Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

ETH zürich

EURECOM
Sophia Antipolis

Bluetooth Security Testing with BlueToolkit

A Large-Scale Automotive Case Study

V. Zubkov[1], **T. Sacchetti**[2,3], D. Antonioli[2], M. Strohmeier[3]

1. ETH Zurich, 2. EURECOM 3. armasuisse

Bluetooth

- Short-range wireless communication protocol
- Bluetooth Classic (BR/EDR)
  - Audio streaming
  - High-rate data transfer
- Bluetooth Low Energy (BLE)
  - Low-rate data transfer
  - Low-latency audio streaming (recent)
- Bluetooth security follows the Trust On First Use (TOFU) model

Bluetooth Security

Pairing

- Long-term key agreement protocol
- Establishes a root-of-trust (pairing key)
- Optionally authenticated
  - User-assisted
  - Out-of-band channel

Session Establishment

- Short-term key agreement protocol
- Derives an encryption key (session key)
- Happens between paired devices

Bluetooth Security

- Bluetooth is affected by many vulnerabilities resulting in **privacy** and **security** threats
  - More than 108 known attacks
  - More than 900 CVE entries
- **Protocol-level**
  - Caused by errors in the standard specifications
  - Can affect all devices
  - Result in spoofing and MitM
- **Implementation-level**
  - Vary across different stacks
  - Result in DoS, RCE, zeroed keys

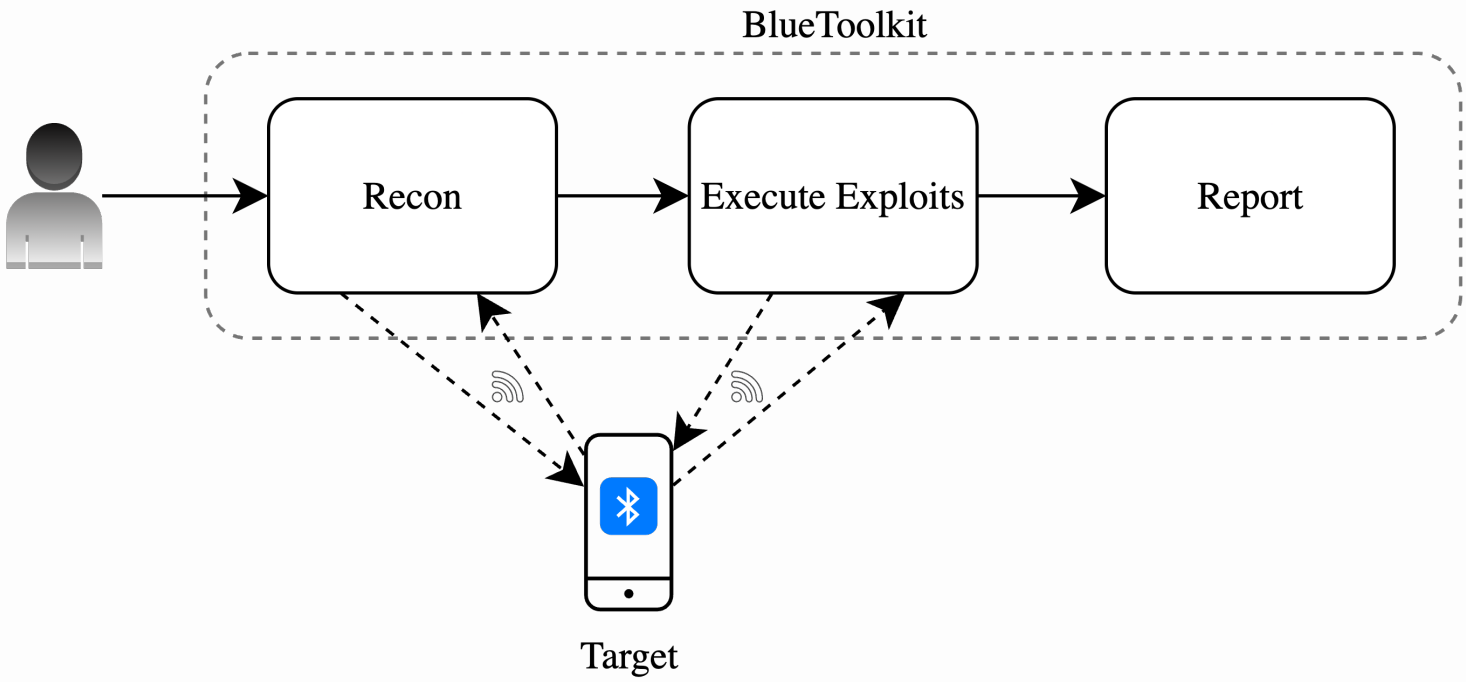
Why BlueToolkit?

- Automated large-scale testing is challenging
  - Bluetooth stacks are mostly closed source
  - Existing tools are not designed to work together
  - Proof of Concepts (PoCs) are scattered and with different requirements
  - Braktooth: ESP-WROVER_KIT + custom firmware
  - Sweyntooth: NRF52 + closed-source custom firmware
  - Internalblue: rooted Google Nexus 5

What is BlueToolkit?

- Comprehnsive Bluetooth security testing framework
- Enables black-box over-the-air vulnerability testing
- Enables systematic Bluetooth testing at scale (ala Metasploit)
- Produces redable reports
- Easily extensible with new exploits (no hawrdare restrictions)
- Initially BR/EDR only, now with BLE support

BlueToolkit Design



Recon Module

- Collects operational parameters
  - Bluetooth version, modes (BLE, BR/EDR), profiles (PBAP, A2DP), security (LSC, SC)
- Produces a JSON report for the exploit execution module
- Testing is tailored to targets’ capabilities

Recon Module

- Abstracts over BLE and BC
- Scapy-based
- Human- and machine-readable JSON report


```
"type": "BLE",
"advertising": true,
"connectable": true,
"version": 0.0,
"vendor": "Unknown",
"ll_features": [],
"pairable": true,
"pairing_features": {
    "io_capabilities": "NoInputNoOutput",
    "auth_req": "Bonding, No MITM, LSC, No Keypress Notifications",
    "max_key_size": 16
}
```

Exploit Execution Module

- Executes exploits based on recon data and user preferences
- Supports design and implementation level attacks
  - MitM, RCE, DoS, Auth bypass, information leak
- Exploits can terminate either with a timeout or via a specific condition

Exploit Execution Module

- Exploits can be Python scripts, binaries, ...
- Uses YAML templates to abstract over exploits
  - Contains execution parameters, scripts names, and location
  - Processed by the engine, which executes the actual exploit

```yaml
author: sacca
bt_type: BLE
bt_version_max: 5.4
bt_version_min: 2.0
command: 'python3 bluekit_knob_ble.py '
directory:
  change: true
  directory: modules/tools/custom_exploits
hardware: default
log_pull:
  from_directory: false
  in_command: false
mass_testing: true
name: reconnaissance_knob_ble
parameters:
- help: Target MAC address
  is_target_param: true
  name: --target
  name_required: true
  parameter_connector: ' '
  required: true
  type: str
type: PoC
```

KNOB Attack Exploit

- Example: KNOB (https://dl.acm.org/doi/abs/10.1145/3394497)
- Pairing lets device negotiating the entropy of the pairing key (7 to 16 bytes)
- A MitM attacker can foce a low key entropy (7) and then brute force it
- Knowledge of the pairing key allows:
  - Decrypting past and future messages
  - Devices impersonation

KNOB Attack Exploit

```
target = args.target

dev = Device(role=BLE_ROLE_CENTRAL, bt_mode=BT_MODE_BLE)
dev.power_on()

vals = [7, 1]
min_key_size = 16
for v in vals:
    time.sleep(1)
    dev.connect(addr=target, bt_type=BT_MODE_BLE)
    print(f"Testing with LTK size: {v} bytes")
    dev.sm.ltk_size = v
    flag, features = dev.pair()
    if flag and features is not None:
        min_key_size = v
    else:
        break

if min_key_size < 16:
    report_vulnerable(f"Minimum key size is {min_key_size}, can be brute-forced")
else:
    report_not_vulnerable(f"Minimum key size is {min_key_size}, enough entropy")
```

Report Module

- Produces a JSON report
- Machine- and human-readable
- Summarizes the results of the exploit execution module

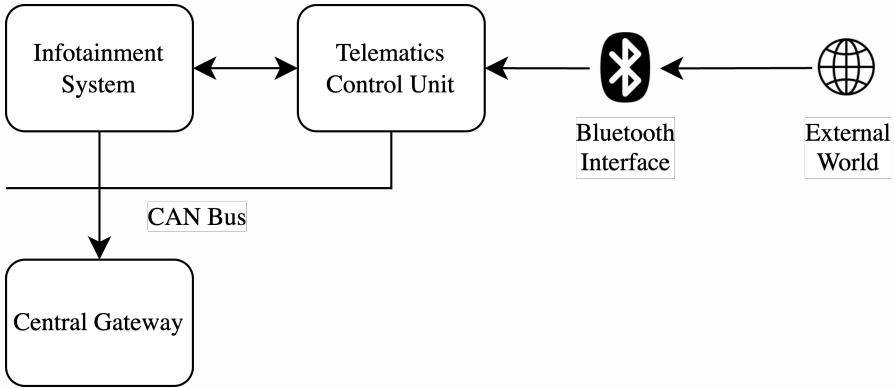
```
"code": 0,
"data": "Minimum key size is 7, can be brute-forced"
```

Evaluation

Large-Scale Automotive Testing

Automotive & Bluetooth

- Cars have Bluetooth (typically BR/EDR)
- Used in Infotainment Systems (IVI) for
  - Calls, contacts sharing, music streaming, and more
- IVIs are connected to the rest of the car
  - Remote Exploitation of Nissan Leaf (Black Hat Asia 2025)



Evaluation

- Tested 22 vehicles from 14 leading manufacturers
  - Years 2015 – 2023
  - Bluetooth versions 2.1 – 5.2
  - 6 chips manufacturers
- Uncovered 189 vulnerabilities affecting IVIs Bluetooth
  - automotive Bluetooth security is weak!

Results (subset)

Vehicle	Year	Type	Tests	Pos.
Skoda Octavia	2022	Classic	44	2
Renault Zoe	2021	EV	44	8
Chevrolet Corvette	2018	Classic	44	7
Audi e-tron	2020	EV	44	4
Mini Cooper	2022	Classic	44	7
Toyota Corolla	2023	Hybrid	44	9
Tesla Model Y	2023	EV	44	2

Uncovering Attacks

Contact Extractor

Information Leak

Insecure Numeric Comparison

Authentication Bypass

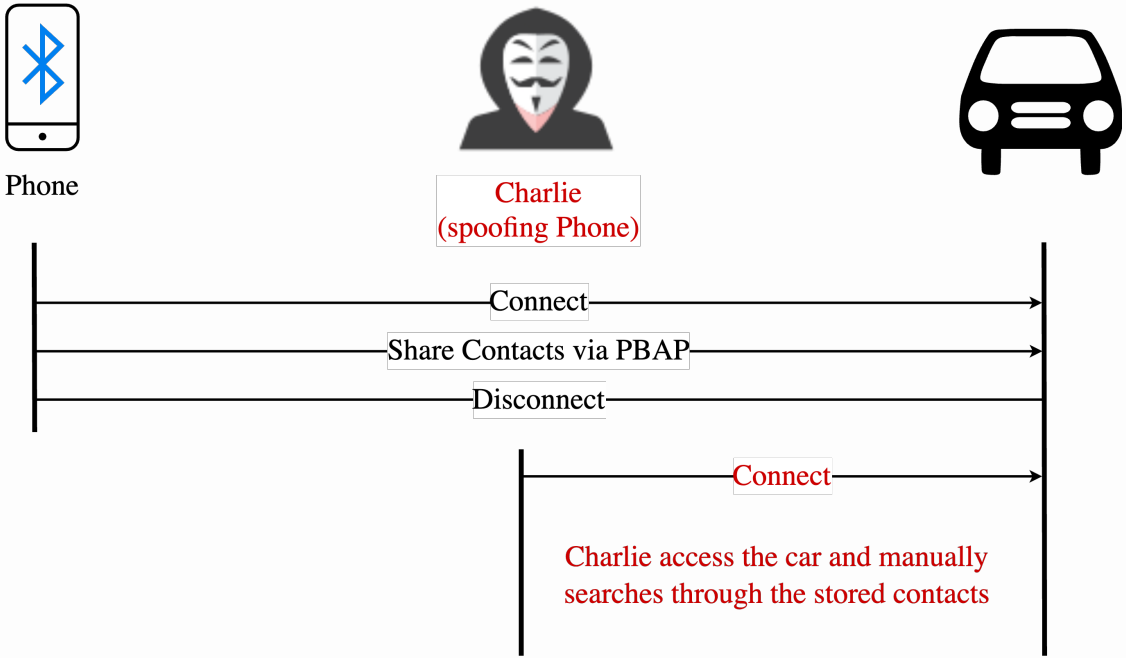
JustWorks Downgrade

Authentication Bypass

Contact Extractor

- Extracting contacts from an IVI system
- Contats are shared via PBAP by a smartphone and stored in the IVI, they
  - Are linked to a Bluetooth address (MAC)
  - Are **not** encrypted
  - Can be accessed **without** authentication
- Attackers can
  - Spoof a legitimate device and access the contacts
- Requires physical vehicle access
  - Car sharing
  - Rental cars

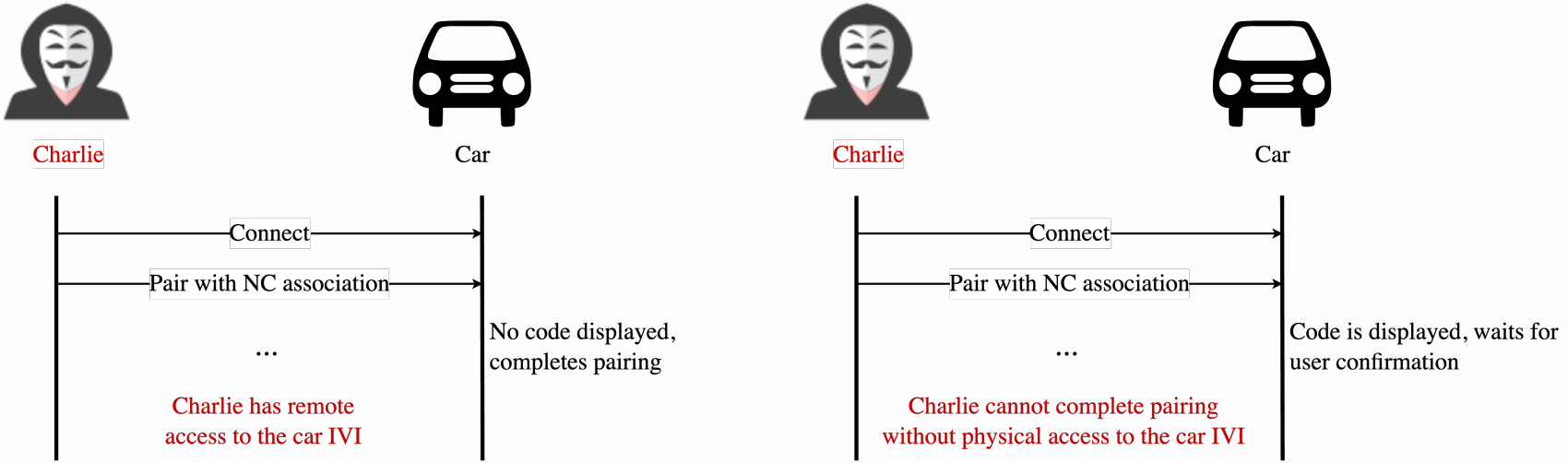
Contact Extractor



Insecure Numeric Comparison

- Pairing authenticated via Numeric Comparison (NC)
  - Devices display a number, user confirms they match
  - Otherwise, pairing is canceled
- Implementations «forgot» to display the number, proceeding with pairing
- Results in authentication bypass without user interaction

Insecure Numeric Comparison



Conclusion

- State-of-the-art Bluetooth security testing framework (490+ stars on GitHub)
  - https://github.com/sgxgsx/BlueToolkit
- First comprehensive security evaluation of Bluetooth in vehicles
  - 22 Vehicles
  - 189 vulnerabilities found
- Uncovered car- and non-car-specific issues
  - Contact extractor
  - Insecure NC

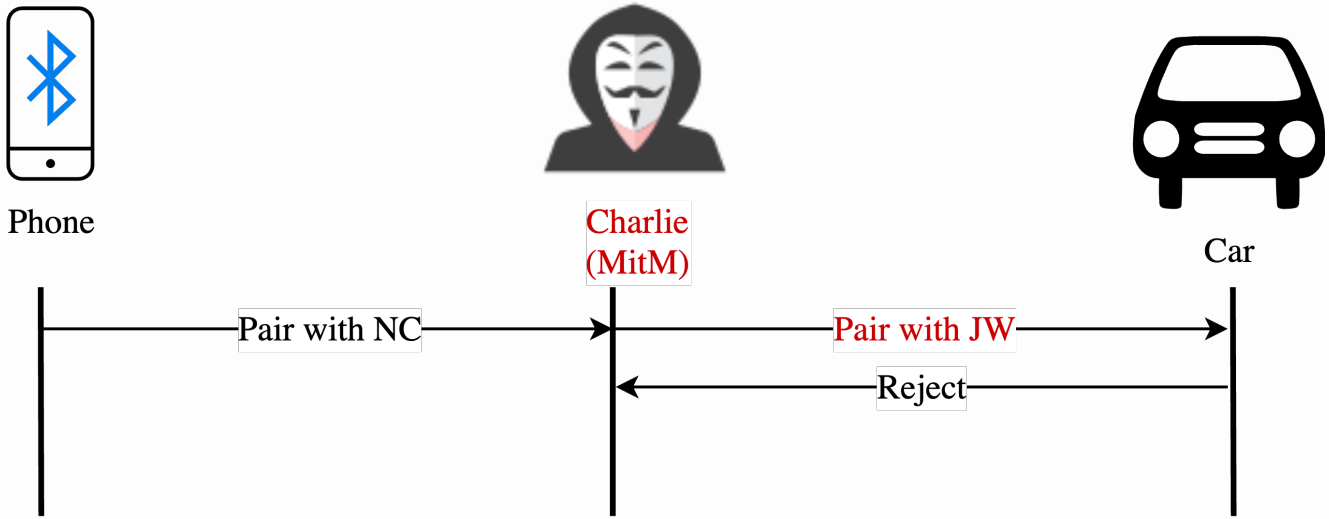


Backup Slides

JustWorks Central Downgrade

- Bluetooth connection has two roles:
  - Initiator = Central
  - Responder = Peripheral
- Vendors can enforce authenticated pairing (e.g., only use NC)
- Depending on how this is done, it can be bypassed
- Some vendors were
  - Rejecting unauthenticated pairing if Peripheral role
  - Accepting unauthenticated pairing if Central role
- In BR/EDR these role can switch
- Result: authentication enforcement is no more

JustWorks Central Downgrade



JustWorks Central Downgrade

