USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Bluetooth Security Testing with BlueToolkit: a Large-Scale Automotive Case Study

Vladyslav Zubkov, *ETH Zurich;* Tommaso Sacchetti and Daniele Antonioli, *EURECOM;* Martin Strohmeier, *armasuisse Science and Technology*

https://www.usenix.org/conference/woot25/presentation/zubkov

This paper is included in the Proceedings of the
19th USENIX WOOT Conference on Offensive Technologies.

August 11–12, 2025 • Seattle, WA, USA

ISBN 978-1-939133-50-2

Open access to the Proceedings of the
19th USENIX WOOT Conference on Offensive Technologies
is sponsored by USENIX.

Bluetooth Security Testing with BlueToolkit: a Large-Scale Automotive Case Study

Vladyslav Zubkov
ETH Zurich

Tommaso Sacchetti
EURECOM

Daniele Antonioli
EURECOM

Martin Strohmeier
armasuisse S+T

Abstract

Bluetooth is a wireless data-transfer protocol used by billions of heterogeneous devices, including vehicles, smartphones, and laptops. Bluetooth devices are affected by security issues, whose automated large-scale testing is challenging. In this work, we focus on Bluetooth Classic (BC) as there is no comprehensive open-source security testing framework. We fill this gap by designing and implementing BlueToolkit, a new BC security testing framework for automating recon, exploit testing, and report generation. Our tool tests the target over the air using a black-box approach, thus without prior hardware or software configuration knowledge. BlueToolkit tests 44 design and implementation exploits from six databases, including critical Machine-in-the-Middle (MITM), Remote Code Execution (RCE), and Denial of Service (DoS) ones. Moreover, it is extensible via a configuration system based on YAML files, allowing, among others, the integration of future exploits.

Despite the rise of Bluetooth usage in vehicles, its security in the automotive domain has been widely overlooked. We address this challenge using BlueToolkit to perform a comprehensive security assessment of real-world automotive In-Vehicle Infotainment (IVI) units. We evaluate 22 vehicles produced between 2016 and 2023 from 14 leading manufacturers. Each car underwent up to 44 tests, including design and implementation attacks, resulting in a total of 891 tests and 128 vulnerabilities. We also present four attacks we discovered during the experiments with the help of BlueToolkit. Our evaluation demonstrates that automotive Bluetooth security posture is inadequate and that BlueToolkit is effective in real-world use cases. We responsibly disclosed our findings to all affected vendors and open-sourced BlueToolkit.

1 Introduction

Bluetooth is a pervasive wireless communication technology, currently at version 6.0, defined in an open standard [15]. It has two main transport protocols: Bluetooth Classic (BC) and Bluetooth Low Energy (BLE). The former is optimized for high-throughput applications requiring reliable, high-speed data transfer, such as wireless speakers and file transfers. BLE is designed for ultra-low power devices, such as fitness trackers and smart home devices. This paper focuses on BC as we further explain in Section 3.1. Unless otherwise specified, we refer to Bluetooth Classic as *BC* or simply *Bluetooth*.

Previous research uncovered critical Bluetooth security design [4, 7–9, 20, 21, 35, 38, 50, 76, 83] and implementation [2, 31] flaws either manually, via formal verification, or through fuzzing. Some of these vulnerabilities have been fixed in the standard, but device vendors are not guaranteed to address them in their Bluetooth stack implementations. These vulnerabilities have a large-scale impact as they are exploitable on any device using Bluetooth security protocols, i.e., billions of devices.

Performing comprehensive Bluetooth security testing is challenging and time-consuming. Proof of Concepts (PoCs) available for known Bluetooth design and implementation vulnerabilities are scattered across different sources and are not interoperable. Available Bluetooth security tools focus on reverse-engineering [53] and fuzzing [31]. However, we lack a Bluetooth security testing framework supporting design and implementation exploits from different sources, enabling us to test them automatically and at scale.

We fill this gap by developing *BlueToolkit*, a new extensible framework for Over The Air (OTA) Bluetooth security testing. The tool performs a *recon* phase to gather information about a target configuration. Then, it runs a series of exploits against a target in a *black-box* manner. Finally, it produces a report that is machine- and human-readable.

BlueToolkit allows testing of existing design and implementation attacks from different sources, including BlueBorne [72] and BrakTooth [31]. BlueToolkit supports multiple exploits requiring distinct radio devices. More can be added using a high-level configuration system based on YAML files. We implemented BlueToolkit for BC and validated it with a large-scale automotive case study. However, our implementation can be used in any other Bluetooth domains, including

mobile, smart homes, and IoT. Moreover, our implementation can be easily extended to support BLE, which can be an added value in other case studies.

The lack of systematic, large-scale, and comprehensive evaluation of Bluetooth security in the automotive domain creates the perfect test case for BlueToolkit. A paper in 2022 [5] evaluated two protocol-level vulnerabilities on a limited sample of three cars. Another work from 2017 [17] presented a framework for automotive Bluetooth security testing based on attack trees and focusing on data extraction attacks. In [67], the authors found an implementation-specific bug on the Android Automotive Bluetooth stack, enabling several privacy threats.

We address this research gap by using BlueToolkit to evaluate 22 vehicles from 14 manufacturers produced between 2016 and 2023. This first large-scale evaluation of automotive Bluetooth provides many valuable real-world insights. It shows the poor security of Bluetooth implementations across vehicles from different vendors and the adoption delay of newer Bluetooth versions, with newly sold cars using versions over 10 years old.

While conducting our experiments on vehicles, we discovered four attacks. Three of them are implementation-specific issues enabling authentication bypass or information leakage. The fourth allows the compromise of user accounts protected by 2FA on the Internet. We discuss mitigations and recommendations on how to deal with our empirical findings. We responsibly disclosed the found vulnerabilities while conducting our experiments ethically (see Section 8.1 and Appendix A).

We summarize our contributions as follows:

- To address the lack of a comprehensive and automated Bluetooth security testing framework, we design and implement BlueToolkit. Our tool is black-box, low-cost, and extensible to handle existing and future Bluetooth design and implementation issues. It currently covers all known BC exploits with available PoCs.
- We evaluate BlueToolkit in a large-scale security evaluation of automotive Bluetooth. We test 22 cars from 14 popular manufacturers. We find 128 vulnerabilities, including 21 design and 46 implementation ones.
- We uncover and test four novel and impactful Bluetooth vulnerabilities: PBAP Extraction, Just Works Central Downgrade, No Numeric Comparison, and a User Account Takeover.
- BlueToolkit is open-source and available at `https://github.com/sgxgsx/BlueToolkit`.

2 Background

We present the necessary background information about Bluetooth, BLE, and their typical usage within the automotive domain.

2.1 Bluetooth

Bluetooth is a standard technology for short-range wireless communication maintained by the Bluetooth Special Interest Group (SIG). Its latest version 6.0 [15] was released in 2024. Bluetooth has two transports: BC, for high-throughput and connection-oriented use, such as audio streaming or data transfer, and BLE for low power and connectionless usage, such as notifications and contact tracing. BC and BLE provide standard profiles to address common wireless services, such as audio and file transfer, or notifications and messages management.

The Bluetooth stack is implemented by two components: the Host and the Controller. The Host manages high-level procedures, such as data exchange and establishing logical channels. The Controller takes care of the low-level functionalities, like managing the link layer and physical layers.

In Bluetooth communication, devices operate as a Central or Peripheral. The Central is the connection initiator, scanning for available devices, establishing connections, and starting security procedures. A Peripheral broadcasts advertising packets to announce its availability and accepts connection requests from a Central. Central and Peripheral roles are fixed in BLE, while BC allows devices to switch roles dynamically.

2.2 Bluetooth Security

Bluetooth security relies on two protocols: pairing and session establishment. Pairing establishes a long-term key called the Pairing Key (PK) and offers user-assisted authentication, known in the standard as an *association*. Two common authenticated associations are Numeric Comparison (NC) and Passkey Entry (PE) and require hardware devices with both input and output capabilities. There is also an unauthenticated association called Just Works (JW), which all devices support. Session establishment generates a Session Key (SK), which encrypts the current session.

For BC, the Controller manages pairing (known in the standard as Secure Simple Pairing (SSP)) and session establishment using the Link Management Protocol (LMP). For BLE, the Controller manages session establishment, and the Host manages pairing. Bluetooth security has two modes: Secure Connections (SC) and Legacy Secure Connections (LSC). SC relies on standard cryptographic primitives and mechanisms, like ECDH and AES-CCM, while LSC employs legacy ones, including E0 and SAFER+, for backward compatibility.

Bluetooth is affected by *design* and *implementation* vulnerabilities and related attacks. Design flaws typically target the protocol flaws in pairing or session establishment and are effective on every compliant Bluetooth device using them. In contrast, implementation issues may vary across devices from different vendors and include issues such as pairing bypass, or memory corruption vulnerabilities leading to Remote Code Execution (RCE) or Denial of Service (DoS).

2.3 Automotive Bluetooth

The automotive industry widely uses Bluetooth to provide wireless services in vehicles. Today, around 87% of all newly sold cars offer Bluetooth connectivity, and around two-thirds of cars currently on the road have it [34]. In 2001, the first Bluetooth hands-free car kits were introduced [14]. Currently, the majority of cars with Bluetooth support BC exclusively. Only a single car in our study supported simultaneous BLE and BC dual-mode.

The component that offers the most Bluetooth services in automotive is the In-Vehicle Infotainment (IVI), which typically supports *Message Access Profile (MAP)*, *Hands-Free Profile (HFP)*, *Phone Book Access Profile (PBAP)*, and *Advanced Audio Distribution Profile (A2DP)*. *MAP* allows exchanging messages between devices. It is tailored for automotive use cases, and it allows for the reading, listing, sending, and managing of SMS, MMS, and potentially instant messaging and emails. *HFP* enables hands-free functionalities such as handling phone calls or interacting with the voice assistant without manually interacting with the smartphone. *PBAP* allows syncing contacts from a smartphone, enabling features like displaying contact information or starting a call from the IVI. *A2DP* allows audio streaming and, in modern vehicles, it can be used with more advanced audio-related functionalities such as Advanced Audio Coding (AAC) or Qualcomm audio processing technology (aptX).

3 Motivation and Threat Model

Here, we motivate this work and introduce its threat model.

3.1 Motivation

There is a vast body of literature concerned with Bluetooth security. Some works uncovered *design* vulnerabilities and attacks on pairing [8, 20, 21, 38, 74, 83] and session establishment [2, 7, 9]. Others presented *implementation* flaws and related threats [61, 72]. The presented security issues often result in critical exploits allowing, among other things, to impersonate and MITM devices, get 0-click RCE, or DoS a connection. The exploits have a large-scale impact as they affect heterogeneous devices, including vehicle IVIs.

Despite the many known design and implementation security vulnerabilities, there is no comprehensive Bluetooth security testing tool. Prior research presented fuzzers focusing on implementation-specific vulnerabilities [1, 31, 37, 63, 64, 70]. Internalblue [53] and BrakTooth [31] implemented some protocol-level attacks like KNOB [9], NiNo [38], and FCIC [12]. However, many existing protocol-level attacks are provided with PoC code that is hardware-dependent and often difficult to test and reuse. Some examples include the toolkits provided by the authors of KNOB [9], BIAS [7], method confusion [83], and BLUFFS [4]. Hence, we still lack a tool capable of quickly and systematically testing Bluetooth exploits.

Several security testing tools exist for BLE [32, 42, 78]. However, they cannot be used for BC as the two have different link and physical layers, security mechanisms, and primitives. Notably, the latter is more challenging to test as its security mechanisms are implemented in the Controller, and there is no open-source Controller, while there are multiple open-source BLE implementations. Hence, we need a comprehensive BC testing tool.

The research community's limited understanding of automotive Bluetooth security is the second issue motivating this work. Vehicles, including cars, trucks, and motorbikes, rely on BC for many sensitive services, like contact sharing and hands-free calls (as introduced in Section 2.3). However, only one recent paper [5] provides experimental evidence of insecurities in automotive Bluetooth. The paper offers partial insights into the KNOB and BIAS attacks, which were manually tested against four IVI found in cars manufactured by KIA, Toyota, Suzuki, and Skoda. We still lack a systematic case study on automotive Bluetooth security and an effective tool for conducting such a case study.

This work addresses the two mentioned gaps by presenting a new and effective BC security testing framework and a systematic automotive Bluetooth security evaluation. Our work validates the tool and raises new concerns about automotive Bluetooth security.

3.2 Threat Model

System Model Our system model includes two BC devices that want to communicate securely, such as a driver's smartphone and their car IVI. The devices are paired and can establish secure sessions. The user accepted the prompts asking for data to be shared among the devices. Hence, the IVI stores the user's contacts, call history, emails, and SMS, and accepts hands-free commands. The user does not necessarily own both devices. For example, users can rent, borrow, or share a car and pair their smartphone with its IVI.

Attacker Model We consider a proximity attacker within Bluetooth range of the target device. They can sniff, inject, modify, drop, record, replay, reflect, and redirect messages to and from the target device (i.e., a Dolev-Yao [24] attacker). The attacker knows public information about the target, such as their Bluetooth address and name. In some scenarios, the attacker may have limited physical access to a target device. For example, an attacker can try to tamper with the IVI of a rented vehicle.

The attacker targets BC design and implementation flaws. They aim at establishing MITM position between devices or impersonating a device by exploiting design flaws, like KNOB. Moreover, they want to bypass BC security protocols, DoS existing sessions, or achieve RCE by exploiting

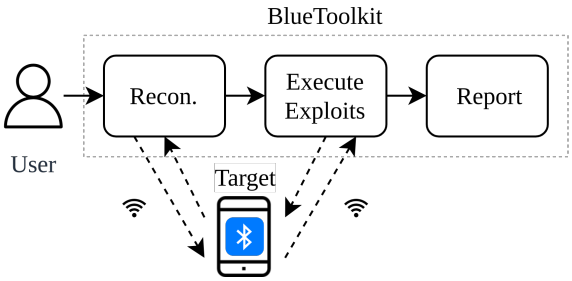

Figure 1: BlueToolkit overview. A user performs black-box recon and exploitation tests on a target OTA.

implementation issues on the target Bluetooth stack.

4 Design

We present the design of **BlueToolkit**, a framework for effective Bluetooth security testing. We set its design requirements, describe its architecture, and compare it with state-of-the-art BC security tools.

4.1 Requirements and Architecture

We set four design requirements for BlueToolkit: *black-box*, *high coverage*, *extensible*, and *usable*. Next, we describe how we address them in our architecture.

4.1.1 Black-box

We propose a black-box design for BlueToolkit as shown in Figure 1. The tool tests a target OTA and does not require information about the target software, which is usually closed-source and device-specific, nor access to the target hardware, like debugging ports. This black-box approach provides scalability, enabling testing of any BC device in a semi-automated fashion. The tool needs only the target's Bluetooth address to work; then, it performs the following three phases.

Recon The recon(naissance) phase collects operational and security-relevant parameters for the upcoming exploitation phase. It acquires the target's Bluetooth capabilities, e.g., connectable, pairable, and discoverable states and features, including Bluetooth versions, supported transport layers, and profiles. Moreover, it discovers the target's security configuration, including support for SSP, SC/LSC, and association.

Execute Exploits The exploitation phase runs a series of tests based on the recon information and the user preferences. The tool can run one, multiple, or all tests from a catalog. Currently, we support *five test classes* covering BC design and implementation issues: MITM, RCE, authentication bypass (AB), information leakage (IL), and DoS. Each test has a termination condition to check success or failure. For example, when testing a DoS attack, the tool waits for 50 seconds and then tries to check if the IVI is connectable and pairable to check whether the exploit is successful. A testing session can be stopped and resumed using a checkpoint to save time.

Report Once the recon and exploitation phases are completed, BlueToolkit generates a report. The report contains the recon information and the list of executed exploit tests and their outcomes. The report is a JSON file and is readable by humans and machines. Figure 3 shows a report excerpt.

4.1.2 High Coverage

BlueToolkit has high coverage as it covers Bluetooth design and implementation issues. This is regardless of the target's hardware and software configuration, including the supported Bluetooth version, security mode, and association. Typical black-box security tools like fuzzers focus on discovering new implementation bugs, while PoC repositories test a specific design or implementation flaw. Instead, our tool does active recon and tests various design and implementation exploits.

4.1.3 Extensible

BlueToolkit is extensible thanks to the abstraction layer it uses to represent recon tests, exploit tests, and testing hardware. The tool uses configuration files and provides a high-level API to integrate exploits from known repositories, like BrakTooth [31], and add new ones, like those discussed in Section 6.

BlueToolkit supports multiple testing hardware peripherals such as an ESP32 dongle (used for BrakTooth) or a Nexus 5 device (used for KNOB). This is helpful because exploits PoCs often require specific hardware, and porting them to a different platform would require additional reverse-engineering work or may not be feasible.

The configuration module allows users to set relevant testing parameters such as exploit name, author, type, target Bluetooth versions, and commands to run. It enables testing an exploit or its variations without requiring deep technical expertise. For example, a user can test KNOB downgrades with different entropy values without knowing how to patch LMP in a BC firmware.

The modular and configurable nature of BlueToolkit enables the straightforward extensibility towards BLE in the future. The extension requires collecting and developing recon and exploit tests for BLE, introducing additional hardware support, and updating the configuration engine accordingly.

4.1.4 Usable

BlueToolkit is usable as it abstracts complex details using a high-level API. As a result, using and extending the tool

Table 1: Comparison between BlueToolkit and RE and fuzzing BC security tools and PoC exploits.

Name	Type	How	Tests
BlueToolkit (new)	SecTest	OTA	Rec, Des, Impl
InternalBlue [53]	RevEng	OTA	Des, Imp
Frankenstein [70]	Fuzz	Rehost	Imp
BlueSpec [63]	Fuzz	OTA	Imp
BrakTooth [31]	Fuzz	OTA	Imp
L2Fuzz [64]	Fuzz	OTA	Imp
VirtFuzz [37]	Fuzz	VirtIO	Imp
BloomFuzz [1]	Fuzz	OTA	Imp
KNOB [9]	PoC	OTA	Des
BIAS [7]	PoC	OTA	Des
MethConf [83]	PoC	OTA	Des
BLUFFS [4]	PoC	OTA	Des
BlueBorne [72]	PoC	OTA	Imp
BleedingBit [73]	PoC	OTA	Imp
BleedingTooth [61]	PoC	OTA	Imp

requires minimal knowledge and engineering effort compared to deploying the individual tests (similar to Metasploit [66]). For example, the user can configure a test via a file or write a new one by reusing an existing one.

BlueToolkit is also easy to install and run. It relies on open-source software and low-cost commodity hardware. It uses Linux's Bluetooth subsystem (Bluez and kernel driver) and cheap and configurable Bluetooth controllers like ESP32 or Raspberry Pi chips. It includes a build script to download and install the necessary code and dependencies. It provides a scriptable Command Line Interface (CLI) to run the tests and automatically generate a report. It also enables the saving and restoration of testing sessions.

4.2 Tools Comparison

Table 1 shows how BlueToolkit advances the state of the art of Bluetooth security testing tools. Unlike the other tools, it can perform recon, design, and implementation vulnerability testing and reporting. It allows the embedding of known design and implementation exploits from popular fuzzers or PoC repositories. However, it can be extended with new recon and exploitation tests, including the ones we present in Section 6.

5 Implementation

We implement BlueToolkit using open-source software and low-cost hardware. We tested our implementation on Ubuntu, but it should run on any Debian-based distribution. Next, we describe how we implemented the BlueToolkit's configuration module and recon/exploit test catalog.

```yaml
name: "au_rand_flooding"
author: "Braktooth team"
type: "DoS"
mass_testing: true
bt_version_min: 2.0
bt_version_max: 5.4
hardware: "esp32"
command: "./bin/bt_exploiter --host-port=/dev/ttyUSB1
 ↪ --exploit=au_rand_flooding --random_bdaddress"
parameters:
  - name: "--target"
    type: "str"
    name_required: true
    help: "Target MAC address"
    required: true
    is_target_param: true
    parameter_connector: "="
log_pull:
  in_command: false
  from_directory: true
  relative_directory: true
  pull_directory:
   ↪ "modules/tools/braktooth/wdissector/logs/Bluetooth"
directory:
  change: true
  directory: "modules/tools/braktooth/wdissector"
```

Listing 1: `braktooth_au_rand_flooding.yaml`.

5.1 Configuration Module

Our implementation provides a configuration module to describe and set recon tests, exploitation tests, and testing hardware. The module uses YAML, a well-supported, readable, and open markup language. BlueToolkit already provides a collection of YAML files for recon and exploitation tests and for required testing hardware. Adding a new test is simple, as someone can start from an existing YAML file.

The YAML design and implementation exploit files follow a naming scheme where the prefix indicates the test source, followed by the name of the test. For example, `braktooth_knob.yaml` allows to run the KNOB attack from the BrakTooth fuzzer repository. We reuse tests from the `bleedingtooth`,`blueborne`, `braktooth`, and `internalblue` toolkits. The `reconnaissance` and `custom` files are based on new tests we developed as part of this work.

Listings 1 shows `braktooth_au_rand_flooding.yaml`. The file contains the exploit name, author, and type, and allows configuring and running the AURAND flooding DoS exploit from BrakTooth. The file enables setting the target Bluetooth versions (between `2.0` and `5.4`), and the tester hardware (`esp32`). The configuration module parses the YAML file, pulls the necessary code, and runs the test via the `command` field. The command can use arbitrary programs, e.g., `python` or `bt_exploiter`.

The module supports *three hardware profiles*: ESP32,

```yaml
name: "esp32"
description: "ESP 32 board for braktooth exploits"
setup_verification: ""
needs_setup_verification: true
working_directory: "/braktooth/wdissector"
bt_version_min: 4.0
bt_version_max: 5.3
```

Listing 2: `esp32.yaml`.

Nexus 5, and Linux (default). It allows reusing hardware-specific tests, such as InternalBlue ones, without re-implementing them for a new platform. For example, Listing 2 shows the hardware YAML for the ESP32 profile. The file contains information about a specific hardware device and points to a `setup_verification` script that allows the system to verify hardware availability. The hardware check is implemented in `setupverification.py`.

Adding extra hardware profiles is straightforward. One has to write a YAML file with the relevant information and provide the system with a way to check if the device is connected and functioning by implementing a static method in the `setupverification.py`. If the device requires any extra dependencies, it must also provide an installation script that shall be located in the `installation` folder.

5.2 Tests

Our BlueToolkit implementation supports *44 tests*: *7 recon* tests, *6 design exploit* tests, and *31 implementation exploit* tests. Next, we describe each test in more detail.

5.2.1 Recon

The reconnaissance tests (R1–R7), listed in Table 2, use Bluetooth discovery (inquiry) packets or interact with the target to obtain preliminary knowledge about a device. They include generic recon tests such as checking if a target supports SC, BLE, or SSP, but also *automotive-specific* ones to check if an IVI is rebootable or accepts connection requests without opening the settings.

The recon tests allow a more efficient and specific exploitation testing phase. For example, knowing that SC is not supported allows skipping tests for SC related vulnerabilities. The R1 test (IVI is not rebootable) is the only one requiring manual intervention. The user must physically interact with the car after connecting with the IVI system via Bluetooth to test under which conditions the vehicle disconnects. For example, if shutting down the vehicle shuts down also, the IVI or not.

Table 2: Seven recon tests implemented in BlueToolkit.

ID	Name
R1	IVI is not rebootable
R2	Not only IVI can initiate a conn.
R3	SC not supported
R4	Always pairable
R5	E0 used
R6	SSP not supported
R7	BLE available

Table 3: Six design tests implemented in BlueToolkit. D6 is a new information leakage (IL) attack.

ID	Name	Type
D1	Method Confusion	MitM
D2	NiNo	MitM
D3	Invalid Curve Attack	MitM
D4	KNOB	MitM
D5	Legacy Pairing	MitM
D6	**PBAP Extraction (new)**	IL

5.2.2 Exploit Selection

We reviewed the state of the art and first collected all known design and implementation exploits for Bluetooth (Classic and Low Energy). Our sources include research papers, security conferences, and standalone code repositories across the past 10 years. This yielded 63 vulnerabilities overall. We aggregated the information in a table and enriched it with status, name, type, affected Bluetooth versions, the year it was released, Common Vulnerabilities and Exposures (CVE) number, and CVSS score.

As in this work we focus on available PoCs, we narrowed the candidates down to 43 concrete exploits, excluding those requiring unavailable hardware or without an implementation. Whether anything changes in the future, an exploit can easily be added to BlueToolkit. From the collected exploits, we finally implemented *6 design* and *31 implementation exploits* as tests in BlueToolkit. Since we cannot fit all the exploits into the paper, we point to the full collection, which is available in BlueToolkit repository.

5.2.3 Design Exploits

Table 3 describes the 6 design exploits (D1–D7) supported by BlueToolkit. They include method confusion, NiNo, invalid curve attacks, KNOB, and legacy pairing. The list includes *PBAP Extraction (D6)*, a new Information Leakage (IL) attack we discovered. The attack is presented in Section 6.1.

Table 4: BlueToolkit has thirty-one implementation tests (new ones are in bold). I1–I8 are Remote Code Execution (RCE), including zero-click ones. I9 and I11 are authentication bypasses (AB), including two new attacks (I10 and I11). I12-I31 are BrakTooth (Brkt) DoS exploits.

ID	Name	Type
I1	BleedingTooth BadChoice	RCE
I2	BleedingTooth BadKarma	RCE
I3	BleedingTooth BadVibes	RCE
I4	CVE-2018-19860	RCE
I5	BlueBorne CVE-2017-1000250	RCE
I6	BlueBorne CVE-2017-0785	RCE
I7	BlueBorne CVE-2017-1000251	RCE
I8	Braktooth Inv. Feature P. Exec.	RCE
I9	CVE-2023-45866	AB
I10	**No Numeric Comparison (new)**	AB
I11	**JW Central Downgrade (new)**	AB
I12	Brkt au rand flooding	DoS
I13	Brkt feature response flooding	DoS
I14	Brkt feature req ping pong	DoS
I15	Brkt paging scan disable	DoS
I16	Brkt wrong encapsulated payload	DoS
I17	Brkt duplicated iocap	DoS
I18	Brkt lmp overflow 2dh1	DoS
I19	Brkt lmp overflow dm1	DoS
I20	Brkt lmp auto rate overflow	DoS
I21	Brkt sdp oversized element size	DoS
I22	Brkt lmp invalid transport	DoS
I23	Brkt lmp max slot overflow	DoS
I24	Brkt repeated host connection	DoS
I25	Brkt duplicated encapsulated payload	DoS
I26	Brkt sdp unkown element type	DoS
I27	Brkt truncated sco link request	DoS
I28	Brkt truncated lmp accepted	DoS
I29	Brkt invalid setup complete	DoS
I30	Brkt invalid max slot	DoS
I31	Brkt invalid timing accuracy	DoS

Our tests check if a target is vulnerable to a MITM attack without the need to actively MITM a connection. This type of test is common for Bluetooth, where implementing a full MITM is not straightforward because of frequency hopping and synchronization issues at the link layer and unavailable open source BC firmware or software-defined radio (SDR) stacks.

5.2.4 Implementation Exploits

Table 4 lists the 31 implementation exploits (I1–I31) implemented by BlueToolkit. We support eight RCE tests. Among these tests, we reuse popular and critical RCE samples from the literature like BlueBorne and BleedingTooth. We include three Authentication Bypass tests checking if the target uses mis-implemented BC security protocols. *No Numeric Comparison (I10)* and *JW Central Downgrade (I11)* are new and described in detail in Sections 6.3 and 6.2. The third one (I9) is an issue affecting the Human Interface Device (HID) profile of BlueZ.

We also support twenty DoS tests from BrakTooth. A DoS crashes the target temporarily or permanently. A permanent crash state requires a device reboot, while a temporary crash causes the device's Bluetooth service to restart without powering it off and on. Moreover, a crash can be partial or full. A partial crash can result in the device not being discoverable but connectable and pairable, in which case existing connections remain active. Alternatively, it can result in a device that is neither discoverable nor connectable or pairable for a specific time, which depends on the implementation.

6 New Attacks

Next, we describe *four new attacks* discovered during our research—the attacks sometimes chain (novel) BC design and implementation flaws. The work on BlueToolkit was instrumental in discovering the attacks, allowing us to scale the tests of newly observed unexpected behaviors.

6.1 PBAP Extraction

The PBAP Extraction attack enables an attacker who can physically interact with an IVI to leak the contacts of paired Bluetooth smartphones via re-pairing. It is a design exploit as it abuses PBAP, a Bluetooth profile, to sync phone contacts between paired BC devices. In the automotive context, the smartphone is the PBAP server that sends its contacts to the IVI acting as the PBAP client. The attack is caused by improper access control management on the IVI between re-pairing events. Hence, we classify it as a design exploit and label it D6 in Table 3. Being a design issue, the attack can work against any PBAP client (other than a vehicle). However, the physical access requirement is not realistic when applied to other domains, which makes it unlikely to be exploited.

The PBAP extraction attack relies on re-pairing to a victim IVI while impersonating a trusted smartphone that already shared its contact with the IVI. The attacker connects to the IVI using the Bluetooth address of the victim's smartphone and pairs with the IVI using whatever association is required. Then, they disable contact synchronization to avoid overriding the victim's contacts stored on the IVI. As a result, the IVI thinks it was repaired with the victim's smartphone and shows the victim's contacts on the screen, and the attacker can exfiltrate them.

The PBAP extraction attack is particularly effective on rented and shared cars. These cars usually contain phone contacts from past drivers, and the attacker has sufficient time to extract the contacts without being bothered or noticed. Moreover, the attack can be performed even when the engine is

off since some vehicles' IVI systems work regardless of the engine state. It is effective regardless of the association mechanism supported by the IVI, including NC and PE. Its main limitation is scalability because it requires manual interaction with the IVI to extract the contacts.

6.2 Just Works Central Downgrade

The JW Central Downgrade is an attack chaining an implementation issue we found and NiNo, a known protocol-level attack [38]. The chain establishes a persistent MITM position between the victim's IVI and smartphone while bypassing pairing authentication. The attack is labeled as I11 in Table 4.

The implementation flaw relates to IVI vendors improperly enforcing authentication while pairing. We discovered that some IVIs reject unauthenticated pairing from a Central but accept it from a Peripheral. Hence, whenever a vulnerable IVI starts the pairing process (Central), a MitM attacker can downgrade pairing to JW, which is not authenticated, and break the confidentiality, authenticity, and integrity of pairing and subsequent sessions.

6.3 No Numeric Comparison

The No Numeric Comparison attack enables the establishment of a MITM position between an IVI and a smartphone even if they use NC authentication, which is supposed to protect against MITM attacks. It is labeled as I10 in Table 4.

The attack is enabled by an implementation-level vulnerability: certain IVIs do not implement the NC dialog as part of their user interface and automatically accept NC in the background. As a result, an attacker can get a MITM position between the devices without user interaction (i.e., 0-click attack). Other IVIs show a Yes/No confirmation dialog without the numeric code, preventing the user from authenticating pairing. In these cases, the attacker still gets a MITM position, but requires the victim to confirm the connection.

6.4 User Account Takeover

The User Account Takeover attack chains a MitM attack (like No NC or JW Central Downgrade) with a two-factor authentication (2FA) account attack. The intercepted account data is a 2FA SMS or email via MAP or an account confirmation phone call via HFP. The attack shows that a Bluetooth attack can reach a broader scope than expected, i.e., compromise an Internet account via a short-range wireless technology. No prior automotive Bluetooth work demonstrated this capability, but it was shown in other contexts, such as mobile [6].

The attack resets and takes over a victim's online account protected with SMS or call-based 2FA. While MAP theoretically supports email and instant messaging, we could only find implementations with SMS support. Hence, we focus on the phone number as a 2FA means. It requires the following five steps:

1. Establish a MITM position between the victim hardware.
2. Retrieve the victim's phone number. This can be done by accessing SMS metadata, sending an SMS to itself via MAP, or calling itself via HFP.
3. Use an OSINT tool, like predictasearch [49], to find victim accounts linked to their phone number.
4. Trigger account reset (or a login flow if the attacker knows the account password) for those with SMS 2FA.
5. Intercept the 2FA code sent via MAP (SMS), then use these codes to take over the account.

The attack is effective on Android and iOS smartphones. Old Android versions (pre-2019) do not require a prompt confirmation to enable MAP or HFP. In contrast, new Android versions require users to manually confirm each re-pairing. On iOS, the user needs to grant permissions manually via settings. However, the permissions are not reset when a device is re-paired, allowing an attacker spoofing a legitimate device to automatically be granted the same permissions.

We highlight that this multi-stage attack is possible not because of some implementation-level flaws but because the standard allows those actions to be performed and does not specify how to handle MAP and HPF settings and permissions. Hence, we classify it as a design-level issue related to these two Bluetooth profiles.

We also clarify that BlueToolkit does not provide a dedicated test for this attack as it goes beyond its capabilities, involving multiple steps and complex browser interaction. However, we developed a specific tool to carry out our User Account Takeover attack, and it is available at https://anonymous.4open.science/r/mapAccountHijack-8B53/.

Table 5: Top ten vehicle manufacturers by sales [56] and revenue [29] in 2023. The 14 manufacturers we tested are in italics.

Rank	Sales	Revenue
1	*Toyota*	*Volkswagen*
2	*Volkswagen*	*Toyota*
3	*Hyundai-Kia*	*Stellantis*
4	*RNMA*	Ford
5	*Stellantis*	*General Motors*
6	*General Motors*	*Mercedes-Benz*
7	*Honda*	*BMW*
8	Ford	*Honda*
9	Suzuki	*Hyundai*
10	BYD	SAIC

7 Evaluation

We evaluate BlueToolkit on 22 vehicles, covering a large part of the current automotive market. Our evaluation shows that BlueToolkit is systematic, scalable, and effective. The results highlight a concerning trend: car manufacturers exhibit poor Bluetooth security practices. On average, brand-new vehicles run Bluetooth versions that are over seven years outdated, and security updates are rare. As a result, vehicle vulnerabilities will remain an issue for the foreseeable future. Next, we describe our evaluation setup and results.

7.1 Experimental Setup

For a broad real-world evaluation, we need access to different types of vehicles from popular vendors and manufacturers. Access to cars was provided by a governmental department, which gave us access to their collection of *22 cars* from *14 popular manufacturers*. While some vehicles were available only in the early development phase of BlueToolkit when not all exploits were implemented, we decided to include them in the evaluation and mark them as partially tested. All tested cars supported BC, and only a single car supported simultaneous operation of BLE and BC, illustrating the motivation to focus on BC for this work.

The car market is highly dynamic, with significant shifts seen in global market shares in recent years due to the rise of electric vehicles in many countries. We selected the test vehicles for our Bluetooth security studies based on the most popular manufacturer in terms of sales and revenues. Table 5 shows the rankings for these two popularity metrics for 2023. Among the listed manufacturers, we could not test Ford, Suzuki, BYD, and SAIC. Except for Ford, those are the least popular on the list.

For each tested car, we placed a laptop with BlueToolkit, an ESP32, and a Nexus 5 inside the cabin. Non-EV cars that required the engine to be turned on to access the IVI were tested outdoors. The typical test time for a vehicle was around two hours, but it strongly depended on the DoS attack results. The time increased to three hours, where it was necessary to turn the car off and physically remove the car key from radio range to perform a reset after an IVI crash. Sometimes, the key must be outside the range for up to 20 minutes.

Figure 2 shows the test setup within the Renault Zoe. We can observe that BlueToolkit does not require access to the car's debug ports as it is a black-box testing tool. The setup is also portable and easy to reproduce.

7.2 Results

Now we discuss the BlueToolkit results of all 22 cars. We define a successful recon or exploitation test as a *positive test*. Table 6 contains the complete evaluation results. There is only one car, the Skoda Enyaq (2022), for which testing resulted in no positives. However, we could not run all 44 tests, but only 30 of them, because they were not implemented at the time of testing (e.g., NiNo, Method Confusion, and Blueborne).

Figure 2: Testing a Renault Zoe (2021) with BlueToolkit.

All other cars had multiple flaws in their Bluetooth systems, and there was significant variation across different models and manufacturers (we do not make claims about representativeness). Performing reconnaissance is possible against most vehicles, with 61 positive tests. We found 21 design vulnerabilities and 46 implementation vulnerabilities. Two cars (Renault Megane 2021 and Toyota Corolla 2023) accounted for eight and nine implementation vulnerabilities each (respectively 17% and 20% of all). This hints at a possible correlation between some types of vulnerabilities.

We can also observe that the Bluetooth version is not an indicator of security posture. The newest tested version (v5.2) has only two positive tests, while one car with v5.1 has nine of them. The older Bluetooth v4.2, released in 2014, is implemented by 10 cars manufactured between 2020 and 2023, with the number of positive tests ranging from zero to 13. We now discuss in detail the different manufacturers tested.

Skoda Skoda has the lowest number of positive tests (2.75 on average). However, we performed the full 44 available tests only on one out of four models. Most positive results are reconnaissance ones, with few in design and implementation.

Audi The two tested Audi cars (A5 and e-tron, both 2020) show a moderate number of positive tests (five), and both use Bluetooth v4.2. Most issues are reconnaissance, with a minority of design and implementation vulnerabilities.

Table 6: Automotive Bluetooth evaluation results. We test 22 vehicles from 14 manufacturers: four from Skoda (Sx), two from Audi (Ax), three from VW (Vx), three from Renault (Rx), two from BMW (Bx), and eight from other popular vendors (Mx). Vehicles' blocks are sorted chronologically. The Positives (Pos.) column counts the successful tests (Positives = Recon + Design + Impl.). See Tables 2 - 4 for the full names and categories of the tested exploits.

ID	Vehicle	Group	Year	Type	Tests	Pos.	Recon	Design	Impl.	BTv	BT Man
S1	Skoda Octavia	VW	2015	Sedan	8	6	3	2	1	3.0	Marvell
S2	Skoda Octavia	VW	2019	Sedan	29	3	2	1	0	3.0	Marvell
S3	Skoda Octavia	VW	2022	Sedan	44	2	2	0	0	4.2	Broadcom
S4	Skoda Enyaq	VW	2023	EV	30	0	0	0	0	4.2	Broadcom
A1	Audi e-tron	VW	2020	EV	44	4	3	0	1	4.2	Broadcom
A2	Audi A5	VW	2020	Sedan	44	6	3	1	2	4.2	Broadcom
V1	VW T6.1	VW	2022	Transporter	44	9	4	4	1	4.1	Toshiba
V2	VW ID.3 Pro	VW	2022	EV	44	3	2	0	1	4.2	Broadcom
V3	VW Caddy	VW	2023	Panel Van	44	3	3	0	0	4.2	Cypress
R1	Renault Megane	RNMA	2016	Hatchback	44	10	5	4	1	2.1	Qualcomm
R2	Renault Megane	RNMA	2021	Hatchback	44	13	3	1	9	4.2	Marvell
R3	Renault Zoe	RNMA	2021	EV	44	8	3	2	3	4.2	Marvell
B1	BMW X2	BMW	2021	SUV	44	10	4	2	4	4.0	Texas Ins.
B2	Mini Cooper	BMW	2022	3-door Hatch	44	7	3	1	3	5.0	Texas Ins.
M1	Chevrolet Corvette	GM	2018	Sports Car	44	7	4	1	2	3.0	Qualcomm
M2	Opel Astra	Stellantis	2019	Compact	44	6	2	0	4	4.1	Cypress
M3	Honda e	Honda	2020	EV	44	8	4	2	2	5.0	Qualcomm
M4	Sprinter 316CDI	MB	2021	Transporter	44	4	3	0	1	4.2	Marvell
M5	Hyundai Kona	Hyundai	2022	SUV	44	5	4	0	1	5.0	Broadcom
M6	Polestar 2	Geely	2022	EV	32	3	2	0	1	4.2	Qualcomm
M7	Toyota Corolla	Toyota	2023	Hybrid	44	9	1	0	8	5.1	Marvell
M8	Tesla Model Y	Tesla	2023	EV	44	2	1	0	1	5.2	Qualcomm
Tot.	**22 Cars, 14 Manuf**		**15–23**		**891**	**128**	**61**	**21**	**46**	**2.1–5.2**	**6**

Volkswagen The three Volkswagen cars show a similar picture to the Audi ones, potentially because both marques belong to the same larger manufacturing group. In total, we found two design and four implementation issues, with the VW T6.1 being the most vulnerable. Among these flaws, we can count KNOB, NiNo, and the new PBAP Extraction.

Renault The three Renault cars tested showed many positive tests across all three categories. Notably, the newer version of the Renault Megane from 2021 showed more issues than the 2016 one, despite a newer Bluetooth version (v4.2 vs. v2.1, the oldest in our sample). This is largely due to a record number of implementation vulnerabilities (nine) in addition to the PBAP Extraction one.

Misc We tested eight more manufacturers with one car each. We picked different vehicles, ranging from a small electric car (Honda E) to a sports car (Chevrolet Corvette) to SUVs (Hyundai Kona) and transporters (Mercedes-Benz Sprinter), covering the full range of modern car categories. The eight tested cars from miscellaneous manufacturers (2018-2023) exhibit between two (Tesla Model Y, 2023) and nine (Toyota Corolla, 2023) positive tests. Most of them were related to reconnaissance. However, the Toyota Corolla and the Opel Astra (2019) have eight and four implementation vulnerabilities. The Honda E results indicate two design flaws, i.e., KNOB and NiNo. Figure 3 shows an excerpt of the JSON report generated for the Renault Zoe. The report contains the performed tests' respective output lists and provides additional data about the target, e.g., Bluetooth version (4.2) and chip manufacturer (Marvell).

7.3 Statistics

We present interesting statistics obtained from our evaluation. First, we find that the delay between the release of a new Bluetooth version and its integration into a car sold on the market is 7.14 years on average, with a standard deviation of 1.96 years. The minimum in our sample was a 3-year delay

```
            {
                "index": 43,
                "name": "steal_contacts",
                "code": 2,
                "data": "Vulnerable"
            },
            {
                "index": 44,
                "name": "custom_numeric_wrong_implementation",
                "code": 2,
                "data": "Vulnerable"
            },
            {
                "index": 45,
                "name": "only_vehicle_can_connect",
                "code": 2,
                "data": 0
            },
            {
                "index": 46,
                "name": "fast_reboot",
                "code": 2,
                "data": "fast reboot is not available"
            }
        ],
        "bt_version": 4.2,
        "manufacturer": "Marvell Technology Group Ltd.",
        "mac_address": [redacted],
        "vehicle_name": "Renault ZOE",
        "vehicle manufacturer": "Renault",
        "parent_company": "Renault-Nissan-Mitsubishi Alliance",
        "year": 2021
}
```


Figure 3: Excerpt of the report generated by BlueToolkit for a Renault Zoe in JSON format.

for the Tesla Model Y, and the maximum was 11 years for the BMW X2. Figure 4 shows an illustration of this delay.

Figure 5 presents the average number of positive findings of BlueToolkit based on the Bluetooth version of the tested cars. As we can see, there is no clear indication that newer Bluetooth version implementations are more secure against the tested exploits. While the sample size for each version is small, our study suggests that implementation vulnerabilities are independent of the Bluetooth version. For reconnaissance and design tests, there is a potential trend towards fewer (known) issues exploitable in newer versions, i.e., 5.1 and 5.2.

Figure 6 presents the number of exploits per category and manufacturer of the Bluetooth chip. Overall, we had six different Bluetooth chip manufacturers in our sample (shown in Table 6). Broadcom (6), Marvell Technology (6), and Qualcomm (5) were the most common, making up 17 of the 22 chipsets. We can see some differences between the manufacturers, with Broadcom having the fewest vulnerabilities on average. Notably, Cypress had no identified design vulnerabilities, but we cannot conclude anything from this fact due to the small sample size.

8 Discussion

Here, we discuss our responsible disclosure process, attack fixes, and recommendations to vehicle manufacturers.

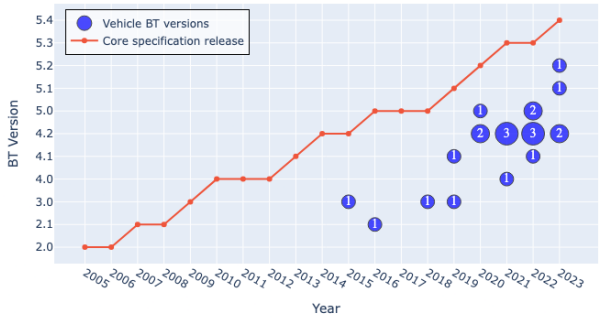

Figure 4: Bluetooth (BT) versions release date compared with the version implemented in vehicles and the year they were manufactured. The numbers indicate cars from the same year and with the same Bluetooth version.

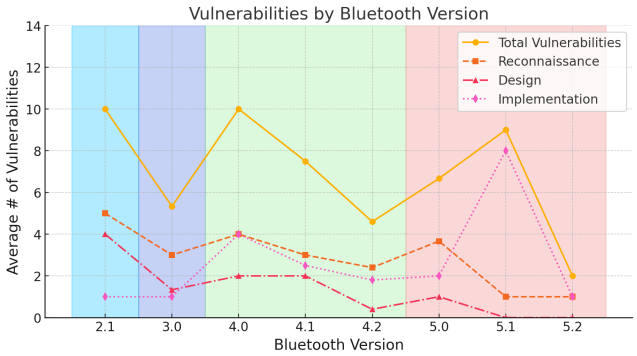

Figure 5: Average number of vulnerabilities found per Bluetooth version, major versions shaded in different colors.

8.1 Responsible Disclosure

We followed a standard responsible disclosure process for all design and implementation vulnerabilities. We contacted the manufacturers directly or via intermediaries such as HackerOne, BugCrowd, and two national CERTs. We provided all manufacturers with detailed information on the vulnerabilities and remediation steps, leaving sufficient time (at least 90 days) before public disclosure. As of the time of writing, 29 of the 46 implementation vulnerabilities have been acknowledged by the manufacturer, and for 15, there is work on a fix in new hardware/firmware versions. For the 21 design vulnerabilities, 12 have been acknowledged, with eight being worked on to be fixed for new versions.

Beyond this, we contacted the Bluetooth SIG with details on the novel User Account Takeover, which was also directly disclosed to Apple and Google as leading smartphone vendors. While Google is working on a fix and hardening, Apple has not recognized the issues in our report. However, we note that not resetting a device's permission after a new pairing is a relevant problem, and it should be addressed since it allows

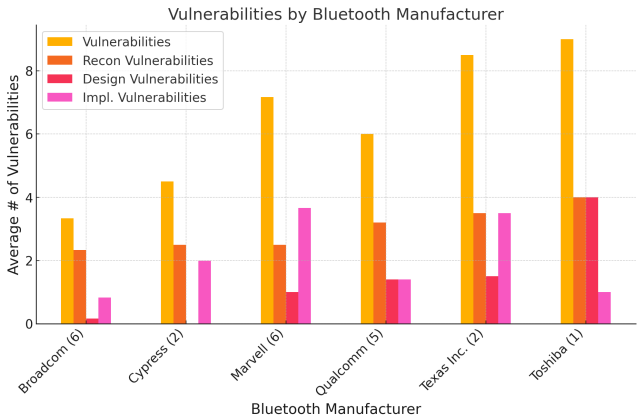

Figure 6: Average number of vulnerabilities per Bluetooth chip manufacturer (number tested in parentheses).

us to skip a user interaction step in our attack scenario.

8.2 Fixes and Recommendations

PBAP Extraction To fix the PBAP extraction attack, we suggest two complementary fixes: i) the IVI should encrypt a device's contacts with the PK, thus preventing an attacker from accessing them with a different PK. ii) The IVI should delete a device's contacts upon a successful repair, thus preventing an attacker from accessing them.

Just Works Central Downgrade To fix the Just Works Central Downgrade, we recommend that IVI vendors check if their implementation properly enforces authenticated pairing regardless of the device's Bluetooth role. During our evaluation, we tested several IVIs that implement such a measure and are not vulnerable to the attack.

No Numeric Comparison To fix the No NC attack, the vulnerable IVI vendors should adequately implement NC following the indication in the Bluetooth standard, i.e., always display the correct confirmation dialog. Moreover, they should test their implementations against accidental or malicious user interactions aiming at bypassing the NC dialog.

User Account Takeover A vendor must prevent MITM attacks to fix this attack. For example, a vendor can mandate authenticated pairing as IVIs, and smartphones support input-output operations.

Recommendations Users should be careful when using Bluetooth in shared vehicles such as car rentals or car sharing. The manufacturers should not store Bluetooth data in clear text, and they should bind it to the PK and update/delete it accordingly. Vehicle manufacturers are well-positioned to fix, remediate, or increase the difficulty of Bluetooth exploitation. In addition to the previous recommendations, they should implement OTA updates to allow users to keep the IVI systems updated on their own.

9 Related Work

We cover the related security work on automotive systems such as the Controller Area Network (CAN) and IVI, as well as emerging fields, including autonomous driving and driver fingerprinting. Finally, we discuss BLE, which is gaining traction outside the low-power, embedded sector and is also seen in some modern vehicles.

IVI Security and Privacy. IVI are a common entry point for attackers. The authors of [40] describe a cybersecurity competition focusing on Automotive Grade Linux (AGL), an infotainment OS, which yielded 33 novel exploits. Further examples are provided by security evaluations of MirrorLink [57], a standard that integrates smartphones into IVI, and Mercedes-Benz User Experience (MBUX) [77], the IVI of Mercedes Benz.

Wireless Attacks on Cars. Beyond Bluetooth, there are other wireless technologies used in cars that have been subject to attacks, some of which are even exploited in the wild, for example, for car theft. Directly related to this, several key-less entry systems were analyzed, including KeeLoq [22, 25, 39], Remote Keyless System (RKS) [28, 41, 86, 87], Near-field communication (NFC) key cards and phone as a key [90].

Researchers also covered vehicle immobilizes like Hitag2 [11, 81], TI DST80 [88] and Megamos [80, 82]. Moreover, they looked at Wi-Fi and Bluetooth diagnostic dongles such as On-board diagnostics (OBD) [43, 85]. The authors of [69] further present a case study about the security and privacy of two popular Tire-pressure Monitoring System (TPMS), showing that tracking and spoofing attacks are possible.

Automotive Attacks. In [43], researchers showed attacks on a vehicle's internal CAN network, including Electronic Control Units (ECUs) such as Anti-lock Braking System (ABS), OBD, telematics, and even engine control. Follow-up studies extended the threat model to remote attackers and showed theoretical and practical attacks [18,58,59,62]. Recent studies presented more sophisticated attacks even capable of circumventing intrusion detection systems [13, 19, 23, 45, 46].

Automotive IDS and IPS The insecure nature of the vehicle internal bus inspired the development of specific Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) [3, 51, 89] to detect anomalies in the typical car network flow. This can be done either at the level of the message content, message intervals [75], or at the physical layer

(e.g., timing [68]). Despite such proposals being discussed for many years, they are not deployed in cars on the market.

Automotive Security Testing Several testing suites and frameworks exist for automotive security testing, and regulations in many countries have begun to require including such testing in the vehicle design process. One approach is the automated test generation and hardware threats enumeration based on Threat Analysis and Risk Assessment (TARA). The authors in [55] use the results of the TARA process, transferring threat models to attack trees and using a Domain Specific Language (DSL) to label the edges automatically. Other work [65] builds on TARA and employs Gajski-Kuhn Y-charts to represent attack manipulation systematically.

Four automotive security evaluation assurance levels (ASEAL) are proposed in [10] to evaluate potential security threats for automotive IT components. As cars are not always easily sourced for testing, there are several existing and proposed low-cost car hacking testbeds [47, 60, 79]. For CAN in particular, many testing and reverse-engineering tools are available [30, 44, 54].

Autonomous Driving While fully autonomous driving remains elusive, much security research is being conducted on the subsystems enabling partial autonomous driving and advanced driver-assistance systems. These include attacks on navigation systems used by autonomous vehicles [71], vision-based machine learning [52, 84], and Lidar [16, 36].

Driver Fingerprinting Several papers studied how to fingerprint a driver using data from a vehicle's internal network for malicious [26, 33] and benign [27, 48] use cases. They find it possible to distinguish drivers using only in-car sensors, at least in small driver pools.

BLE Security Testing BLE has attracted interest in the security community due to its strong growth in the embedded and asset-tracking sectors. Sweyntooth [32], a framework for fuzzing BLE devices, was evaluated on 12 devices and found 12 vulnerabilities. BLEDiff [42] uses differential testing in a black-box environment to identify noncompliant behavior in BLE implementations. It was evaluated on 25 devices, uncovering 10 attacks. None of these tools work on BC, and they cannot be easily extended to support it as they are tailored to BLE.

10 Conclusion

We present BlueToolkit, a new Bluetooth security testing framework. The tool performs OTA recon, exploitation, and reporting in a black-box fashion. It provides high coverage by testing design and implementation issues. It is extensible to future exploit tests and related software and hardware requirements. It is easy to install and use. Currently, BlueToolkit includes 44 tests.

We also present a systematic evaluation of automotive Bluetooth security using BlueToolkit. We tested 22 cars from 14 popular manufacturers produced between 2016 and 2023. Our experiments uncover 128 vulnerabilities, showing that the security of the automotive Bluetooth ecosystem is concerning and BlueToolkit is practical for real-world usage. Our results encourage anyone responsible for building and procuring cars, especially in sensitive domains, to strongly emphasize Bluetooth security. The reports produced by BlueToolkit can be a strong starting point.

While doing our experiments, we also discovered four attacks: *PBAP extraction*, an impersonation attack that allows learning previously synced phone contacts from an IVI, *JW Central Downgrade*, a pairing downgrade attack that enables a MITM position, *no NC*, an attack on pairing association enabling a MITM position, and the *user account takeover*, an attack leveraging a MITM position to hijack online accounts relying on SMS 2FA, thus with implications beyond the scope of Bluetooth. We responsibly disclosed our findings to the affected vendors and open source BlueToolkit.

Acknowledgments

Work funded by the European Union under grant agreement no. 101070008 (ORSHIN project) and partially supported by the French National Research Agency under the France 2030 label (NF-HiSec ANR-22-PEFT-0009). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or any other granting authority. Hence, they cannot be held responsible for them.

References

[1] Pyeongju Ahn, Yeonseok Jang, Seunghoon Woo, and Heejo Lee. BloomFuzz: Unveiling Bluetooth L2CAP Vulnerabilities via State Cluster Fuzzing with Target-Oriented State Machines. In *European Symposium on Research in Computer Security*, pages 110–129. Springer, 2024.

[2] Mingrui Ai, Kaiping Xue, Bo Luo, Lutong Chen, Nenghai Yu, Qibin Sun, and Feng Wu. Blacktooth: Breaking through the Defense of Bluetooth in Silence. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 55–68, 2022.

[3] Omar Y Al-Jarrah, Carsten Maple, Mehrdad Dianati, David Oxtoby, and Alex Mouzakitis. Intrusion detection systems for intra-vehicle networks: A review. *Ieee Access*, 7:21266–21289, 2019.

[4] Daniele Antonioli. BLUFFS: Bluetooth Forward and Future Secrecy Attacks and Defenses. In *ACM conference on Computer and Communications Security (CCS)*, November 2023.

[5] Daniele Antonioli and Mathias Payer. On the Insecurity of Vehicles Against Protocol-Level Bluetooth Threats. In *IEEE S&P Workshop On Offensive Technologies (WOOT)*, May 2022.

[6] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. Nearby Threats: Reversing, Analyzing, and Attacking Google's 'Nearby Connections' on Android. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, February 2019.

[7] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. BIAS: Bluetooth Impersonation AttackS. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 549–562. IEEE, 2020.

[8] Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen, and Mathias 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.

[9] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper 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)*, pages 1047–1061, 2019.

[10] Stephanie Bayer, Thomas Enderle, Dennis-Kengo Oka, and Marko Wolf. Security crash test-practical security evaluations of automotive onboard IT components. In *"Automotive - Safety & Security 2014"*. Gesellschaft für Informatik eV, 2015.

[11] Ryad Benadjila, Mathieu Renard, José Lopes-Esteves, and Chaouki Kasmi. One car, two frames: attacks on Hitag-2 remote keyless entry systems revisited. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[12] Eli Biham and Lior 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.

[13] Gedare Bloom. WeepingCAN: A stealthy CAN bus-off attack. In *Workshop on Automotive and Autonomous Vehicle Security*, 2021.

[14] Bluetooth SIG. Bluetooth: 20 years of innovation. https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/20year/default.html, 2020.

[15] Bluetooth SIG. Bluetooth Core Specification v6.0. https://www.bluetooth.com/specifications/specs/core-specification-6-0/, 2024.

[16] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *2021 IEEE symposium on security and privacy (SP)*, pages 176–194. IEEE, 2021.

[17] Madeline Cheah, Siraj A Shaikh, Olivier Haas, and Alastair Ruddle. Towards a systematic security evaluation of the automotive Bluetooth interface. *Vehicular Communications*, 9:8–18, 2017.

[18] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX security symposium (USENIX Security 11)*, 2011.

[19] Kyong-Tak Cho and Kang G Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1044–1055, 2016.

[20] Tristan Claverie, Gildas Avoine, Stéphanie Delaune, and José Lopes Esteves. Tamarin-based Analysis of Bluetooth Uncovers Two Practical Pairing Confusion Attacks. In *European Symposium on Research in Computer Security*, pages 100–119. Springer, 2023.

[21] Tristan Claverie and José Lopes Esteves. Bluemirror: reflections on Bluetooth pairing and provisioning protocols. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 339–351. IEEE, 2021.

[22] Nicolas T Courtois, Gregory V Bard, and David Wagner. Algebraic and slide attacks on KeeLoq. In *Fast Software Encryption: 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers 15*, pages 97–115. Springer, 2008.

[23] Alvise de Faveri Tron, Stefano Longari, Michele Carminati, Mario Polino, and Stefano Zanero. Canflict: exploiting peripheral conflicts for data-link layer attacks on automotive networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 711–723, 2022.

[24] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[25] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T Manzuri Shalmani. On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme. In *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28*, pages 203–220. Springer, 2008.

[26] Miro Enev, Alex Takakuwa, Karl Koscher, and Tadayoshi Kohno. Automobile driver fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016.

[27] Saad Ezzini, Ismail Berrada, and Mounir Ghogho. Who is behind the wheel? Driver identification and fingerprinting. *Journal of Big Data*, 5(1):1–15, 2018.

[28] Aurélien Francillon, Boris Danev, and Srdjan Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 2011.

[29] Frank Brown. Report: Biggest Car Manufacturers by Revenue, 2023. https://ceoworld.biz/2023/10/25/report-biggest-car-manufacturers-by-revenue-2023/, 2024.

[30] Daniel Frassinelli, Sohyeon Park, and Stefan Nürnberger. I know where you parked last summer: Automated reverse engineering and privacy analysis of modern cars. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1401–1415. IEEE, 2020.

[31] Matheus E Garbelini, Vaibhav Bedi, Sudipta Chattopadhyay, Sumei Sun, and Ernest Kurniawan. BrakTooth: Causing Havoc on Bluetooth Link Manager via Directed Fuzzing. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1025–1042, 2022.

[32] Matheus E. Garbelini, Chundong Wang, Sudipta Chattopadhyay, Sun Sumei, and Ernest Kurniawan. SweynTooth: Unleashing Mayhem over Bluetooth Low Energy. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 911–925. USENIX Association, 2020.

[33] András Gazdag, Szilvia Lestyán, Mina Remeli, Gergely Ács, Tamás Holczer, and Gergely Biczók. Privacy pitfalls of releasing in-vehicle network data. *Vehicular Communications*, 39:100565, 2023.

[34] Răzvan Andrei Gheorghiu, Valentin Iordache, and Angel Ciprian Cormoș. Analysis of the possibility to detect road vehicles via bluetooth technology. *Sensors*, 21(21):7281, 2021.

[35] Keijo Haataja and Pekka Toivanen. Two practical man-in-the-middle attacks on Bluetooth Secure Simple Pairing and countermeasures. *Transactions on Wireless Communications*, 9(1):384–392, 2010.

[36] R Spencer Hallyburton, Yupei Liu, Yulong Cao, Z Morley Mao, and Miroslav Pajic. Security analysis of Camera-LiDAR fusion against Black-Box attacks on autonomous vehicles. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1903–1920, 2022.

[37] Sönke Huster, Matthias Hollick, and Jiska Classen. To boldly go where no Fuzzer has gone before: Finding bugs in Linux wireless stacks through VirtIO devices. In *2024 IEEE Symposium on Security and Privacy (SP). Los Alamitos, CA, USA: IEEE Computer Society*, pages 24–24, 2024.

[38] Konstantin Hypponen and Keijo MJ Haataja. Nino man-in-the-middle attack on Bluetooth Secure Simple Pairing. In *Proceedings of the International Conference in Central Asia on Internet*, pages 1–5. IEEE, 2007.

[39] Sebastiaan Indesteege, Nathan Keller, Orr Dunkelman, Eli Biham, and Bart Preneel. A practical attack on keeloq. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–18. Springer, 2008.

[40] Seonghoon Jeong, Minsoo Ryu, Hyunjae Kang, and Huy Kang Kim. Infotainment System Matters: Understanding the Impact and Implications of In-Vehicle Infotainment System Hacking with Automotive Grade Linux. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy*, pages 201–212, 2023.

[41] Samy Kamkar. Drive it like you hacked it: New attacks and tools to wirelessly steal cars. *Presentation at DEFCON*, 23:10, 2015.

[42] Imtiaz Karim, Abdullah Al Ishtiaq, Syed Rafiul Hussain, and Elisa Bertino. BLEDiff: Scalable and Property-Agnostic Noncompliance Checking for BLE Implementations. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1082–1100. IEEE Computer Society, 2023.

[43] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. Experimental security analysis of a modern automobile. In *2010 IEEE symposium on security and privacy*, pages 447–462. IEEE, 2010.

[44] Sekar Kulandaivel, Tushar Goyal, Arnav Kumar Agrawal, and Vyas Sekar. CANvas: Fast and inexpensive automotive network mapping. In *28th USENIX*

Security Symposium (USENIX Security 19), pages 389–405, 2019.

[45] Sekar Kulandaivel, Shalabh Jain, Jorge Guajardo, and Vyas Sekar. Cannon: Reliable and stealthy remote shutdown attacks via unaltered automotive microcontrollers. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 195–210. IEEE, 2021.

[46] Sekar Kulandaivel, Shalabh Jain, Jorge Guajardo, and Vyas Sekar. CANdid: A Stealthy Stepping-Stone Attack to Bypass Authentication on ECUs. *Journal on Autonomous Transportation Systems*, 2024.

[47] Sekar Kulandaivel, Wenjuan Lu, Brandon Barry, and Jorge Guajardo. Towards a New Configurable and Practical Remote Automotive Security Testing Platform. *arXiv preprint arXiv:2404.02291*, 2024.

[48] Byung Il Kwak, JiYoung Woo, and Huy Kang Kim. Know your master: Driver profiling-based anti-theft method. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 211–218. IEEE, 2016.

[49] Predicta Lab. Predicta search: Reverse email or phone number lookup. https://www.predictasearch.com.

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

[51] Siti-Farhana Lokman, Abu Talib Othman, and Muhammad-Husaini Abu-Bakar. Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–17, 2019.

[52] Giulio Lovisotto, Henry Turner, Ivo Sluganovic, Martin Strohmeier, and Ivan Martinovic. SLAP: Improving physical adversarial examples with Short-Lived adversarial perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1865–1882, 2021.

[53] Dennis Mantz, Jiska Classen, Matthias Schulz, and Matthias Hollick. InternalBlue Bluetooth binary patching and experimentation framework. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 79–90, 2019.

[54] Mirco Marchetti and Dario Stabili. READ: Reverse engineering of automotive data frames. *IEEE Transactions on Information Forensics and Security*, 14(4):1083–1097, 2018.

[55] Stefan F Marksteiner, Christoph Schmittner, Korbinian Christl, Dejan Nickovic, Mikael Sjödin, and Marjan Sirjani. From TARA to Test: Automated Automotive Cybersecurity Test Generation Out of Threat Modeling. In *Proceedings of the 7th ACM Computer Science in Cars Symposium*, pages 1–10, 2023.

[56] Mathilde Carlier. Leading car manufacturers by sales in 2023. https://www.statista.com/statistics/1419835/leading-car-manufacturers-by-cumulative-sales/, 2024.

[57] Sahar Mazloom, Mohammad Rezaeirad, Aaron Hunter, and Damon McCoy. A security analysis of an in-vehicle Infotainment and App platform. In *10th USENIX workshop on offensive technologies (WOOT 16)*, 2016.

[58] Charlie Miller. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015.

[59] Charlie Miller and Chris Valasek. Adventures in automotive networks and control units. *Def Con*, 21(260-264):15–31, 2013.

[60] Charlie Miller and Chris Valasek. Car hacking: for poories. *Tech. rep., IOActive Report, Tech. Rep.*, 2015.

[61] Andy Nguyen. BleedingTooth: Linux Bluetooth Zero-Click Remote Code Execution. https://google.github.io/security-research/pocs/linux/bleedingtooth/writeup, 2020.

[62] Sen Nie, Ling Liu, and Yuefeng Du. Free-fall: Hacking Tesla from wireless to CAN bus. *Briefing, Black Hat USA*, 25(1):16, 2017.

[63] Tom Nijholt, Erik Poll, and Frits Vaandrager. BlueSpec: Development of an LMP state machine and a stateful black-box BR/EDR LMP fuzzer. https://www.cs.ru.nl/masters-theses/2020/T_Nijholt___BlueSpec:_Development_of_an_LMP_state_machine_and_a_stateful_black-box_BR%26sol%3BEDR_LMP_fuzzer.pdf, 2020.

[64] Haram Park, Carlos Kayembe Nkuba, Seunghoon Woo, and Heejo Lee. L2Fuzz: Discovering Bluetooth L2CAP Vulnerabilities Using Stateful Fuzz Testing. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 343–354. IEEE, 2022.

[65] James Pickford, Rasadhi Attale, Siraj Shaikh, Hoang Nga Nguyen, and Lee Harrison. Systematic Risk Characterisation of Hardware Threats to Automotive Systems. *Journal on Autonomous Transportation Systems*, 1(4):1–36, 2024.

[66] Rapid7. Metasploit. https://www.metasploit.com.

[67] Vishnu Renganathan, Ekim Yurtsever, Qadeer Ahmed, and Aylin Yener. Valet attack on privacy: a cybersecurity threat in automotive Bluetooth infotainment systems. *Cybersecurity*, 5(1):30, 2022.

[68] Marc Roeschlin, Giovanni Camurati, Pascal Brunner, Singh Mridula, and Capkun Srdjan. EdgeTDC: On the security of time difference of arrival measurements in CAN bus systems. In *NDSS*, 2023.

[69] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. Security and privacy vulnerabilities of In-Car wireless networks: A tire pressure monitoring system case study. In *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[70] Jan Ruge, Jiska Classen, Francesco Gringoli, and Matthias Hollick. Frankenstein: Advanced wireless fuzzing to exploit new Bluetooth escalation targets. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 19–36, 2020.

[71] Harshad Sathaye, Martin Strohmeier, Vincent Lenders, and Aanjhan Ranganathan. An experimental study of GPS spoofing and takeover attacks on UAVs. In *31st USENIX security symposium (USENIX security 22)*, pages 3503–3520, 2022.

[72] Ben Seri and Gregory Vishnepolsky. The Attack Vector BlueBorne Exposes Almost Every Connected Device. https://armis.com/blueborne/, 2017.

[73] Ben Seri, Gregory Vishnepolsky, and Dor Zusman. BLEEDINGBIT: The hidden attack surface within BLE chips. https://armis.com/bleedingbit/, 2019.

[74] Yaniv Shaked and Avishai Wool. Cracking the Bluetooth PIN. In *Proceedings of the conference on Mobile systems, applications, and services (MobiSys)*, pages 39–50. ACM, 2005.

[75] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In *2016 international conference on information networking (ICOIN)*, pages 63–68. IEEE, 2016.

[76] Da-Zhi Sun, Yi Mu, and Willy Susilo. Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard v5. 0 and its countermeasure. *Personal and Ubiquitous Computing*, 22(1):55–67, 2018.

[77] Tencent Security Team. Experimental Security Assessment of Mercedes-Benz Cars . https://keenlab.tencent.com/en/2021/05/12/Tencent-Security-Keen-Lab-Experimental-Security-Assessment-on-Mercedes-Benz-Cars/, 2021.

[78] WHAD Team. Whad client. https://github.com/whad-team/whad-client.

[79] Tsuyoshi Toyama, Takuya Yoshida, Hisashi Oguma, and Tsutomu Matsumoto. PASTA: Portable automotive security testbed with adaptability. *London, blackhat Europe*, 2018.

[80] Roel Verdult and Flavio D Garcia. Cryptanalysis of the Megamos Crypto automotive immobilizer. *USENIX; login*, 40(6):17–22, 2015.

[81] Roel Verdult, Flavio D Garcia, and Josep Balasch. Gone in 360 seconds: Hijacking with Hitag2. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 237–252, 2012.

[82] Roel Verdult, Flavio D Garcia, and Baris Ege. Dismantling Megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In *Supplement to the Proceedings of 22nd USENIX Security Symposium (Supplement to USENIX Security 15)*, pages 703–718, 2015.

[83] Maximilian von Tschirschnitz, Ludwig Peuckert, Fabian Franzen, and Jens Grossklags. Method confusion attack on Bluetooth pairing. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1332–1347. IEEE, 2021.

[84] Ningfei Wang, Yunpeng Luo, Takami Sato, Kaidi Xu, and Qi Alfred Chen. Does physical adversarial example really matter to autonomous driving? Towards system-level effect of adversarial object evasion attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4412–4423, 2023.

[85] Haohuang Wen, Qi Alfred Chen, and Zhiqiang Lin. Plug-N-Pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new Over-the-Air attack surface in automotive IoT. In *29th USENIX Security Symposium (USENIX Sec 20)*, pages 949–965, 2020.

[86] Lennert Wouters, Benedikt Gierlichs, and Bart Preneel. My other car is your car: compromising the Tesla Model X keyless entry system. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 149–172, 2021.

[87] Lennert Wouters, Eduard Marin, Tomer Ashur, Benedikt Gierlichs, and Bart Preneel. Fast, furious and insecure: Passive keyless entry and start systems in modern supercars. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):66–85, 2019.

[88] Lennert Wouters, Jan Van den Herrewegen, Flavio D Garcia, David Oswald, Benedikt Gierlichs, and Bart Preneel. Dismantling DST80-based immobiliser systems. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):99–127, 2020.

[89] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):919–933, 2019.

[90] Xinyi Xie, Kun Jiang, Rui Dai, Jun Lu, Lihui Wang, Qing Li, and Jun Yu. Access Your Tesla without Your Awareness: Compromising Keyless Entry System of Model 3. In *NDSS*, 2023.

A Open Science and Ethics

Open Science The toolkit and its sub-modules are fully open-source at https://github.com/sgxgsx/BlueToolkit. The hardware required to run the tests is commercial and available off the shelf.

Ethical Considerations This research aims to investigate protocol-level and implementation-specific vulnerabilities in Bluetooth technology as applied to modern vehicles. Through a comprehensive case study, the study seeks to identify novel vulnerabilities within the Bluetooth standard and to develop an open-source toolkit to facilitate reproducibility and further research. The findings of this study have the potential to significantly benefit a wide range of end-users and organizations by enhancing the security of Bluetooth-enabled systems in vehicles.

All experiments were conducted in a controlled environment using vehicles provided by a government entity, which granted explicit informed consent to use their vehicles in this research. The research exclusively utilized test data, ensuring no personal or user data was accessed, compromised, or otherwise affected. After the case study, all vehicles remained fully operational, with no adverse effects resulting from the conducted tests.

We provided the Bluetooth SIG and all affected manufacturers with detailed information on the identified vulnerabilities and weaknesses as well as remediation steps. At the time of submission, some vendors acknowledged our findings. However, since the responsible disclosure period has ended, all the results are publicly available in our GitHub repository.