown in Figure 5.1, FP-tracer analyzes client-side JavaScript code from first and third-party websites and instruments it to enable *dynamic taint tracking*.

This allows for the detection of *data flows* from sensitive browser attributes (sources), such as `Navigator.userAgent`, into JavaScript APIs (sinks), such as `img.src`, which are sent to first- and third-party domains. Then, FP-tracer computes the *joint entropy* of all attributes sent to a particular domain and classifies its fingerprinting activity into six categories ranging from no activity to very high activity to provide a privacy impact score. FP-tracer supports 62 sources, including canvas, audio, and storage APIs, and 25 sinks, including XHR requests, element attributes such as `src` and the `sendBeacon` API.

5.5 Implementation

We implemented our methodology by *extending Foxhound* [58], an open-source Firefox fork developed for client-side web vulnerability scanning. We enhanced its instrumented JavaScript engine to support numeric data types. We added support for multiple sources using set operations. We improved Foxhound's list of supported sinks and sources and automated their addition using Firefox's WebIDL C++ code generator. Finally, we extended Foxhound's logging capabilities. We also implemented an automated *crawling* infrastructure based on Playwright and integrated our Foxhound extension into the crawler. Our crawler

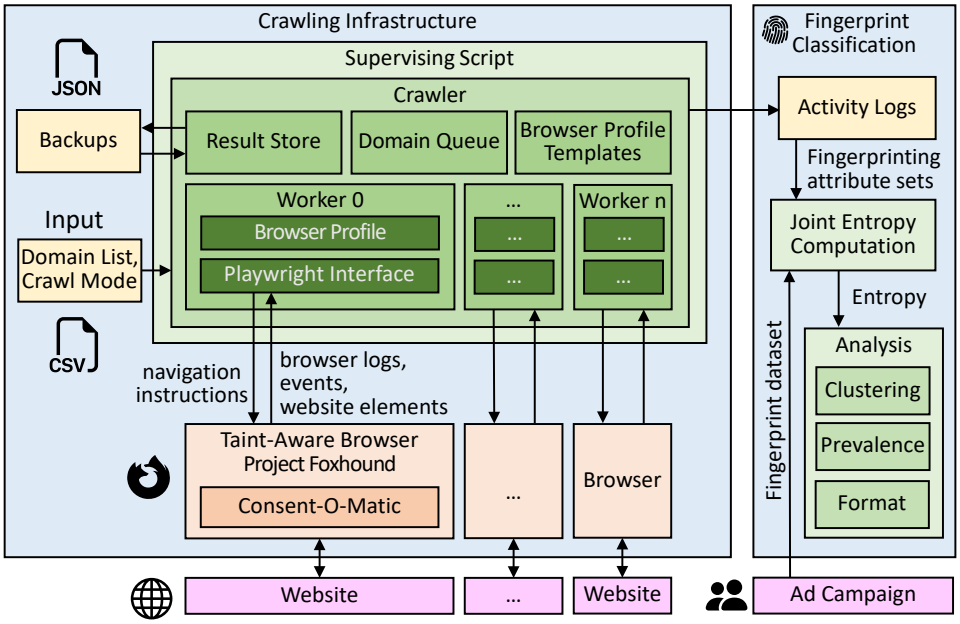

Figure 5.2: FP-tracer crawler.

can automatically and reproducibly visit multiple websites using parallel browser instances and a programmable page visit logic. Each instance logs the fingerprinting activities and can be stopped, restarted, or restored in case of a crash.

We built our multi-threshold joint entropy classification using a real-world dataset of approximately 86k fingerprints. The dataset was collected in [59] by running a large-scale experiment on consenting users. Each user browser was fingerprinted using ads containing JavaScript code, which accessed sensitive browser attributes. We found that the choice of attributes is critical to the entropy and, therefore, the extent to which users can be identified. For example, while some sites collect up to 25 attributes, similar entropies can be obtained with just seven carefully chosen ones.

To scale our fingerprinting collection technique to a large number of websites, we integrated our modified Foxhound browser into the Microsoft Playwright browser automation framework (v1.21.1). We then created a node.js crawler application for automated website browsing. As presented in Figure 5.2, our crawler takes a list of target domains as input, which are added to a queue. A scalable number of workers manage the queue, each using a Playwright-controlled browser to visit websites and log fingerprinting activity.

Our crawler implements an *automatic* and *re-usable* website visiting logic. We visit each target website's top-level (home) page, followed by visits to three subpages chosen randomly from hyperlinks on the home page. The duration of the visits to a page is carefully structured: the home page loads first, and the crawler waits for 20 seconds with a mid-point scroll to

the bottom.

As shown by previous work [51], fingerprinting scripts may first be active once the user navigates to secondary pages. As such, we subsequently revisit the home page, followed by the three randomly selected secondary pages. The crawler waits 10 seconds on each page with a mid-point scroll. A transition to `about:blank` and a 5-second pause occurs between each page to ensure a clear separation of measurement data.

The crawler aggregates activity logs from multiple workers into a single output and allows backup and restoration of the crawling state in the event of expected errors. We log all fingerprinting-related dataflows for each web page visited during the crawl. We then aggregate this information to build the attributes sent to a particular destination domain from all data flows detected on the crawled domain (referred to as a *domain-destination pair*). The attributes are then used as input for the subsequent analysis stage.

5.6 Evaluation

We evaluated FP-tracer with a large-scale fingerprinting experiment visiting 80 618 *domains* from the Tranco Top 100K. Using our fine-grained collection approach, we observed 269 784 fingerprinting flows exfiltrating 15 239 unique browser attribute combinations. With our joint entropy classification technique, we found five types of fingerprinting activities that we label as Negligible, Low, Medium, High, and Very High. We also extract the attribute vectors associated with each activity (e.g., `userAgent` and `storageEstimate` have very high joint entropy).

Our study revealed *novel insights* offering a consistent reading of the inconsistent results published so far and demonstrating the importance of our collection and classification approaches. For example, we found that 8% of domains perform fingerprinting in the very high category. This is comparable to the lower rates presented in prior studies [51, 52] at around 10%. Additionally, we observed more moderate fingerprinting activity in 75% of the successfully crawled domains, aligning with the approximate 70% prevalence rates reported previously [49, 53].

Or, for a high level of activity, we find that fingerprinting is almost five times more likely to be performed by a third-party script than a first-party one. By examining over 6 million string values transmitted in our sample, we find that while up to 90% of fingerprinting attributes are transmitted in plain text, very high severity scripts perform some form of obfuscation in 47% of the time. We also find that in this very high category, 38% of fingerprinting is performed by a collusion of two or more domains. We saw two websites sending attributes to a single destination with scripts from seven different domains.

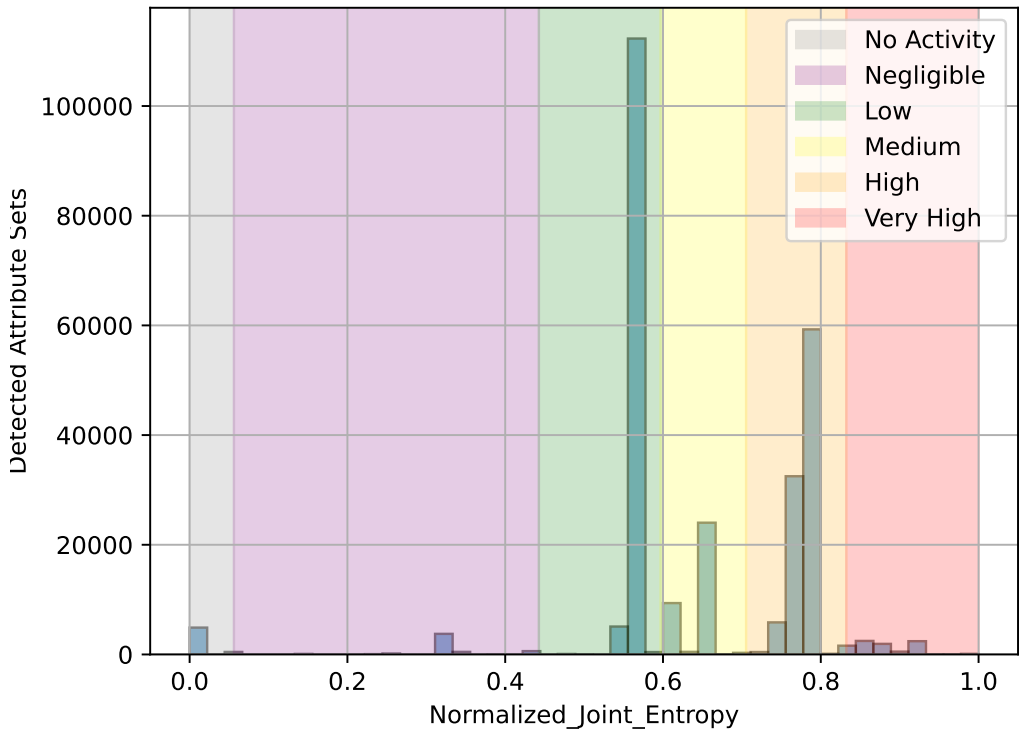

Figure 5.3: Detected tainted attribute sets per normalized joint entropy cluster.

Figure 5.3 shows the normalized joint entropy of the tainted attributes we collected during our experiments. We find a total of 269 784 transmissions between crawled and destination domains. We bin the values using six intervals: No Activity, Negligible, Low, Medium, High, and Very High. We set the thresholds using the Jenks natural breaks algorithm. We also performed clustering using the K-means algorithm and found only minimal differences between the intervals obtained. We find some activity below the normalized entropy of $H_n < 0.5$, with very distinct structure and peaks between $0.5 < H_n < 0.8$, followed by a tail up to 0.989. Note that there is some activity with $H_n = 0$, which, interestingly, is caused by scripts transmitting deprecated attributes such as `Navigator.appName`, which will always return the string `Netscape`.

We validate FP-tracer against the popular Disconnect [60] and EasyPrivacy [61] lists. Suppose our high and very high grouping broadly aligns with these lists. Our results also identify intensive fingerprinting activities for domains only tagged as general (not invasive) fingerprinters in Disconnect. We also agree when comparing our findings with results from FP-inspector [52]. Of the 911 domains crawled by both studies, we find at least a moderate fingerprinting activity in 95% of cases.

Chapter 6: Conclusion

This thesis tackles the ambitious and unsolved problem of secure and privacy-preserving connected systems. It focuses on four challenges: standard communication protocols, proprietary device ecosystems, threat modeling and risk assessemt, and device and user tracking.

Chapter 2 demonstrates that current communication standards used by billions of devices are still vulnerable to critical design flaws, and standardization bodies are reluctant to fix them promptly. This calls for developing better methods to write and test a protocol specification. Specifically, for Bluetooth, we are working on a next-generation Bluetooth security protocol suite that is currently under submission and will be shared with the Bluetooth Special Interest Group (SIG).

Chapter 3 shows that leading IoT device ecosystems still adopt security through obscurity approaches and ignore standard and battle-tested security protocols and mechanisms. Vendors who adopt proprietary S&P mechanisms should take responsibility for the related vulnerabilities and adopt a more transparent approach. We are developing more efficient reverse-engineering techniques and frameworks to quickly and reliably reconstruct proprietary security protocols.

Chapter 4 highlights the need for a more holistic threat modeling framework that considers multiple facets of connected systems, like attack and defense, security and privacy, hardware and software, and devices and lifecycles. Threat modeling should become a pillar of S&P research, along with software security, hardware security, cryptography, and privacy. We are pushing the ADF for standardization with national (BSI, ANSSI, CNI) and international bodies (ENISA).

Chapter 5 proves that web tracking is more alive than ever, despite recent privacy regulations, including GDPR. Modern trackers can bypass conventional tracking methods, like cookies and advertising ID, by fingerprinting a web browser using hardware and software details. These hard truths are wake-up calls for further system security and privacy research. We are working on EntroShield, a browser plugin that offers strong tracking detection and prevention (like FP-tracer) but is also deployable on consumer browsers.

In the future, we will continue working on the *open-source hardware revolution* sustained by academic and industrial efforts such as ORSHIN, RISC-V, and OpenTitan. Our devices will include open, reprogrammable, and specialized hardware, including chiplets, neural processing units, dedicated security chips, and quantum processors. This setup will bring new opportunities and security and privacy challenges. For example, how can we keep a secret in hardware and simultaneously open-source its design and implementation? How

can we build trustworthy hardware that can also be reverse-engineered? How can we secure a connected system where device hardware looks like software? *Soft hardware*, qu'est-ce que c'est?

Bibliography

[1] Statista, "Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2033." https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/, 2025.

[2] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," *Computer Networks*, vol. 192, p. 108040, 2021.

[3] P. Delgado-Santos, G. Stragapede, R. Tolosana, R. Guest, F. Deravi, and R. Vera-Rodriguez, "A survey of privacy vulnerabilities of mobile device sensors," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–30, 2022.

[4] P. M. Rao and B. D. Deebak, "Security and privacy issues in smart cities/industries: technologies, applications, and challenges," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 8, pp. 10517–10553, 2023.

[5] C. Organization, "Common Criteria." https://www.commoncriteriaportal.org/index.cfm, 2025.

[6] NIST, "FIPS 140-3: Security Requirements for Cryptographic Modules." https://csrc.nist.gov/pubs/fips/140-3/final, 2022.

[7] EU, "GDPR: General Data Protection Regulation." https://gdpr-info.eu/, 2025.

[8] W. Central, "A PR disaster: Microsoft has lost trust with its users, and Windows Recall is the straw that broke the camel's back." https://www.windowscentral.com/software-apps/windows-11/microsoft-has-lost-trust-with-its-users-windows-recall-is-the-last-straw, 2024.

[9] D. Antonioli, "BLUFFS: Bluetooth Forward and Future Secrecy Attacks and Defenses," in *ACM conference on Computer and Communications Security (CCS)*, November 2023.

[10] C. Marco, C. Riccardo, L. Eleonora, C. Mauro, and A. Daniele, "E-Spoofer: Attacking and Defending Xiaomi Electric Scooter Ecosystem," in *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2023. Award: 2nd prize best Master Thesis from Italian Association for Information Security (CLUSIT).

[11] T. Sacchetti, M. Bognar, J. De Meulemeester, B. Gierlichs, F. Piessens, V. Bezsmertnyi, M. C. Molteni, S. Cristalli, A. Gringiani, O. Thomas, *et al.*, "AttackDefense Framework (ADF): Enhancing IoT Devices and Lifecycles Threat Modeling," *ACM Transactions on Embedded Computing Systems (ACM TECS)*, 2024.

[12] S. Boussaha, L. Hock, M. Bermejo, R. C. Rumin, A. C. Rumin, D. Klein, M. Johns, L. Compagna, D. Antonioli, and T. Barber, "FP-tracer: Fine-grained Browser Fingerprinting Detection via Taint-tracking and Multi-level Entropy-based Thresholds," in *Privacy Enhancing Technologies Symposium (PETS)*, July 2024.

[13] M. Casagrande, R. Cestaro, E. Losiouk, M. Conti, and D. Antonioli, "E-Trojans: Ransomware, Tracking, DoS, and Data Leaks on Battery-powered Embedded Systems," 2024.

[14] M. Casagrande and D. Antonioli, "CTRAPS: CTAP Impersonation and API Confusion Attacks and Defenses on FIDO2," in *IEEE European Symposium on Security and Privacy (Euro S&P)*, July 2025.

[15] M. Casagrande, E. Losiouk, M. Conti, M. Payer, and D. Antonioli, "BreakMi: Reversing, Exploiting and Fixing Xiaomi Fitness Tracking Ecosystem," in *IACR Transactions on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 330–366, 2022.

[16] D. Antonioli and M. Payer, "On the Insecurity of Vehicles Against Protocol-Level Bluetooth Threats," in *2022 IEEE Security and Privacy Workshops (SPW)*, pp. 353–362, IEEE, 2022.

[17] D. Antonioli, N. O. Tippenhauer, K. Rasmussen, and M. Payer, "BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy," in *Proceedings of the Asia conference on computer and communications security (ASIACCS)*, May 2022.

[18] Bluetooth SIG, "Bluetooth Market Update 2022." https://www.bluetooth.com/2022-market-update/, 2022.

[19] Bluetooth SIG, "Bluetooth Market Update 2021." https://www.bluetooth.com/bluetooth-resources/2021-bmu/, 2021.

[20] Bluetooth SIG, "Bluetooth Market Update 2020." https://www.bluetooth.com/bluetooth-resources/2020-bmu/, 2020.

[21] Bluetooth SIG, "Bluetooth Core Specification v5.3." https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=521059, 2021.

[22] M. von Tschirschnitz, L. Peuckert, F. Franzen, and J. Grossklags, "Method confusion attack on Bluetooth pairing," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1332–1347, IEEE, 2021.

[23] E. Biham and L. Neumann, "Breaking the Bluetooth Pairing–Fixed Coordinate Invalid Curve Attack." http://www.cs.technion.ac.il/~biham/BT/bt-fixed-coordinate-invalid-curve-attack.pdf, 2018.

[24] D.-Z. Sun, Y. Mu, and W. Susilo, "Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard v5. 0 and its countermeasure," *Personal and Ubiquitous Computing*, vol. 22, no. 1, pp. 55–67, 2018.

[25] K. Haataja and P. Toivanen, "Two practical man-in-the-middle attacks on Bluetooth Secure Simple Pairing and countermeasures," *Transactions on Wireless Communications*, vol. 9, no. 1, pp. 384–392, 2010.

[26] A. Y. Lindell, "Attacks on the pairing protocol of Bluetooth v2.1," *Black Hat USA, Las Vegas, Nevada*, 2008.

[27] K. Hypponen and K. M. Haataja, "Nino man-in-the-middle attack on Bluetooth Secure Simple Pairing," in *Proceedings of the International Conference in Central Asia on Internet*, pp. 1–5, IEEE, 2007.

[28] D. Antonioli, N. O. Tippenhauer, and K. B. Rasmussen, "The KNOB is Broken: Exploiting Low Entropy in the Encryption Key Negotiation Of Bluetooth BR/EDR," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1047–1061, 2019.

[29] D. Antonioli, N. O. Tippenhauer, and K. Rasmussen, "BIAS: Bluetooth impersonation attacks," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 549–562, IEEE, 2020.

[30] Infineon, "CYW20819." https://www.infineon.com/cms/en/product/wireless-connectivity/airoc-bluetooth-le-bluetooth-multiprotocol/airoc-bluetooth-le-bluetooth/cyw20819/, 2022.

[31] iResearch, "iResearch Coverage On Xiaomi." http://www.iresearchchina.com/Upload/201808/20180824143739_3256.pdf, 2018.

[32] L. Beijing Xiaomi Mobile Software Co., "Mi Home (Android)." https://play.google.com/store/apps/details?id=com.xiaomi.smarthome, 2022.

[33] L. Beijing Xiaomi Mobile Software Co., "Mi Home (iOS)." https://apps.apple.com/us/app/mi-home-xiaomi-smart-home/id957323480, 2022.

[34] N. Vinayaga-Sureshkanth, R. Wijewickrama, A. Maiti, and M. Jadliwala, "An Investigative Study On The Privacy Implications Of Mobile E-Scooter Rental Apps," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, (New York, NY, USA), p. 125–139, Association for Computing Machinery, 2022.

[35] M. Casagrande, E. Losiouk, M. Conti, M. Payer, and D. Antonioli, "BreakMi: Reversing, Exploiting And Fixing Xiaomi Fitness Tracking Ecosystem," *IACR Transactions On Cryptographic Hardware And Embedded Systems*, vol. 2022, no. 3, p. 330–366, 2022.

[36] E. P. from InsideEVs, "Segway-Ninebot Has Sold More Than One Million E-Scooters In China." https://insideevs.com/news/613420/segway-ninebot-one-million-sold-china/, 2022.

[37] Xiaomi, "Xiaomi Bug Bounty Program On HackerOne." https://hackerone.com/xiaomi?type=team, 2022.

[38] C. Salter, O. S. Saydjari, B. Schneier, and J. Wallner, "Toward a secure system engineering methodology," in *Proceedings of the 1998 workshop on New security paradigms*, pp. 2–10, 1998.

[39] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[40] I. Tarandach and M. J. Coles, *Threat modeling: a practical guide for development teams*. O'Reilly, 2021.

[41] M. Hardy, "Beyond continuous monitoring: Threat modeling for real-time response," *SANS Institute*, 2012.

[42] M. Muckin and S. C. Fitch, "A threat-driven approach to cyber security," *Lockheed Martin Corporation*, pp. 4–26, 2014.

[43] R. Stevens, D. Votipka, E. M. Redmiles, C. Ahern, P. Sweeney, and M. L. Mazurek, "The Battle for New York: A Case Study of Applied Digital Threat Modeling at the Enterprise Level," in *USENIX Security Symposium*, pp. 621–637, 2018.

[44] G. Digital, "Linux Security Advisories." https://linuxsecurity.com/advisories, 2024.

[45] Google, "Android Security Bulletins." https://source.android.com/docs/security/bulletin, 2024.

[46] J. Williams, "What You Need to Know About the SolarWinds Supply-Chain Attack." https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/, 2020.

[47] B. Schneier, "Chinese Supply-Chain Attack on Computer Systems." https://www.schneier.com/blog/archives/2021/02/chinese-supply-chain-attack-on-computer-systems.html, 2021.

[48] P. Vallina, A. Feal, J. Gamba, N. Vallina-Rodriguez, and A. F. Anta, "Tales from the porn," in *Proceedings of the Internet Measurement Conference*, pp. 245–258, ACM, 2019.

[49] N. M. Al-Fannah, W. Li, and C. J. Mitchell, "Beyond cookie monster amnesia," in *Developments in Language Theory* (M. Hoshi and S. Seki, eds.), vol. 11088 of *Lecture Notes in Computer Science*, pp. 481–501, Springer International Publishing, 2018.

[50] P. Eckersley, "How unique is your web browser?," in *Privacy Enhancing Technologies* (M. J. Atallah and N. J. Hopper, eds.), vol. 6205 of *Lecture Notes in Computer Science*, pp. 1–18, Springer Berlin Heidelberg, 2010.

[51] M. Ashouri, "A large-scale analysis of browser fingerprinting via chrome instrumentation," in *ICIMP 2019, The Fourteenth International Conference on Internet Monitoring and Protection*, IARIA, 2019.

[52] U. Iqbal, S. Englehardt, and Z. Shafiq, "Fingerprinting the fingerprinters:," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1143–1161, IEEE, 2021.

[53] T. Li, X. Zheng, K. Shen, and X. Han, "Fpflow: Detect and prevent browser fingerprinting with dynamic taint analysis," in *Cyber Security* (W. Lu, Y. Zhang, W. Wen, H. Yan, and C. Li, eds.), vol. 1506 of *Communications in Computer and Information Science*, pp. 51–67, Springer Nature Singapore, 2022.

[54] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "Fpdetective," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13* (A.-R. Sadeghi, V. Gligor, and M. Yung, eds.), pp. 1129–1140, ACM Press, 2013.

[55] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (G.-J. Ahn, M. Yung, and N. Li, eds.), pp. 674–689, ACM, 2014.

[56] A. Durey, P. Laperdrix, W. Rudametkin, and R. Rouvoy, "Fp-redemption," in *Detection of Intrusions and Malware, and Vulnerability Assessment* (L. Bilge, L. Cavallaro, G. Pellegrino, and N. Neves, eds.), vol. 12756 of *Lecture Notes in Computer Science*, pp. 237–257, Springer International Publishing, 2021.

[57] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster," in *2013 IEEE Symposium on Security and Privacy*, pp. 541–555, IEEE, 2013.

[58] SAP, "Project foxhound github repository," 2022.

[59] M. A. Bermejo-Agueda, P. Callejo, R. Cuevas, and Ángel Cuevas, "adf: A novel system for measuring web fingerprinting through ads," 2023.

[60] D. Developers, "Disconnect Me." Github, 2023.

[61] E. Developers, "EasyPrivacy." Github, 2023.