
# Car Security in Bluetooth Era

## Threats, Tools, and Tactics

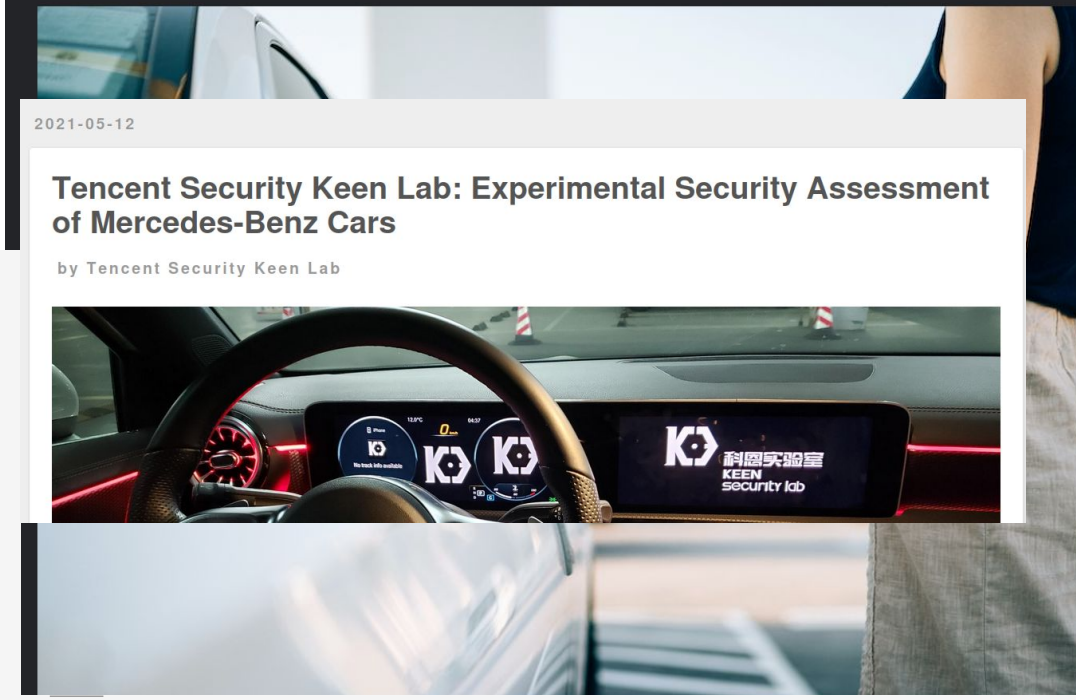
# Hackers Remotely Kill a Jeep on the Highway—With Me in It

I was driving 70 mph on the edge of downtown St. Louis when the exploit began to take hold.

# New Bluetooth hack can unlock your Tesla—and all kinds of other devices

All it takes to hijack Bluetooth-secured devices is custom code and $100 in hardware.

DAN GOODIN – MAY 19, 2022 5:00 PM | 274



2021-05-12

## Tencent Security Keen Lab: Experimental Security Assessment of Mercedes-Benz Cars

by Tencent Security Keen Lab




## Hacking Kia: Remotely Controlling Cars With Just a License Plate

Fri Sep 20 2024

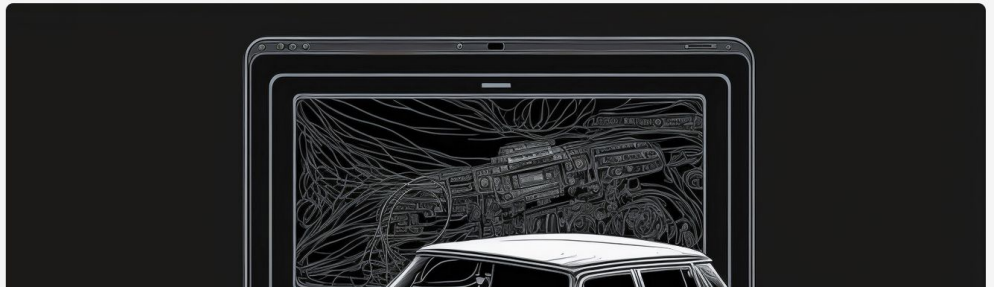
2020-01-02

## Exploiting Wi-Fi Stack on Tesla Model S

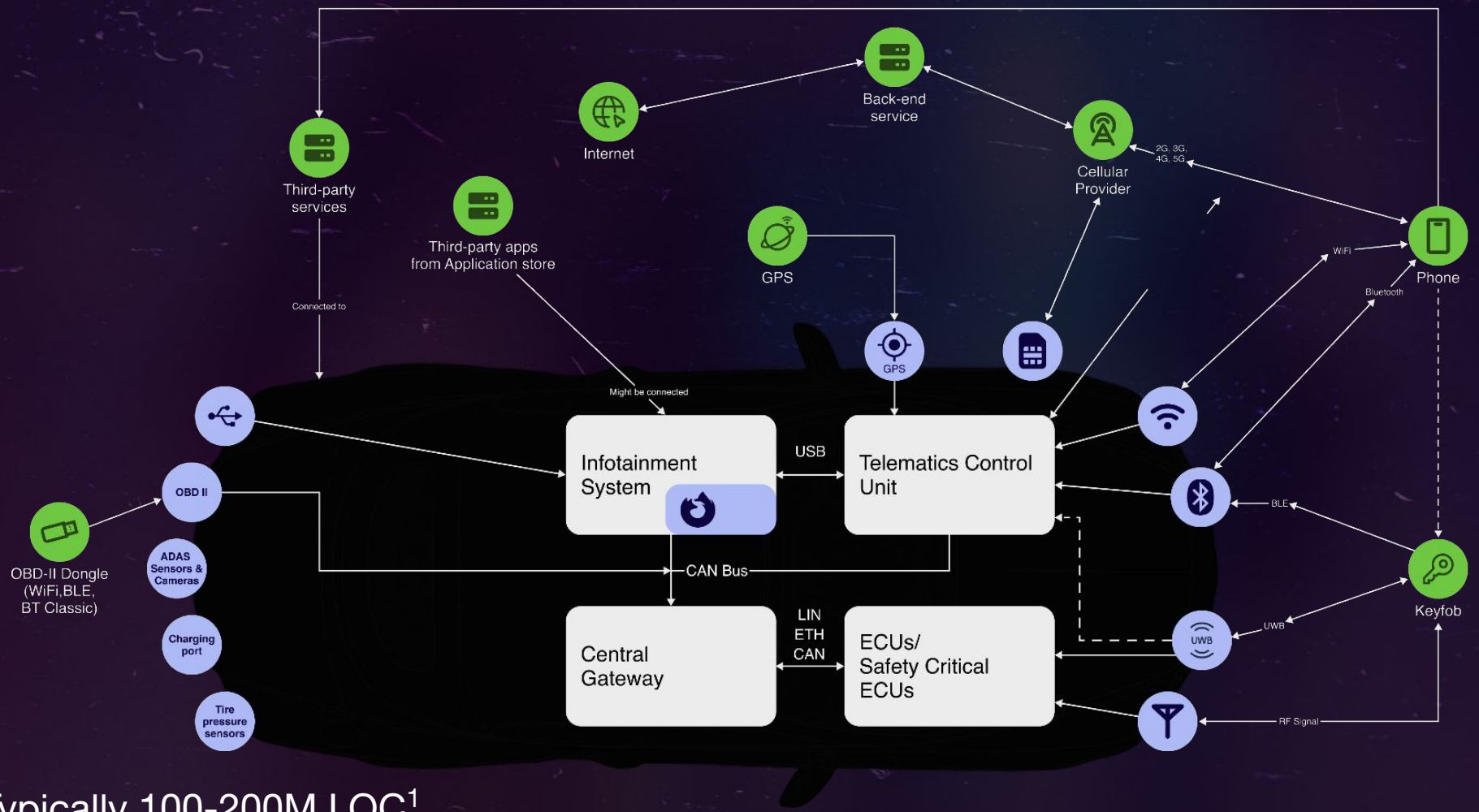
by Tencent Keen Security Lab


## Web Hackers vs. The Auto Industry: Critical Vulnerabilities in Ferrari, BMW, Rolls Royce, Porsche, and More

Tue Jan 03 2023



# Cars are complex

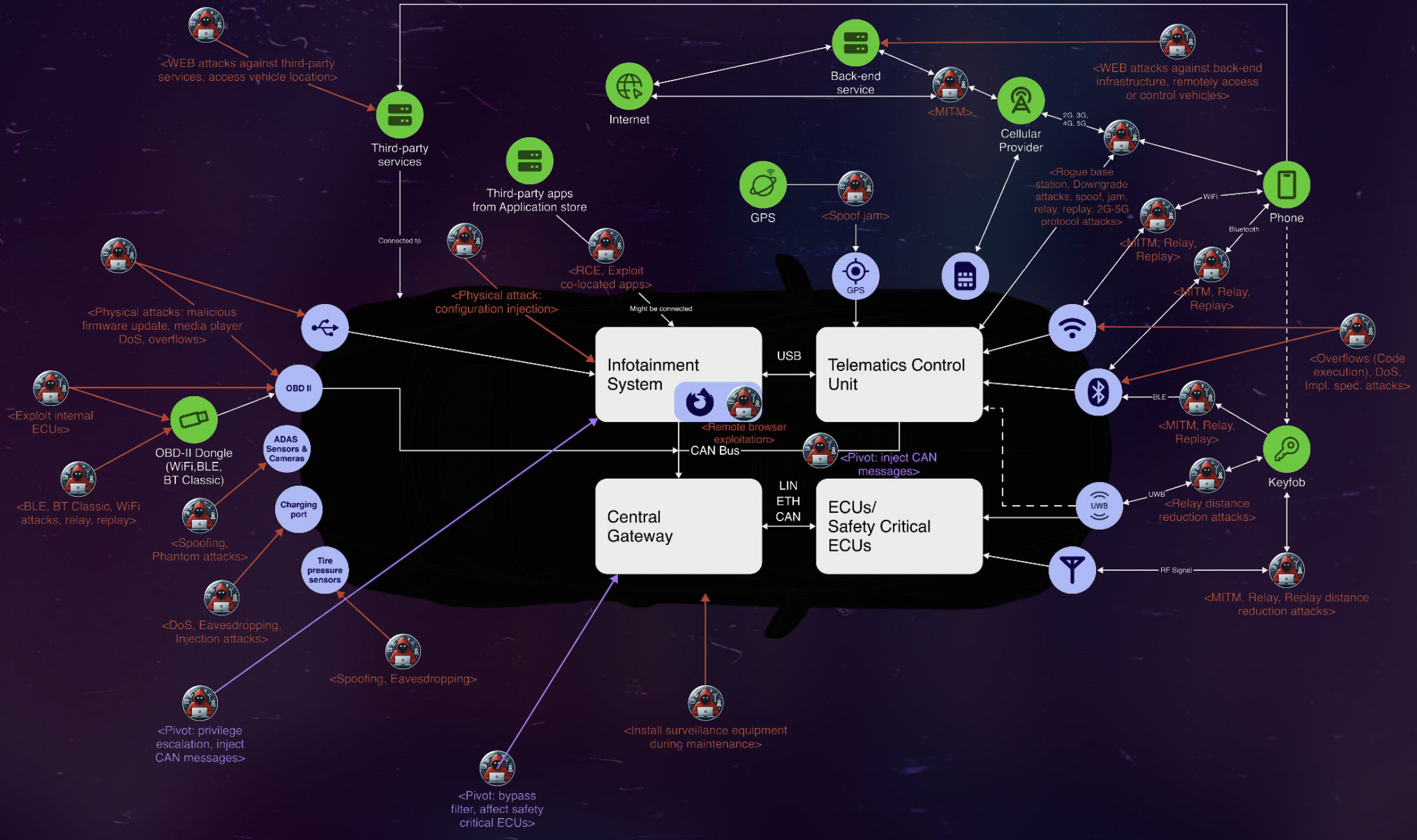



- Typically 100-200M LOC[1]
- Includes QNX, Android Automotive OS, AGL with libraries for interfaces (Bluetooth, WiFi, GPS, Cellular, UWB)

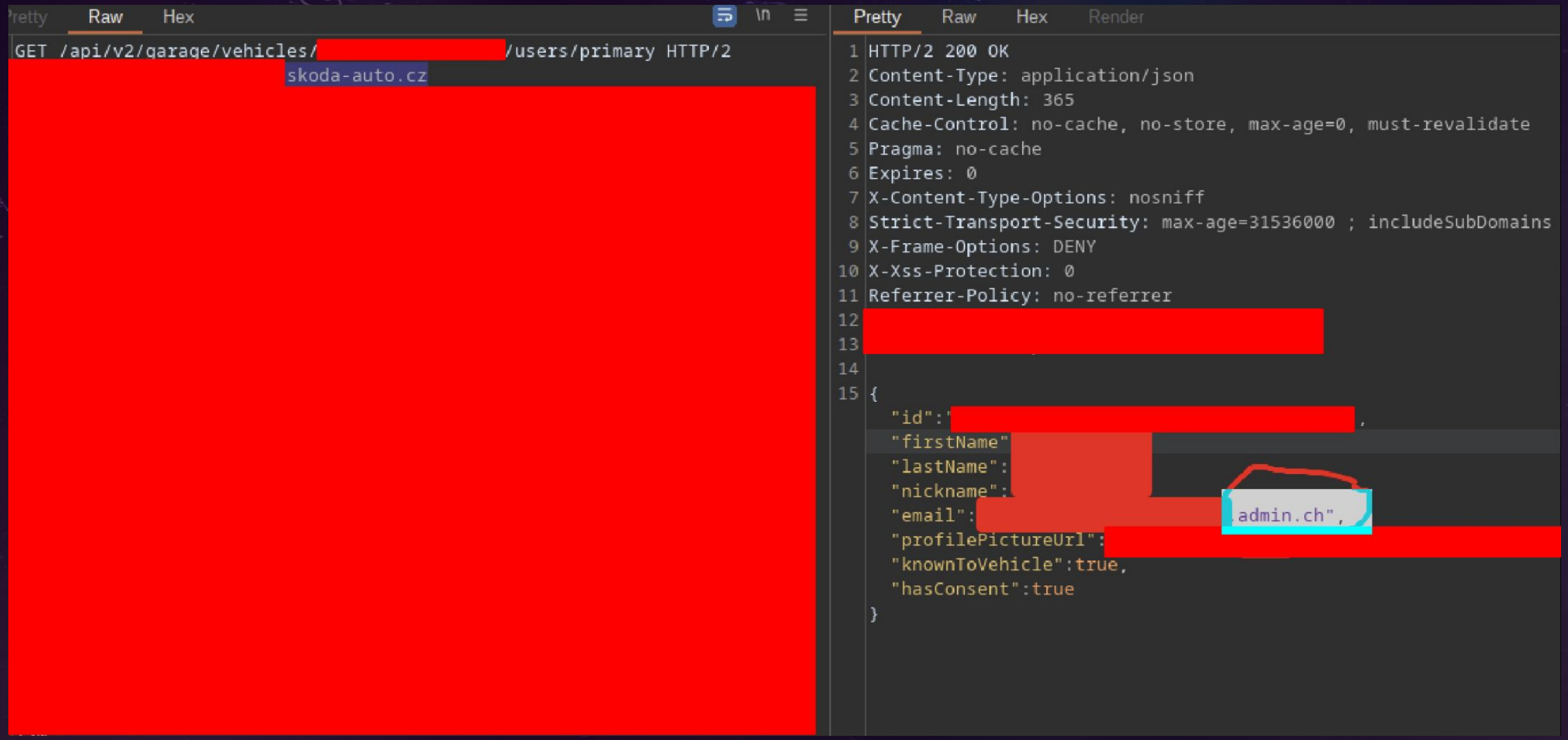
1. https://www.statista.com/statistics/1370978/automotive-software-average-lines-of-codes-per-vehicle-globally/


WITH GREAT POWER
COMES GREAT RESPONSIBILITY
makeameme.org

# Cars are complex … and so is their attack surface



# Cars are complex … and so is their attack surface




Responsibly reported to Skoda Auto Bug Bounty program - https://app.intigriti.com/programs/skodaauto/skodaautoprivatebugbountyprogram/detail

# Focus

**Remote**

- Web, UWB, Bluetooth, WiFi, Cellular, GPS, …

**Local**

- USB, OBD-II, pivoting attacks from 1 component to another, …

**Implementation-specific**

- IoT hacking* (components)
- Reverse Engineering
- DAST/SAST

Pros: Higher Impact* (RCE)

Cons: Lower exposure* (Firmware specific)

**Protocol-specific**

- Specification review
- Studying protocol attacks
- Reengineering known vulnerabilities

Pros: Higher exposure (standard functionality)

Cons: Medium-High impact (depends on the protocol)

# Focus

**Remote**

- Web, UWB, Bluetooth, WiFi, Cellular, GPS, …

**Local**

- USB, OBD-II, pivoting attacks from 1 component to another, …

**Implementation-specific**

- IoT hacking* (components)
- Reverse Engineering
- DAST/SAST

Pros: Higher Impact* (RCE)

Cons: Lower exposure* (Firmware specific)

**Protocol-specific**

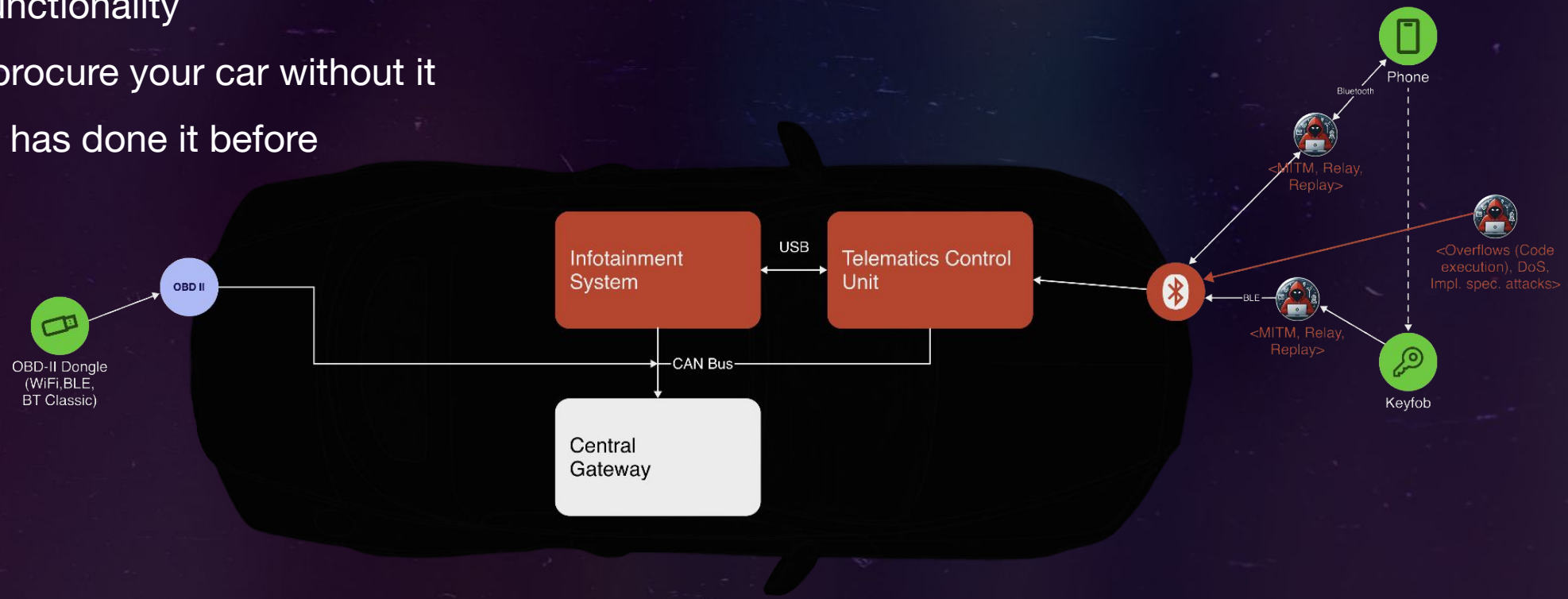
- Specification review
- Studying protocol attacks
- Reengineering known vulnerabilities

Pros: Medium-High impact (depends on Protocol)

Cons: Higher exposure (standard functionality

# Why Examine Bluetooth in Cars?

- 1 billion cars have Bluetooth (typically BT Classic) [1,2]
- 87% of new cars have BT[2]
- Rich functionality
- Can’t procure your car without it
- Noone has done it before




1. https://www.thedrive.com/guides-and-gear/how-many-cars-are-there-in-the-world 2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8588186/

# whoami

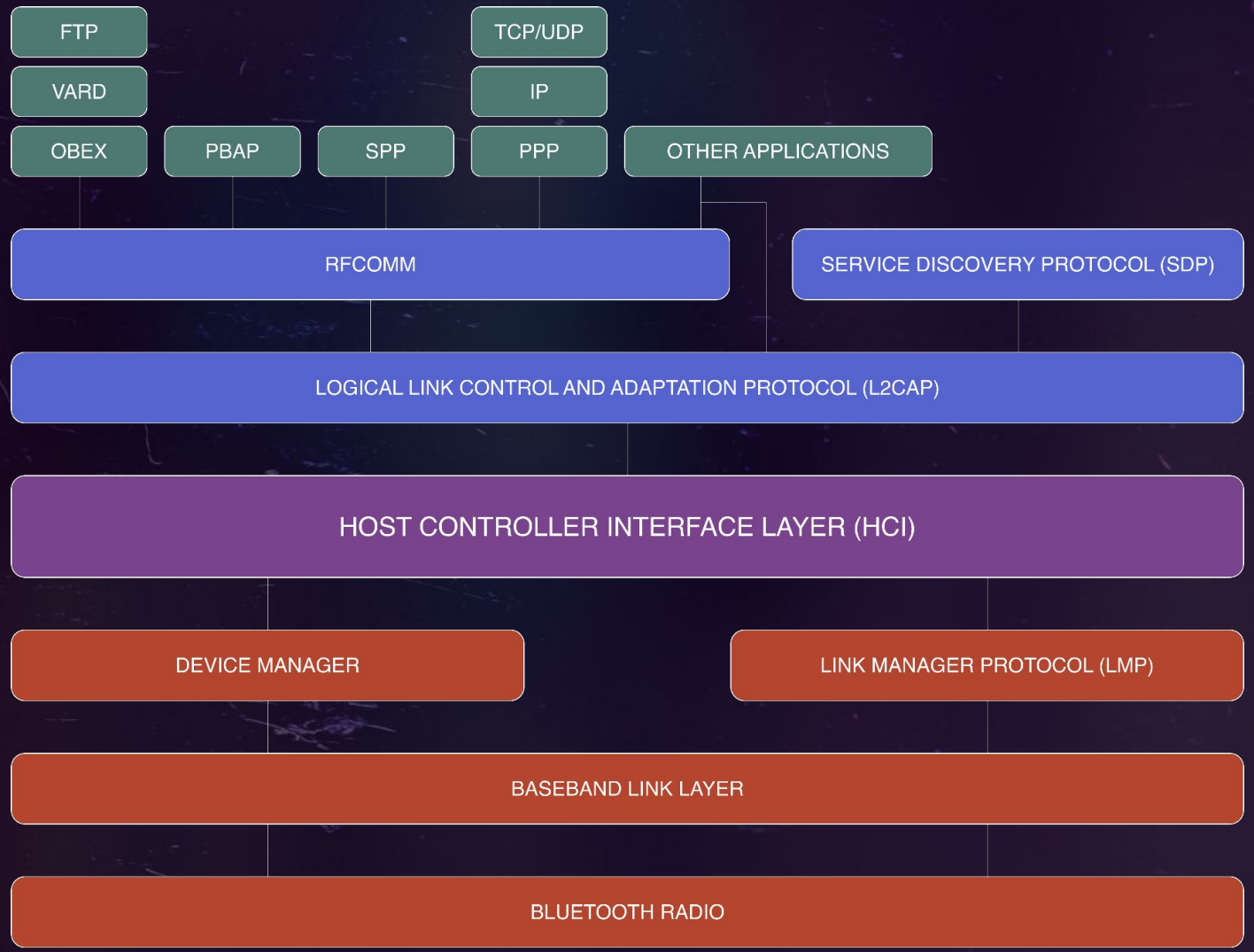
- Bug Bounty hunter (Security Researcher :) ) - known as yso, schwytz and en_de_ru_cn
- Focus areas include:
  - Penetration testing & security research from web to E-Voting and OS
  - Development/SE of tools & systems
- OSCP, BSCP;  (CRTO & CRTL in progress)
- Spoke at some confs like DEFCON, Area41 and others.

- Bluetooth project was done with Cyber Defence Campus, ETH Zurich and afterwards on my own


# Bluetooth

# II - Bluetooth - Shared responsibility

- Short-range wireless comm. Standard
- Complexity due to shared responsibility
  - errors
  - misunderstandings
- We are going to talk only about Bluetooth Classic
- Transport + Functionality




1. https://www.youtube.com/watch?v=VwM3sUYGIBk Original diagram was coloured by Kamel Ghali. We redesigned it.

# Bluetooth Classic or BLE in Vehicles

- In cars:
  - BLE supported only in a few cars Honda HFT, Toyota Corolla 2023, Tesla Model Y, Audi E-tron, …
  - BLE is generally supported on a Controller level but not on Host
  - In future, this might change
    - Bluetooth 6.0 new secure positioning* features (think keys)
    - Other use-cases or supported by OS by default.
- Bluetooth Classic is dominant in cars, e.g. there are specific profiles that are in the “Car Kit”.
- It’s easier to develop custom profiles for BLE than for BC (Bluetooth Classic)

# II - Bluetooth Profiles

- HFP
  - making calls
- A2DP
  - Transferring audio
- PBAP
  - extracting contacts
- MAP
  - extracting messages (Car kit profile)
- SAP
  - full access to the SIM card (Car kit profile)
  - Used in older cars
- FTP
  - file transfer
- HID
  - Human interface devices such as Keyboard, pointers, etc.
- Object Push, SPP and others







Icons used on this slide are taken from icon8.com

# II - Bluetooth Security Model - Capabilities


DisplayYesNo

DisplayOnly

KeyboardOnly

NoInputNoOutput

# II - Bluetooth Security Model

- “Everything or Nothing”
- Pairing modes
  - Legacy pairing
    - No SSP and SC
  - Just works
    - NiNo MitM
  - Passkey Entry
    - Method Confusion MitM
  - Numeric Comparison
    - The only secure option*

| RESPONDER \ INITIATOR | DisplayOnly | DisplayYesNo | KeyboardOnly | NoInputNoOutput |
|---|---|---|---|---|
| DisplayOnly | Just Works | Just Works | Passkey Entry | Just Works |
| DisplayYesNo | Just Works | Numeric Comparison | Passkey Entry | Just Works |
| KeyboardOnly | Passkey Entry | Passkey Entry | Passkey Entry | Just Works |
| NoInputNoOutput | Just Works | Just Works | Just Works | Just Works |

SSP - Secure simple pairing. SC - Secure Connections. * - Idea - method confusion attack on DisplayYesNo cap with Legacy Pairing instead of Passkey

# II - Bluetooth Security Model

- “Everything or Nothing”
- Pairing modes
  - Legacy pairing
    - No SSP and SC
  - Just works
    - NiNo MitM
  - Passkey Entry
    - Method Confusion MitM
  - Numeric Comparison
    - The only secure option*

| RESPONDER \ INITIATOR | DisplayOnly | DisplayYesNo | KeyboardOnly | NoInputNoOutput |
|---|---|---|---|---|
| DisplayOnly | Just Works | Just Works | Passkey Entry | Just Works |
| DisplayYesNo | Just Works | Numeric Comparison | Passkey Entry | Just Works |
| KeyboardOnly | Passkey Entry | Passkey Entry | Passkey Entry | Just Works |
| NoInputNoOutput | Just Works | Just Works | Just Works | Just Works |

SSP - Secure simple pairing. SC - Secure Connections. * - Idea - method confusion attack on DisplayYesNo cap with Legacy Pairing instead of Passkey

# II - Bluetooth Security Model

- “Everything or Nothing”
- Pairing modes
  - Legacy pairing
    - No SSP and SC
  - Just works
    - NiNo MitM
  - Passkey Entry
    - Method Confusion MitM
  - Numeric Comparison
    - The only secure option*

| RESPONDER \ INITIATOR | DisplayOnly | DisplayYesNo | KeyboardOnly | NoInputNoOutput |
|---|---|---|---|---|
| DisplayOnly | Just Works | Just Works | Passkey Entry | Just Works |
| DisplayYesNo | Just Works | Numeric Comparison | Passkey Entry | Just Works |
| KeyboardOnly | Passkey Entry | Passkey Entry | Passkey Entry | Just Works |
| NoInputNoOutput | Just Works | Just Works | Just Works | Just Works |

SSP - Secure simple pairing. SC - Secure Connections. * - Idea - method confusion attack on DisplayYesNo cap with Legacy Pairing instead of Passkey

# II - Bluetooth Security Model

- “Everything or Nothing”
- Pairing modes
  - Legacy pairing
    - No SSP and SC
  - Just works
    - NiNo MitM
  - Passkey Entry
    - Method Confusion MitM
  - Numeric Comparison
    - The only secure option*

| RESPONDER \ INITIATOR | DisplayOnly | DisplayYesNo | KeyboardOnly | NoInputNoOutput |
|---|---|---|---|---|
| DisplayOnly | Just Works | Just Works | Passkey Entry | Just Works |
| DisplayYesNo | Just Works | Numeric Comparison | Passkey Entry | Just Works |
| KeyboardOnly | Passkey Entry | Passkey Entry | Passkey Entry | Just Works |
| NoInputNoOutput | Just Works | Just Works | Just Works | Just Works |

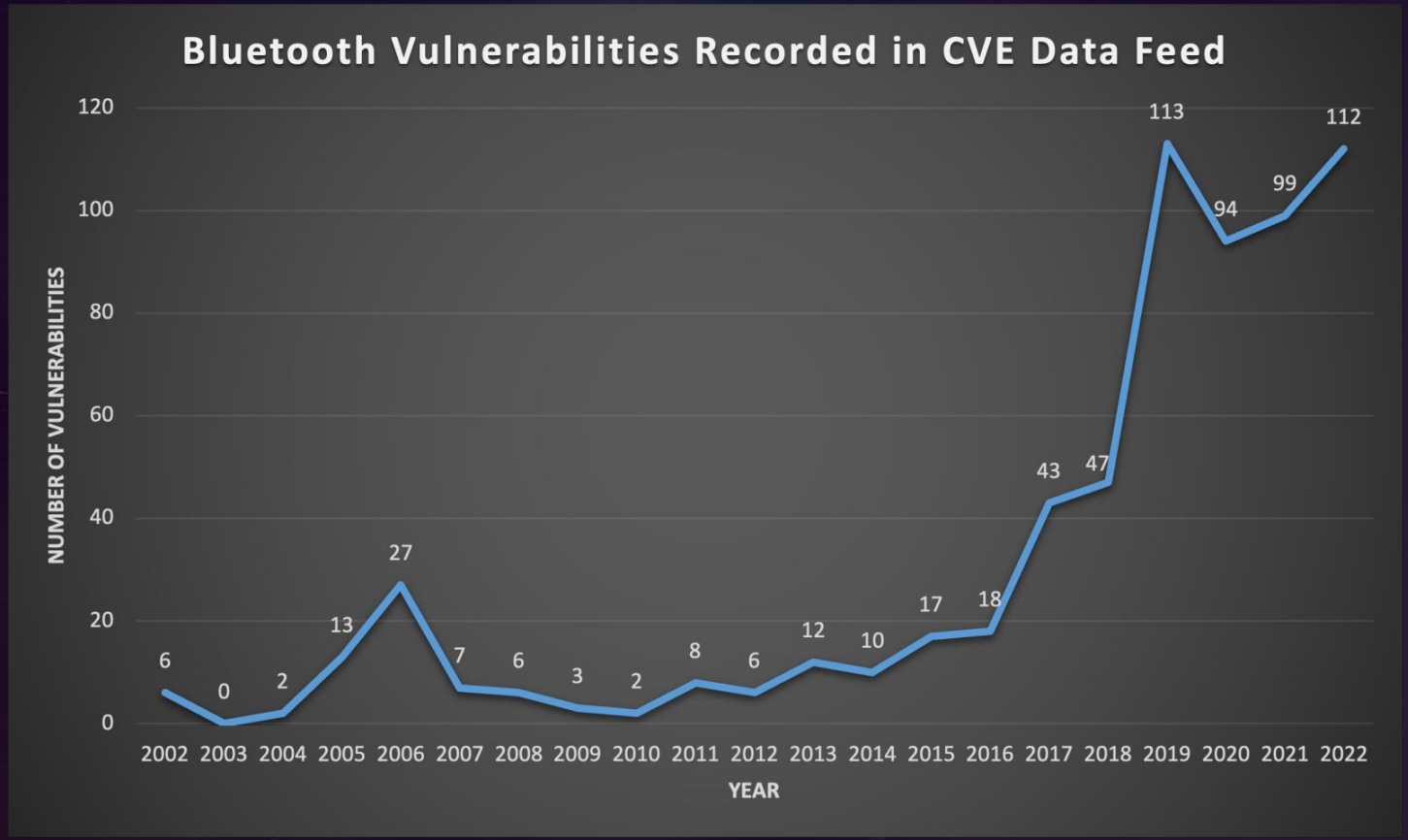
- Once established there is no difference for profiles whether the secure pairing mode was used or not

SSP - Secure simple pairing. SC - Secure Connections. * - Idea - method confusion attack on DisplayYesNo cap with Legacy Pairing instead of Passkey

# II - Previous Research

- We found more than 108 attacks on Bluetooth standards (March 2024)
- 647 CVEs related to Bluetooth devices[1] as of 2022

Source: SANS a Survey of Bluetooth Vulnerabilities Trends




1. SANS a Survey of Bluetooth Vulnerabilities Trends. 2. https://ieeexplore.ieee.org/document/10179358 3. https://github.com/TylerTucker/BluesClues

# Challenges and Toolkit

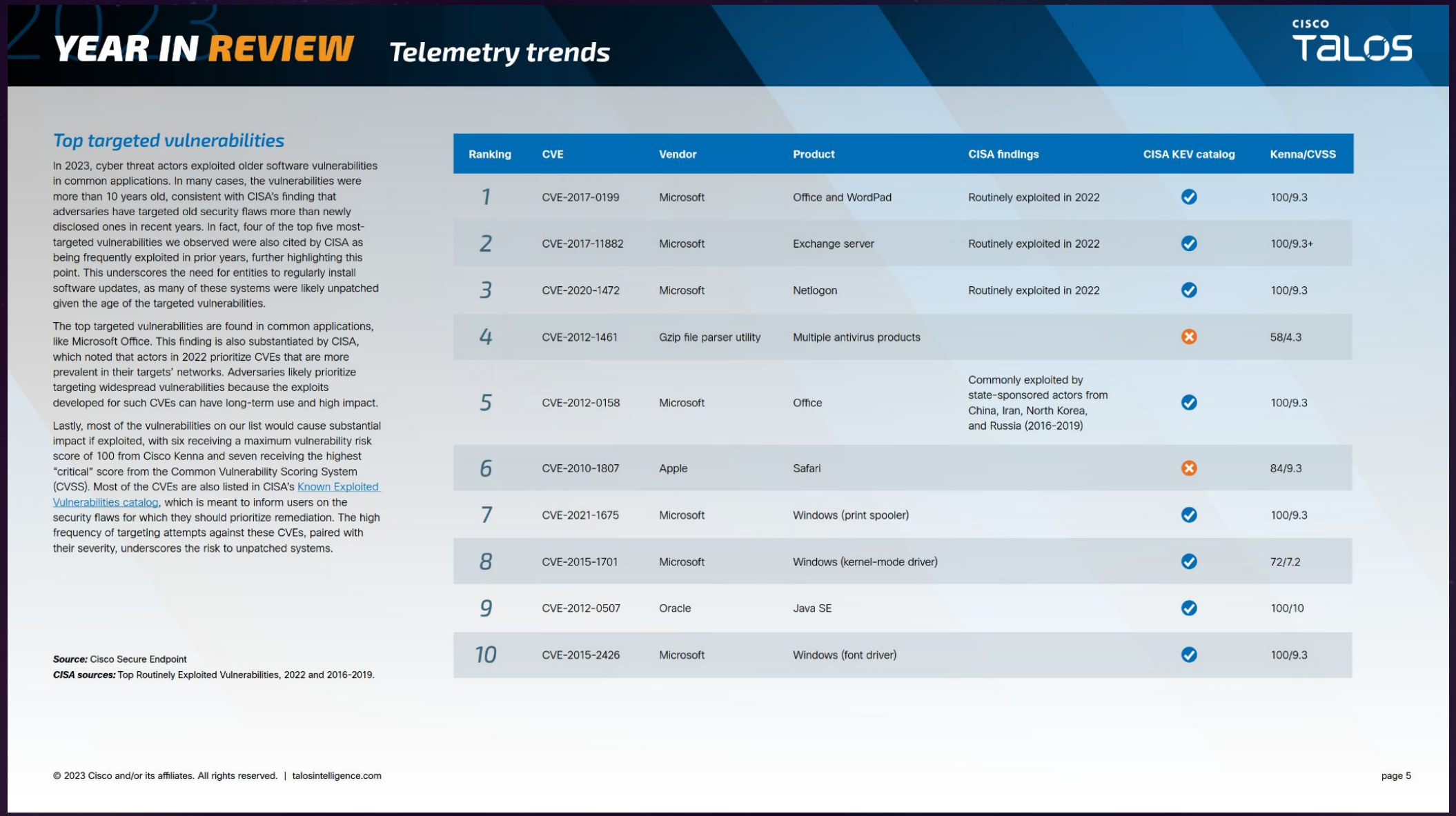
# III - BlueToolkit Why

- Bluetooth testing comes with hurdles such as:
  - No common database of vulnerabilities
  - No vulnerability testing framework
  - Finding and reconstructing PoCs from minimal public information including specification.
  - Some PoCs are based on older hardware.

- Opportunities:
  - Some manufacturers might not do regression testing because of the lack of PoC or indicators.

- Remaining issues:
  - Unpredictable issues with Bluetooth stack (Host or Controller)
  - DoS attacks might be very powerful especially with vehicles.
  - Generally impossible to fix some of the vulnerabilities without a recall (found in a specification)

# III - BlueToolkit Why



YEAR IN REVIEW Telemetry trends

cisco Talos

## Top targeted vulnerabilities

In 2023, cyber threat actors exploited older software vulnerabilities in common applications. In many cases, the vulnerabilities were more than 10 years old, consistent with CISA's finding that adversaries have targeted old security flaws more than newly disclosed ones in recent years. In fact, four of the top five most-targeted vulnerabilities we observed were also cited by CISA as being frequently exploited in prior years, further highlighting this point. This underscores the need for entities to regularly install software updates, as many of these systems were likely unpatched given the age of the targeted vulnerabilities.

The top targeted vulnerabilities are found in common applications, like Microsoft Office. This finding is also substantiated by CISA, which noted that actors in 2022 prioritize CVEs that are more prevalent in their targets' networks. Adversaries likely prioritize targeting widespread vulnerabilities because the exploits developed for such CVEs can have long-term use and high impact.

Lastly, most of the vulnerabilities on our list would cause substantial impact if exploited, with six receiving a maximum vulnerability risk score of 100 from Cisco Kenna and seven receiving the highest "critical" score from the Common Vulnerability Scoring System (CVSS). Most of the CVEs are also listed in CISA's Known Exploited Vulnerabilities catalog, which is meant to inform users on the security flaws for which they should prioritize remediation. The high frequency of targeting attempts against these CVEs, paired with their severity, underscores the risk to unpatched systems.

| Ranking | CVE | Vendor | Product | CISA findings | CISA KEV catalog | Kenna/CVSS |
|---|---|---|---|---|---|---|
| 1 | CVE-2017-0199 | Microsoft | Office and WordPad | Routinely exploited in 2022 | ✓ | 100/9.3 |
| 2 | CVE-2017-11882 | Microsoft | Exchange server | Routinely exploited in 2022 | ✓ | 100/9.3+ |
| 3 | CVE-2020-1472 | Microsoft | Netlogon | Routinely exploited in 2022 | ✓ | 100/9.3 |
| 4 | CVE-2012-1461 | Gzip file parser utility | Multiple antivirus products | | ✗ | 58/4.3 |
| 5 | CVE-2012-0158 | Microsoft | Office | Commonly exploited by state-sponsored actors from China, Iran, North Korea, and Russia (2016-2019) | ✓ | 100/9.3 |
| 6 | CVE-2010-1807 | Apple | Safari | | ✗ | 84/9.3 |
| 7 | CVE-2021-1675 | Microsoft | Windows (print spooler) | | ✓ | 100/9.3 |
| 8 | CVE-2015-1701 | Microsoft | Windows (kernel-mode driver) | | ✓ | 72/7.2 |
| 9 | CVE-2012-0507 | Oracle | Java SE | | ✓ | 100/10 |
| 10 | CVE-2015-2426 | Microsoft | Windows (font driver) | | ✓ | 100/9.3 |

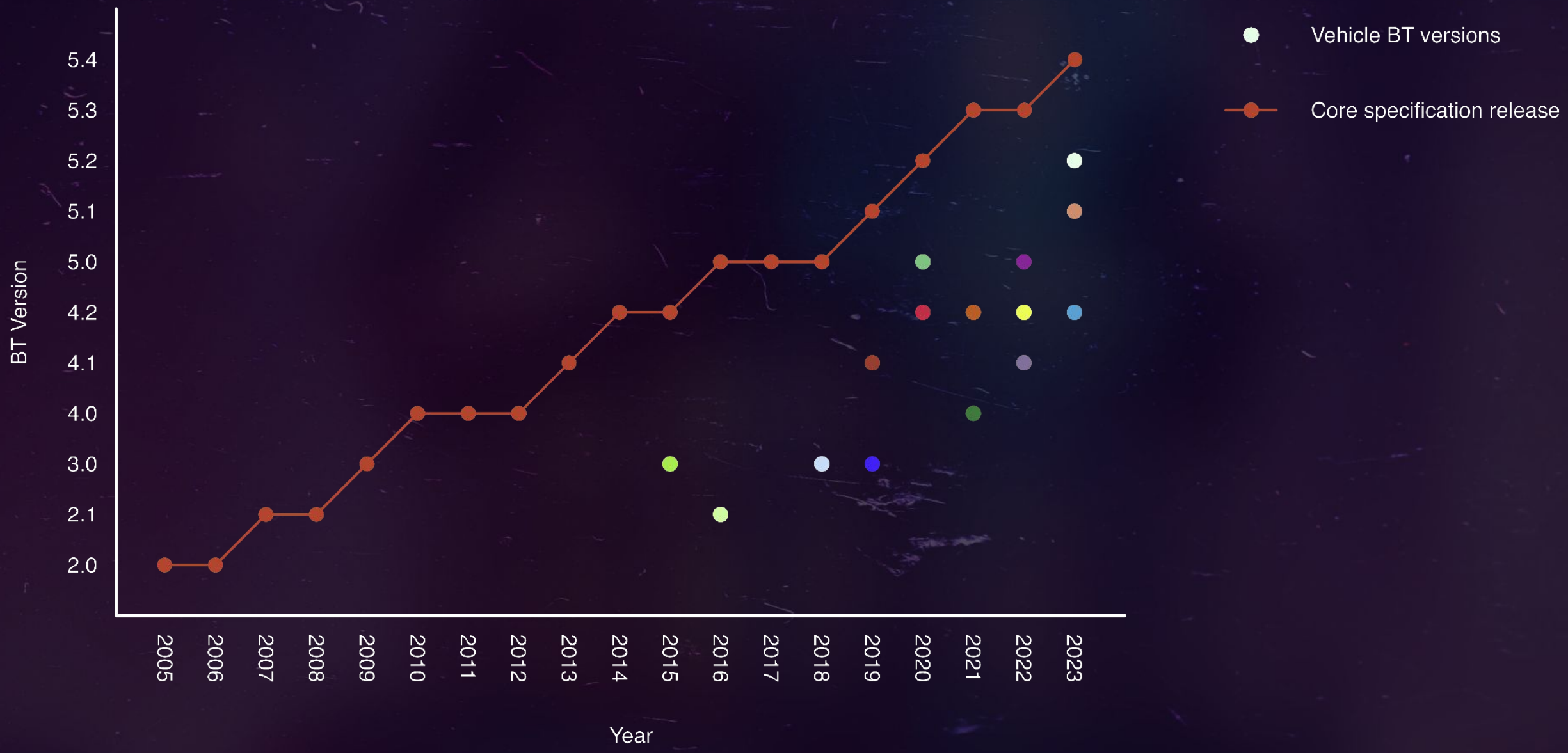
***Source:*** Cisco Secure Endpoint
***CISA sources:*** Top Routinely Exploited Vulnerabilities, 2022 and 2016-2019.





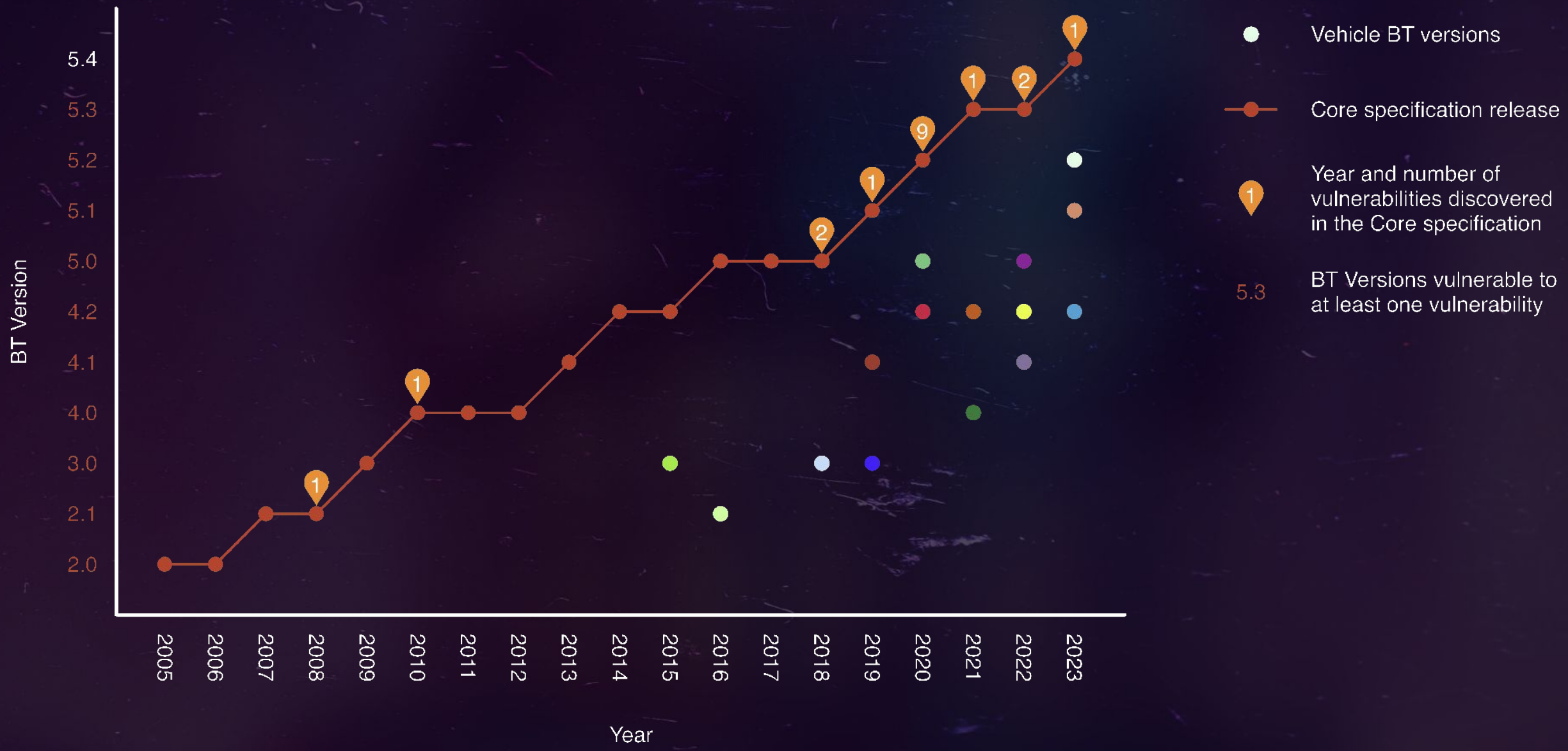
1. https://blog.talosintelligence.com/content/files/2023/12/2023_Talos_Year_In_Review.pdf

# Cars lag in adoption of the newest BT standards




Mean - 7 years, Average - 7 years. Min - 3 years, Max - 11 years. Std = 1.5 years

# IV - Cars lag in adoption of the newest BT standards



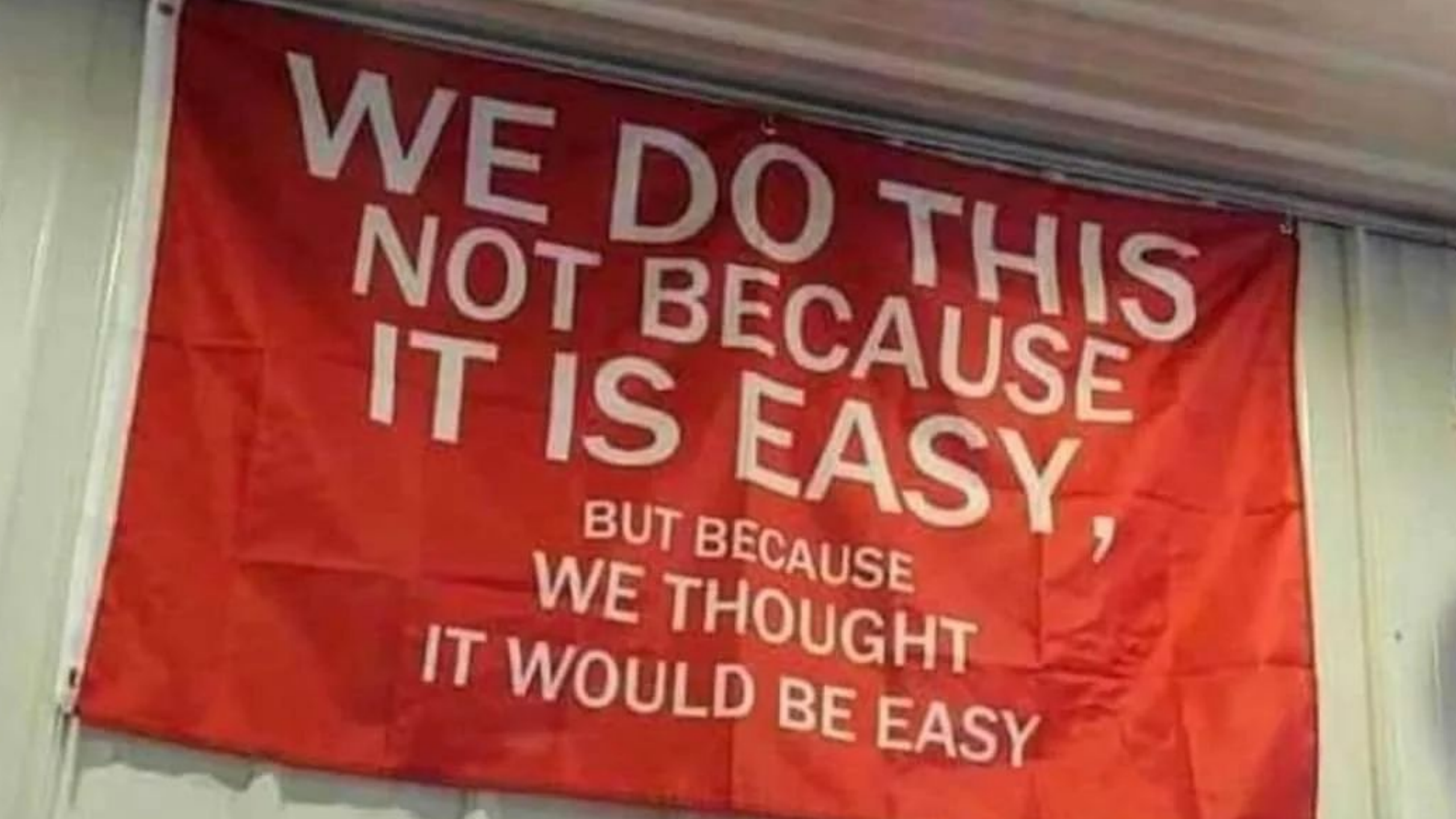

Mean - 7 years, Average - 7 years. Min - 3 years, Max - 11 years. Std = 1.5 years

# BlueToolkit - what

- Extensible Bluetooth Classic Vulnerability testing framework that helps uncover old and new vulnerabilities in Bluetooth-enabled devices.
- Think - nuclei for Bluetooth
- Useful for security and quality assurance. Could be used in procurement.
- Automates and scales security research
- At least 65 vulnerabilities discovered

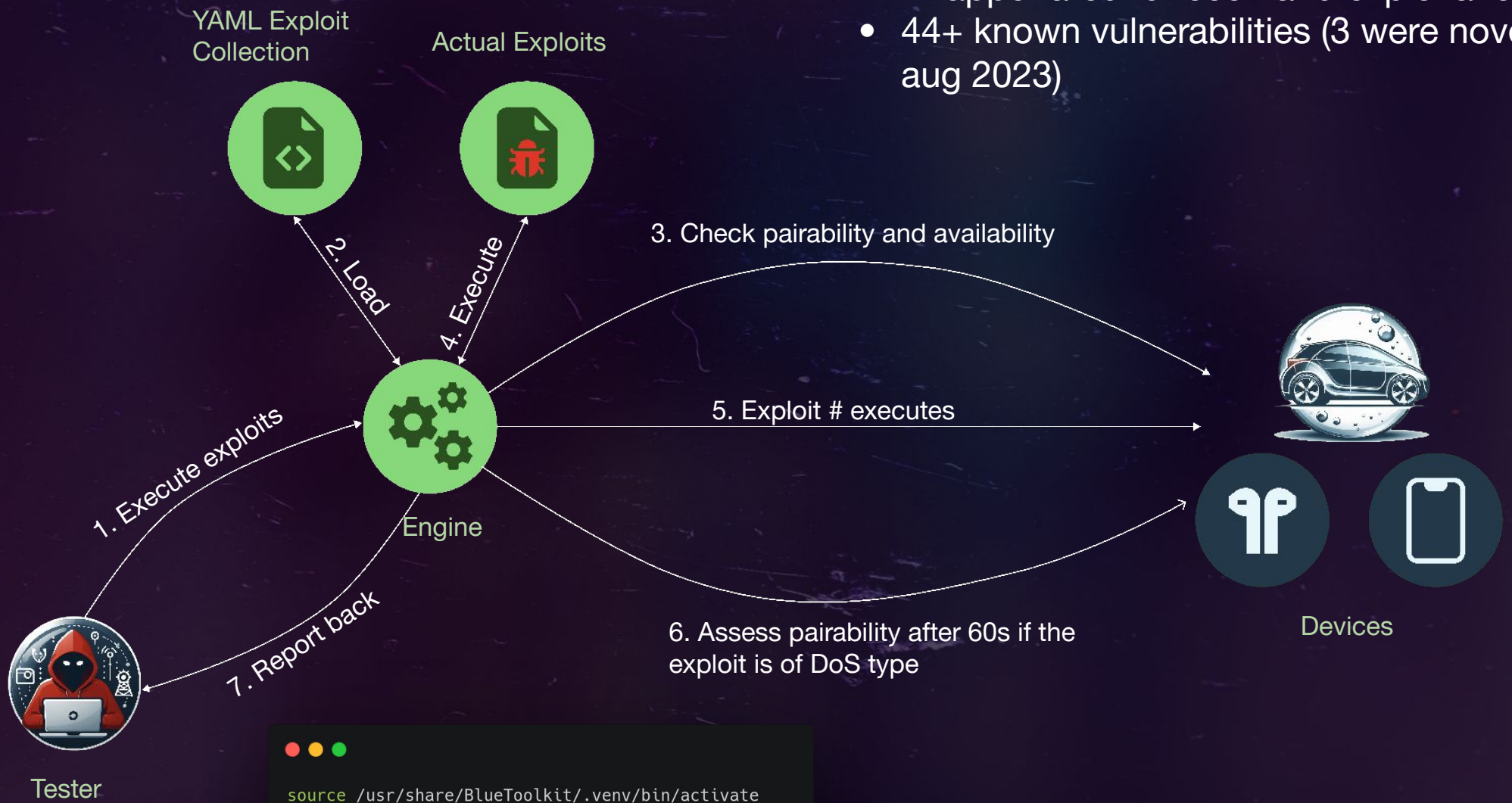

BlueToolkit

https://github.com/sgxgsx/BlueToolkit


WE DO THIS
NOT BECAUSE
IT IS EASY,
BUT BECAUSE
WE THOUGHT
IT WOULD BE EASY

# III - BlueToolkit - How

- Wrapper around recon and exploitation
- 44+ known vulnerabilities (3 were novel as of aug 2023)




```
source /usr/share/BlueToolkit/.venv/bin/activate
sudo -E env PATH=$PATH bluekit -t AA:BB:CC:DD:EE:FF
```


# Methodology

# IV - Setting up

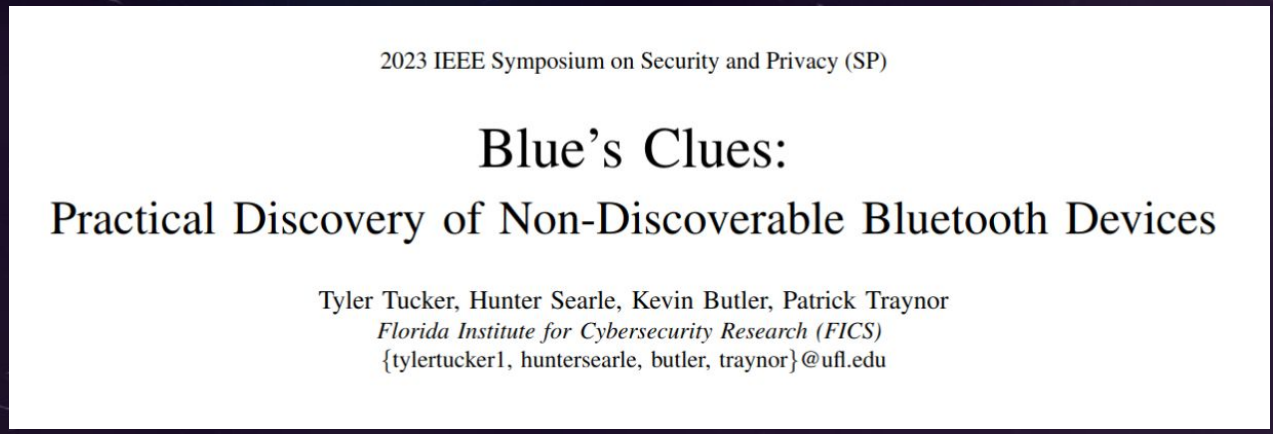
- Ubuntu-based laptop or VM
- USB cables C/Micro USB
- ESP-WROVER-KIT-VE -> $50
- Nexus 5 phone -> CYW20735 -> $50
- Ubertooth and USRP B210 (might be skipped)
- Additional hardware

# Discovering MAC addresses

- For our own tests
  - Run bluetoothctl
  - Ensure that IVI is waiting for pairing

```
some:~$ bluetoothctl --agent=NoInputNoOutput
[bluetooth]# scan on
```

- In case of the simulated* attack:
  - The 1st technique will work for some cars (always pairable and discoverable)
  - More costly solution is to use Blue's Clues[1][2] attack (needs Ubertooth and USRP B210) but will discover full MAC addresses even if the device is in non-discoverable mode.



2023 IEEE Symposium on Security and Privacy (SP)

Blue's Clues:

Practical Discovery of Non-Discoverable Bluetooth Devices

Tyler Tucker, Hunter Searle, Kevin Butler, Patrick Traynor
*Florida Institute for Cybersecurity Research (FICS)*
{tylertucker1, huntersearle, butler, traynor}@ufl.edu

1. https://ieeexplore.ieee.org/document/10179358 2. https://github.com/TylerTucker/BluesClues

# Recon

- Collect more general information on the target
  - Device name
  - manufacturer
  - LMP features (supported features)
  - SDP information (services)
- With some devices you need to pair first to get SDP info
  - Tesla Model Y ~(version) Bluetooth stack crashes on this command -_-

```
sudo -E env PATH=$PATH bluekit -t AA:BB:CC:DD:EE:FF -r
```

=

```
sudo hcitool info AA:BB:CC:DD:EE:FF
sudo sdptool browse AA:BB:CC:DD:EE:FF
sudo bluing br --sdp AA:BB:CC:DD:EE:FF
sudo bluing br --lmp-features AA:BB:CC:DD:EE:FF
```

# Recon

```
Service Record
0x0000: ServiceRecordHandle (uint32)
    0x00010003
0x0001: ServiceClassIDList (sequence)
    0x110b: AudioSink
0x0004: ProtocolDescriptorList (sequence)
    0x0100: L2CAP
        PSM: 0x0019
    0x0019: AVDTP
        v1.3
0x0005: BrowseGroupList (sequence)
    0x1002: PublicBrowseRoot
0x0009: BluetoothProfileDescriptorList (sequence)
    0x110d: AdvancedAudioDistribution v1.3
0x0100: ServiceName (guess) (text)
    SWN05LGHA2DP013
0x0311: unknown
    <uint16 value="0x0001" />


Service Record
0x0000: ServiceRecordHandle (uint32)
    0x00010004
0x0001: ServiceClassIDList (sequence)
    0x112e: Phonebook Access - PCE
0x0005: BrowseGroupList (sequence)
    0x1002: PublicBrowseRoot
0x0009: BluetoothProfileDescriptorList (sequence)
    0x1130: Phonebook Access v1.2
0x0100: ServiceName (guess) (text)
    PBAP PCE
```

```
Requesting information ...
    BD Address:  AA:BB:CC:DD:EE:FF
    OUI Company: LG Innotek (CC-88-26)
    Device Name: Kona
    LMP Version: 5.0 (0x9) LMP Subversion: 0x420e
    Manufacturer: Broadcom Corporation (15)
```

```
LE Supported (Controller): True
Simultaneous LE and BR/EDR to Same Device Capable (Controller): True
LE Supported (Host): False
Simultaneous LE and BR/EDR to Same Device Capable (Host): False
```

# Sniffing

- Difficult and costly to sniff Bluetooth Classic
- Impractical due to encryption deployment since 2013
- If you want to dump on LMP level
  - use InternalBlue or Braktooth
- What you actually might need:
  - is to dump traffic between you and your test device (only on HCI level)
  - You can produce a log on a target device too

# Testing for known vulnerabilities or regression testing

- BlueToolkit is your helper!
- 44 tests in total (6 are manual)
- Simply run all available to you exploits with the following command.
- Observe the output and look at the IVI

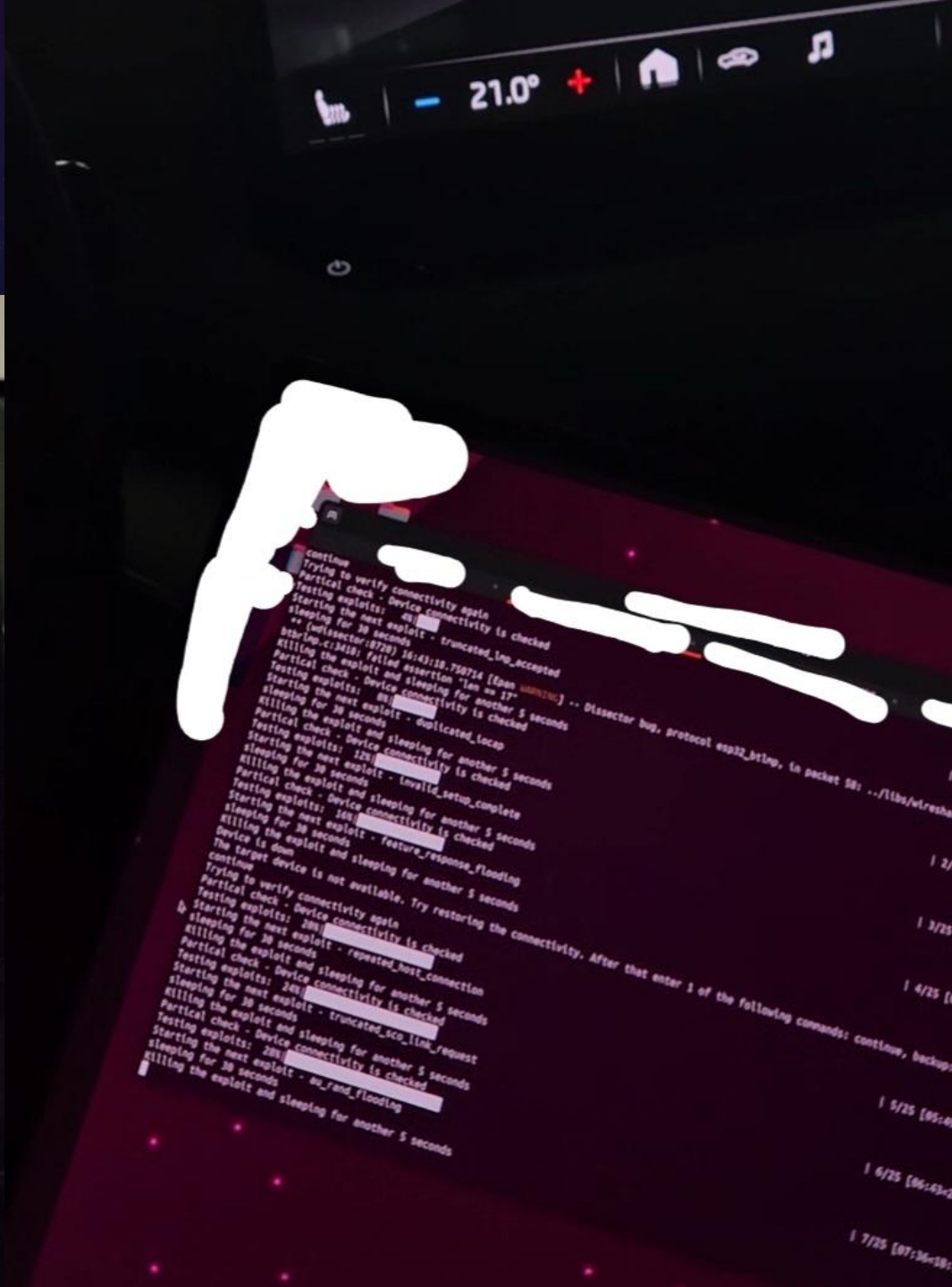
```
source /usr/share/BlueToolkit/.venv/bin/activate
sudo -E env PATH=$PATH bluekit -t AA:BB:CC:DD:EE:FF
```

```
some@some:/usr/share/Bluetoolkit$ source /usr/share/Bluetoolkit/venv/bin/activate
(.venv) some@some:/usr/share/Bluetoolkit$ sudo -E env PATH=$PATH bluekit -l
```

| Index | Exploit | Type | Hardware | Available | BT min | BT max |
|---|---|---|---|---|---|---|
| 1 | bleedingtooth_badchoice_cve_2020_12352 | DoS | default | ✅ | 1.0 | 5.4 |
| 2 | blueborne_CVE_2017_1000250 | DoS | default | ✅ | 2.0 | 5.2 |
| 3 | bleedingtooth_badkarma_cve_2020_12351 | DoS | default | ✅ | 5.0 | 5.2 |
| 4 | bleedingtooth_badvibes_cve_2020_24490 | DoS | default | ✅ | 1.0 | 5.2 |
| 5 | blueborne_CVE_2017_1000251 | DoS | default | ✅ | 2.0 | 5.2 |
| 6 | reconnaissance_SSP_supported | PoC | default | ✅ | 1.0 | 5.4 |
| 7 | blueborne_CVE_2017_0785 | PoC | default | ✅ | 2.0 | 5.2 |
| 8 | custom_nino_check | PoC | default | ✅ | 1.0 | 5.4 |
| 9 | custom_legacy_pairing_second_check | PoC | default | ✅ | 1.0 | 5.4 |
| 10 | reconnaissance_possible_BLUR | PoC | default | ✅ | 1.0 | 5.4 |
| 11 | reconnaissance_SC_supported | PoC | default | ✅ | 1.0 | 5.4 |
| 12 | custom_method_confusion_check | PoC | default | ✅ | 2.1 | 5.4 |
| 13 | truncated_lmp_accepted | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 14 | repeated_host_connection | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 15 | sdp_oversized_element_size | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 16 | duplicated_encapsulated_payload | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 17 | wrong_encapsulated_payload | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 18 | feature_req_ping_pong | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 19 | sdp_unkown_element_type | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 20 | lmp_auto_rate_overflow | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 21 | lmp_overflow_2dh1 | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 22 | truncated_sco_link_request | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 23 | invalid_timing_accuracy | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 24 | invalid_max_slot | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 25 | lmp_invalid_transport | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 26 | paging_scan_disable | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 27 | duplicated_iocap | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 28 | lmp_overflow_dm1 | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 29 | invalid_feature_page_execution | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 30 | lmp_max_slot_overflow | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 31 | feature_response_flooding | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 32 | au_rand_flooding | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 33 | invalid_setup_complete | DoS | esp32 | ❌ | 2.0 | 5.4 |
| 34 | braktooth_knob | PoC | esp32 | ❌ | 2.0 | 5.4 |
| 35 | internalblue_CVE_2018_19860_0a_00 | DoS | nexus5 | ❌ | 2.0 | 5.2 |
| 36 | internalblue_CVE_2018_19860_20_17 | DoS | nexus5 | ❌ | 2.0 | 5.2 |
| 37 | internalblue_CVE_2018_19860_16_0b | DoS | nexus5 | ❌ | 2.0 | 5.2 |
| 38 | internalblue_CVE_2018_5383_Invalid_second | Manual | nexus5 | ❌ | 2.0 | 5.2 |
| 39 | internalblue_knob | PoC | nexus5 | ❌ | 2.0 | 5.2 |

# Testing for known vulnerabilities

- 2 options: accept or decline (try both)

# Testing for known vulnerabilities

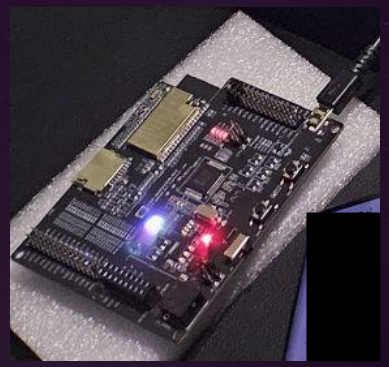
- If you see smth like this - either there is a deadlock or device DoS itself.
- You can try to “continue” and check the availability and pairability again.

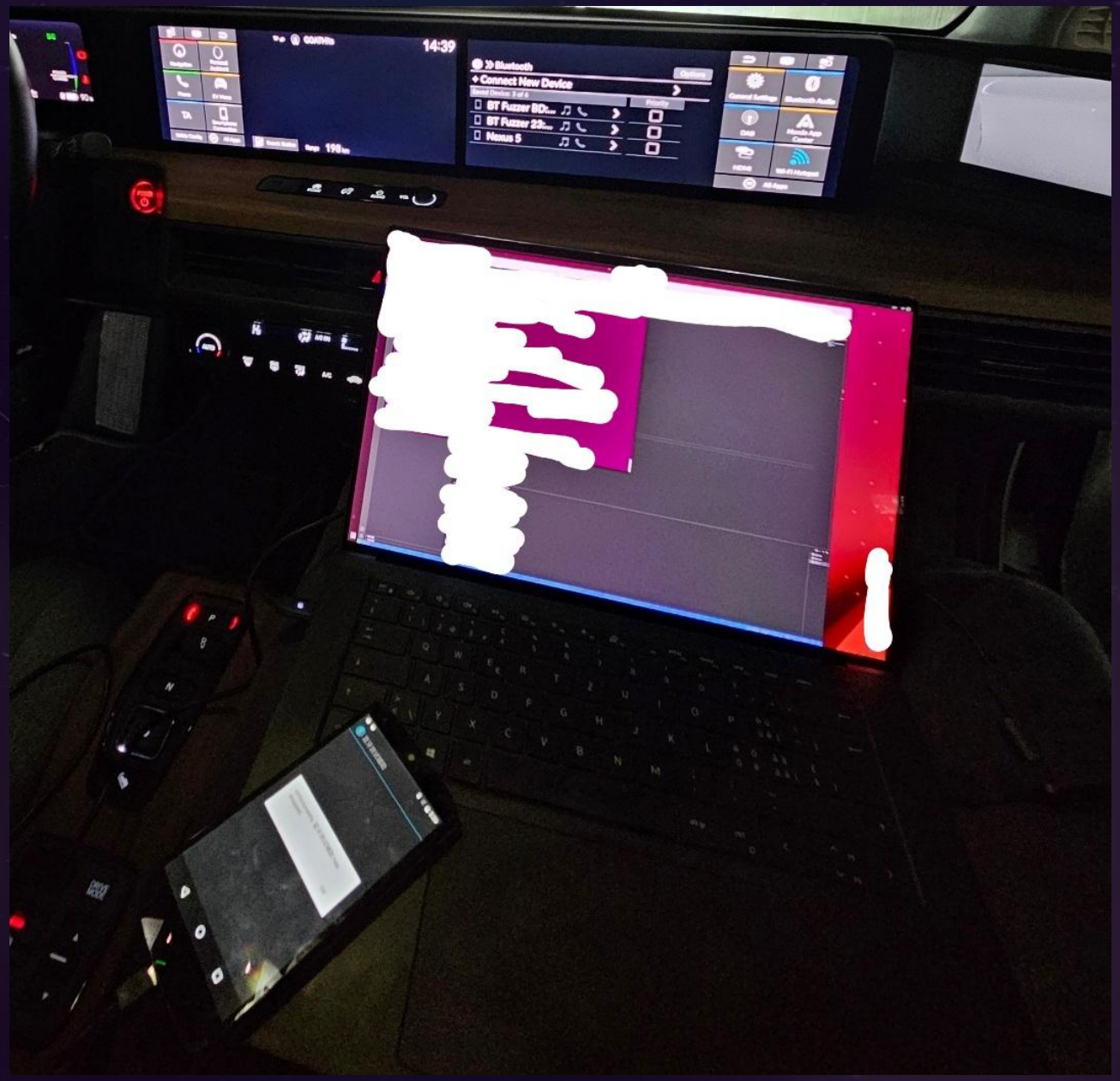
```
Device is down
Device is down
Device is down
Device is down
Device is down
Device is down
Device is down
Device is down
Device is down
Device is down
The target device is not available. Try restoring the connectivity. After that enter 1 of the following
commands: continue, backup:
```

# Testing for known vulnerabilities

# Testing for known vulnerabilities

- Once done:
  - Add hardware
  - ESP-WROVER-KIT-VE
    - DoS attacks
  - Nexus 5 phone
    - Internalblue attacks
  - Other hardware could be available once added





# Testing for known vulnerabilities ESP-WROVER-KIT-VE

- Enables us to run 22 tests
- You need to setup it on the first run (automated[3] for you)

```
ls -la /dev/tty*
chmod +x /usr/share/Btoolkit/installation/braktooth_additional_install.sh
/usr/share/Btoolkit/installation/braktooth_additional_install.sh
```

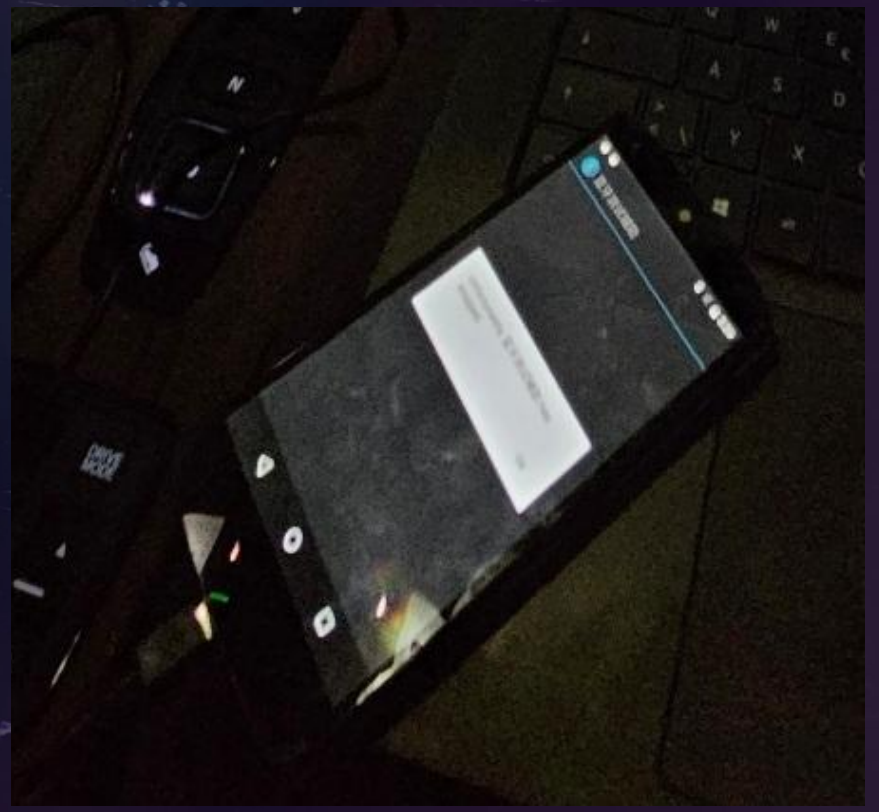
- Not all tests are described as ones leading to vulnerabilities as per Braktooth[1,2] authors
- You don’t need to pair on your target device while testing

1. https://asset-group.github.io/disclosures/braktooth/ 2. https://github.com/Matheus-Garbelini/braktooth_esp32_bluetooth_classic_attacks 3. https://github.com/sgxgsx/BlueToolkit/wiki/Using-hardware%E2%80%90based-exploits#braktooth

# Testing for known vulnerabilities Nexus 5

- Enables us to run 5 tests
- Setup is a bit more challenging
  - Root the phone and follow Internalblue instructions
- Install helper Android tool

```
adb devices
chmod +x /usr/share/Btoolkit/installation/bluetoothassistant_additional_install.sh
/usr/share/Btoolkit/installation/bluetoothassistant_additional_install.sh
```



1. https://github.com/sgxgsx/BlueToolkit/wiki/Using-hardware%E2%80%90based-exploits#internalblue 2. https://github.com/seemoo-lab/internalblue/blob/master/doc/android.md

# Testing for known vulnerabilities - report

- You might need to take notes during the tests, just in case the tool reports false positives (DoS)
  - In the end it's a PoC

```
sudo -E env PATH=$PATH bluekit -t AA:BB:CC:DD:EE:FF -re
```

```
(.venv) some@some:/usr/share/Bluetoolkit$ sudo -E env PATH=$PATH bluekit -t AA:BB:CC:DD:EE:FF -re
+-------+-----------------------------------------+----------------
+-------------------------------------------------------------------------------------------+
| Index | Exploit                                 | Result         | Data
        |
+-------+-----------------------------------------+----------------
+-------------------------------------------------------------------------------------------+
|   1   | au_rand_flooding                        | Not vulnerable | 3
        |
|   2   | feature_response_flooding               | Not vulnerable | 3
        |
|   3   | braktooth_knob                          | Not vulnerable | not vulnerable
        |
|   4   | feature_req_ping_pong                   | Not vulnerable | 1
        |
|   5   | reconnaissance_SC_supported             | Vulnerable!    | No Secure Connections supported
        |
|   6   | reconnaissance_possible_BLUR            | Not vulnerable | No LE supported, Cross transport attacks are not going to work
        |
|   7   | reconnaissance_SSP_supported            | Vulnerable!    | SSP supported, secure cryptography is used, there might be a
problem with Messag |
|   8   | bleedingtooth_badkarma_cve_2020_12351   | Not vulnerable | 0
        |
|   9   | bleedingtooth_badchoice_cve_2020_12352  | Not vulnerable | 0
        |
|  10   | bleedingtooth_badvibes_cve_2020_24490   | Not vulnerable | 0
        |
|  11   | paging_scan_disable                     | Not vulnerable | 1
        |
|  12   | wrong_encapsulated_payload              | Not vulnerable | 0
        |
|  13   | duplicated_iocap                        | Not vulnerable | 0
        |
|  14   | lmp_overflow_2dh1                       | Not vulnerable | 0
        |
|  15   | lmp_overflow_dm1                        | Not vulnerable | 2
        |
|  16   | lmp_auto_rate_overflow                  | Not vulnerable | 2
        |
|  17   | sdp_oversized_element_size              | Not vulnerable | 4
        |
|  18   | lmp_invalid_transport                   | Not vulnerable | 4
        |
|  19   | lmp_max_slot_overflow                   | Not vulnerable | 3
        |
|  20   | repeated_host_connection                | Not vulnerable | 0
        |
|  21   | duplicated_encapsulated_payload         | Not vulnerable | 5
        |
|  22   | sdp_unkown_element_type                 | Not vulnerable | 1
        |
|  23   | custom_method_confusion_check           | Not vulnerable | Device uses DisplayYesNo
        |
|  24   | custom_legacy_pairing_second_check      | Not vulnerable | Legacy pairing is not enabled
        |
|  25   | custom_nino_check                       | Vulnerable!    | vulnerable
        |
|  26   | internalblue_CVE_2018_19860_16_0b       | Not vulnerable | 0
        |
|  27   | internalblue_CVE_2018_19860_20_17       | Not vulnerable | 0
        |
|  28   | internalblue_CVE_2018_5383_Invalid_second | Not vulnerable | Was not able to establish a connection
        |
|  29   | internalblue_knob                       | Not vulnerable | not vulnerable
        |
|  30   | internalblue_CVE_2018_19860_0a_00       | Not vulnerable | 0
        |
|  31   | blueborne_CVE_2017_1000250              | Not vulnerable | 0
        |
|  32   | truncated_sco_link_request              | Not vulnerable | 4
        |
|  33   | truncated_lmp_accepted                  | Not vulnerable | 2
        |
|  34   | blueborne_CVE_2017_0785                 | Not vulnerable | Target device OS is not Android
        |
|  35   | blueborne_CVE_2017_1000251              | Not vulnerable | 0
        |
|  36   | invalid_setup_complete                  | Not vulnerable | 0
        |
|  37   | invalid_feature_page_execution          | Not vulnerable | 0
        |
|  38   | invalid_max_slot                        | Vulnerable!    | 10
        |
|  39   | invalid_timing_accuracy                 | Not vulnerable | 4
        |
+-------+-----------------------------------------+----------------
+-------------------------------------------------------------------------------------------+
```

1. https://asset-group.github.io/disclosures/braktooth/ 2. https://github.com/Matheus-Garbelini/braktooth_esp32_bluetooth_classic_attacks

# Looking for design- and state-based vulnerabilities

- MitM or BAC issues
- Idea:
  - MitM: Look for vulnerabilities that allow you to connect to a target device without a confirmation or with a predictable confirmation
  - BAC: Look for vulnerabilities that assume you are authorized to get a specific functionality on a target device

1. https://asset-group.github.io/disclosures/braktooth/ 2. https://github.com/Matheus-Garbelini/braktooth_esp32_bluetooth_classic_attacks

# Looking for design- and state-based vulnerabilities: MitM

- Change your device type
- Change your device capabilities
- Simply look for a violation of a specification

```
some:~$ bt-agent -c NoInputNoOutput
some:~$ bluetoothctl --agent=NoInputNoOutput
[bluetooth]# scan on
[bluetooth]# connect B_MAC
```

```
some:~$ bluetoothctl --agent=DisplayYesNo
some:~$ bluetoothctl --agent=DisplayOnly
some:~$ bluetoothctl --agent=KeyboardOnly
some:~$ bluetoothctl --agent=NoInputNoOutput
```

```
sudo hciconfig hci0 class 0x5a020c
sudo hciconfig hci0 name 'name'
```

1. https://asset-group.github.io/disclosures/braktooth/ 2. https://github.com/Matheus-Garbelini/braktooth_esp32_bluetooth_classic_attacks

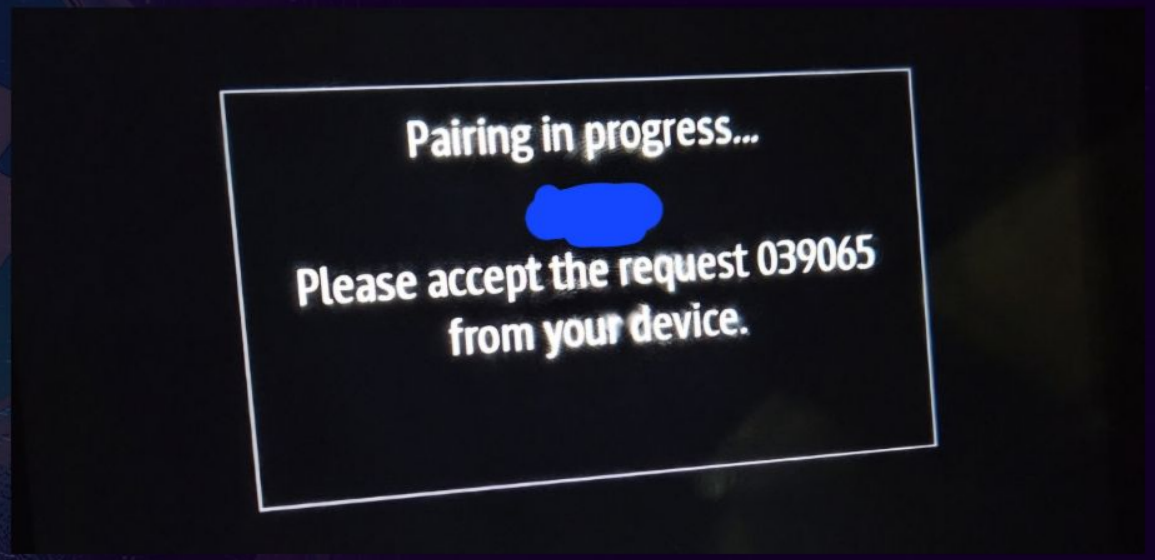
# Looking for design- and state-based vulnerabilities: MitM

- + Force legacy pairing (4 digit static number)
  - Doesn’t always work. Probably there is a race condition that allows us to turn off SC and SSP
  - Should work on older devices

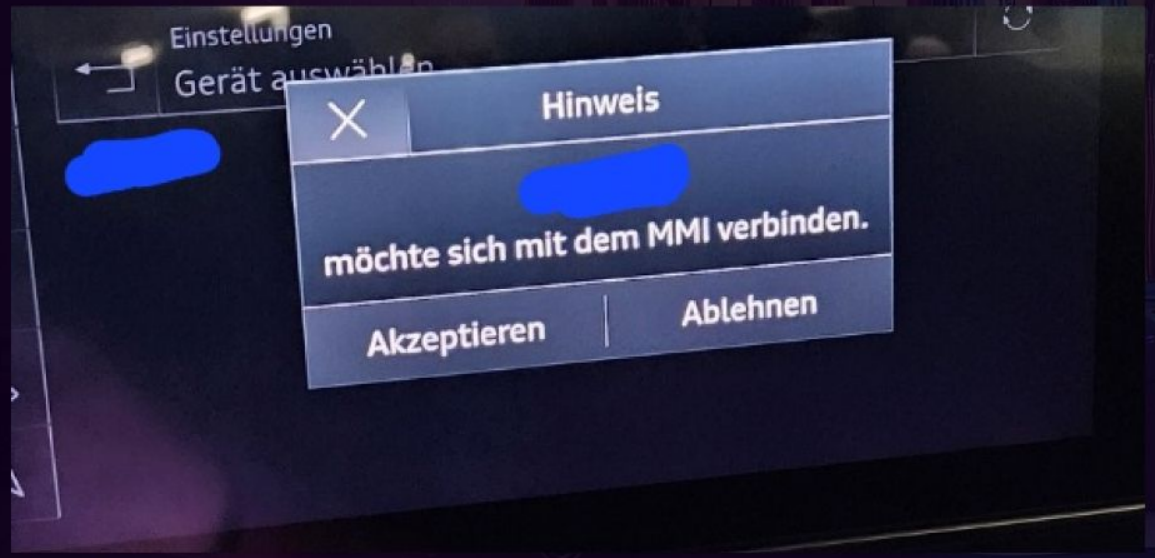
```
while true; do sudo btmgmt ssp off; sudo btmgmt sc off; done
```

# Looking for design- and state-based vulnerabilities: MitM Example

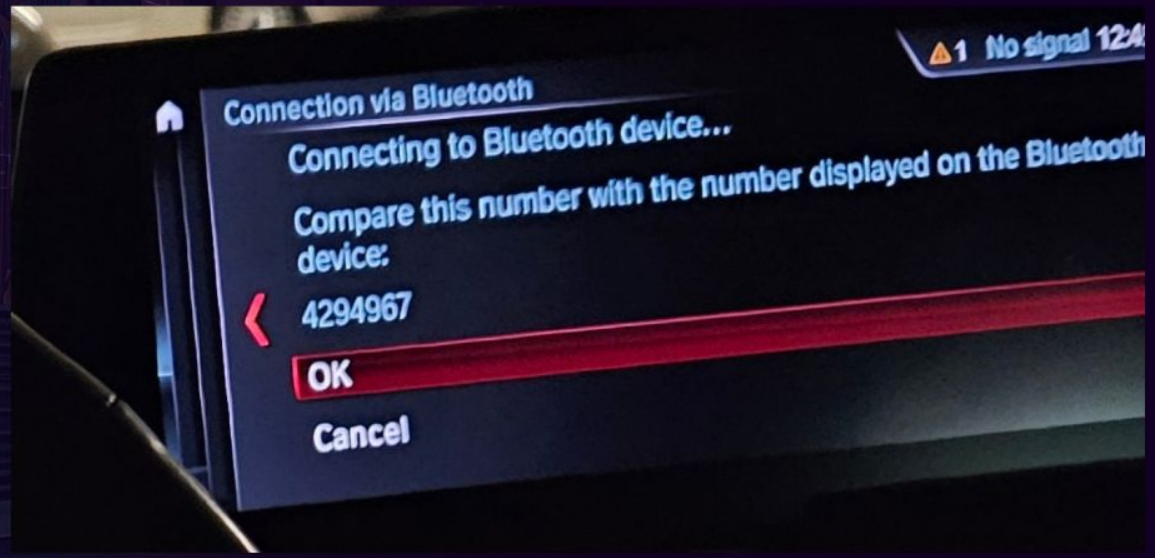

Hyundai




Renault




Audi




BMW

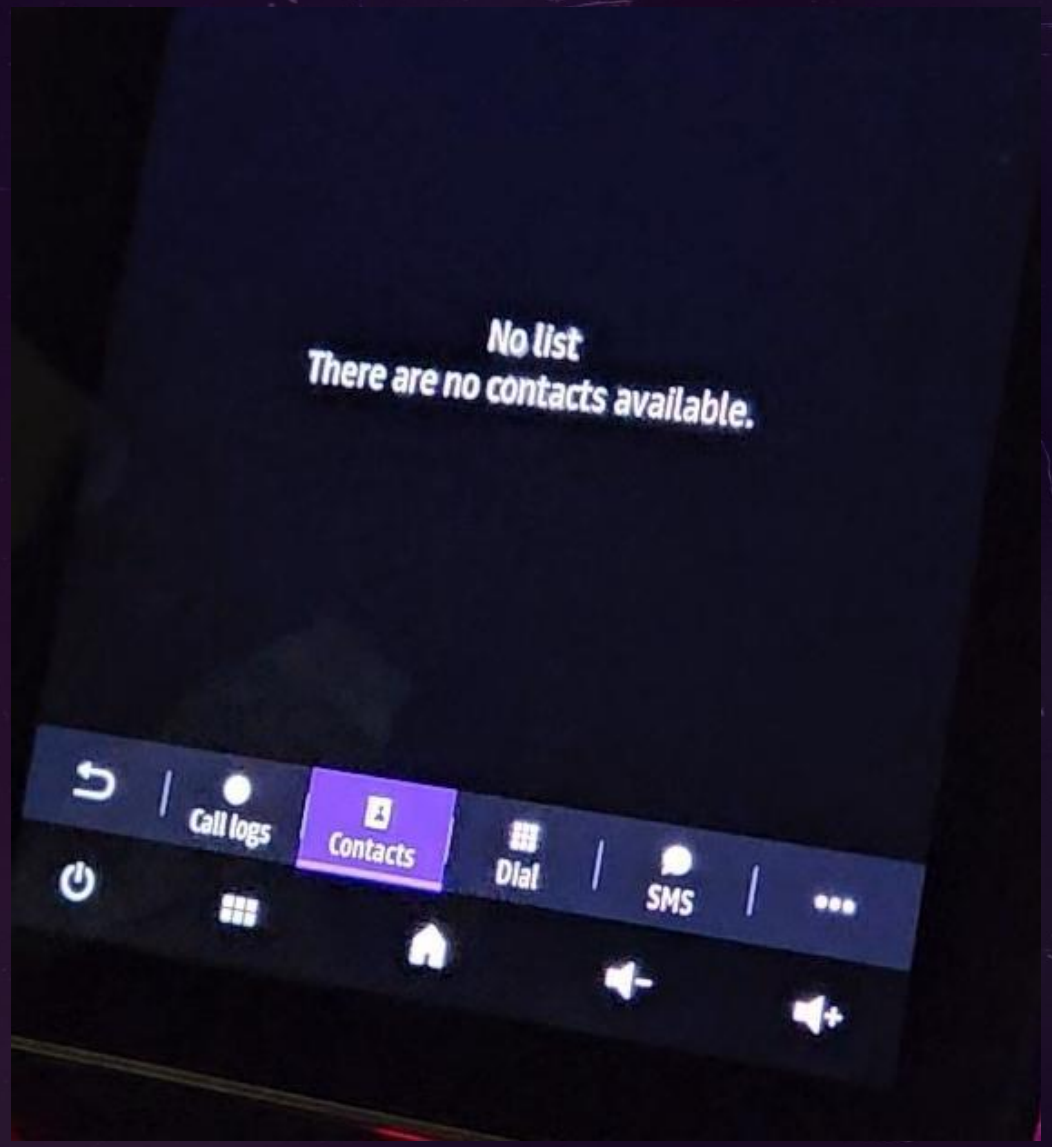
# Looking for design- and state-based vulnerabilities: BAC

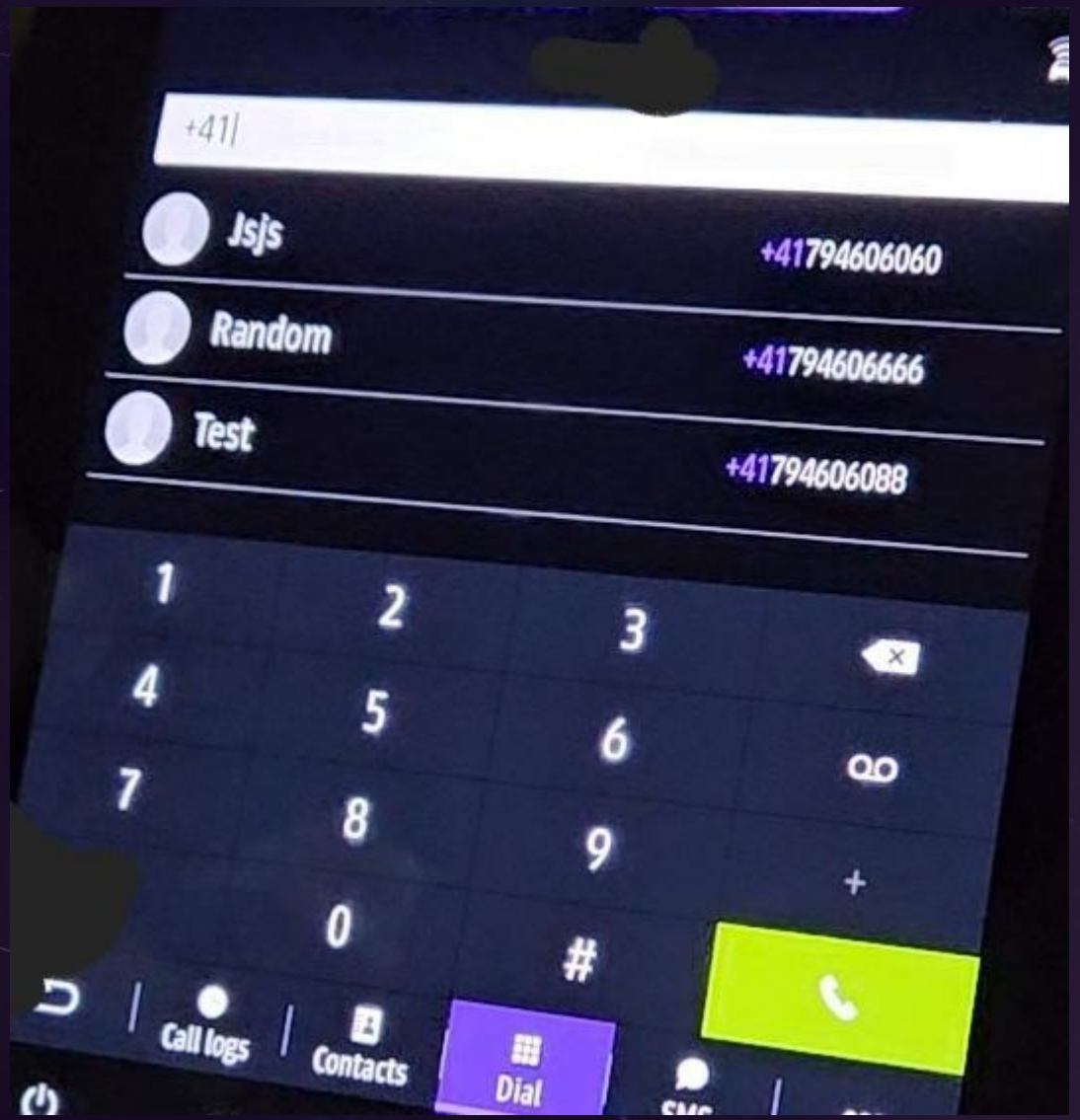
- Change your device type as in a previous slide
- Change your MAC address

```
sudo bdaddr -i hci0 FF:EE:DD:CC:BB:AA
sudo hciconfig hci0 resetw
sudo systemctl restart bluetooth.service
```

1. https://asset-group.github.io/disclosures/braktooth/ 2. https://github.com/Matheus-Garbelini/braktooth_esp32_bluetooth_classic_attacks

# Looking for design- and state-based vulnerabilities: BAC



# Troubleshooting problems and some tips

- BlueZ doesn’t see a MAC address or (loses it)

```
while true; do sudo hcitool info AA:BB:CC:DD:EE:FF; done
```

- Making sure that we don’t have an established pairing to a target device (delete an LTK/LinkKey)
  - Or you want to extract an LTK

```
some@some: sudo su
root@some: cd /var/lib/bluetooth/[YOUR MAC ADDR]/[TARGET MAC ADDR]
root@some: cd /var/lib/bluetooth/AA:BB:CC:DD:EE:FF/FF:EE:DD:CC:BB:AA
root@some: ls
attributes  info
root@some: cat info
[General]
Name=Polestar PS2
Class=0x260408
SupportedTechnologies=BR/EDR;
Trusted=false
Blocked=false
Services=UUIDS;........;
[DeviceID]
Source=1
Vendor=200
Product=4000
Version=5000

[LinkKey]
Key=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Type=5
PINLength=0
```

# Testing Profiles

- All cars have:
  - A2DP, AVCTP, AVDTP
  - HFP
  - PBAP (client)
  - MAP (Notification service)
- Older vehicles also have
  - SAP
  - Object push profile
  - FTP
- Apart from implementation we can test how these profiles save data and whether we can retrieve it without knowing LTK. Think about BAC
- FTP and Object push profile are of high interest for older vehicles
  - might get you access to a file system or RCE

# Redeveloping CVEs

- Why
  - If a vulnerability has been found in one stack it might come up in another one. (BlueZ of particular interest)
- Where to find CVEs:
  - https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=bluetooth
  - https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=bluez
- Where to find new attacks on Bluetooth itself:
  - Papers & Conf: ACM, IEEE SP, Blackhat, usenix, NDSS, Google scholar
  - Github, Google, Bing, …
  - https://darkmentor.com/bt.html - Great resource and timeline
  - https://github.com/sgxgsx/BlueToolkit - Description also has a table with other BLE and BT vulnerabilities
- Add your 1-days to BlueToolkit so that others can test them too!
  - Reach out to me and I can help writing a YAML profile

# Extra mile

- CTKD (Cross transport key derivation)
  - you may impact Bluetooth Classic via an attack on BLE e.g. get a LTK for BLE connection and then rewrite link key for Bluetooth Classic one to establish MitM
  - https://github.com/francozappa/blur
- Fuzzing
  - https://github.com/Matheus-Garbelini/braktooth_esp32_bluetooth_classic_attacks
  - Can use it similar to Internalblue for your attacks.
- Documentation & Specification
  - https://www.bluetooth.com/specifications/specs/
  - It’s not that big and you don’t need to read 3000+ pages. Get the basics and then review what you are interested in
    - Same approach as in Code review (instead of `grep` we have Ctrl+F)

# Extra mile

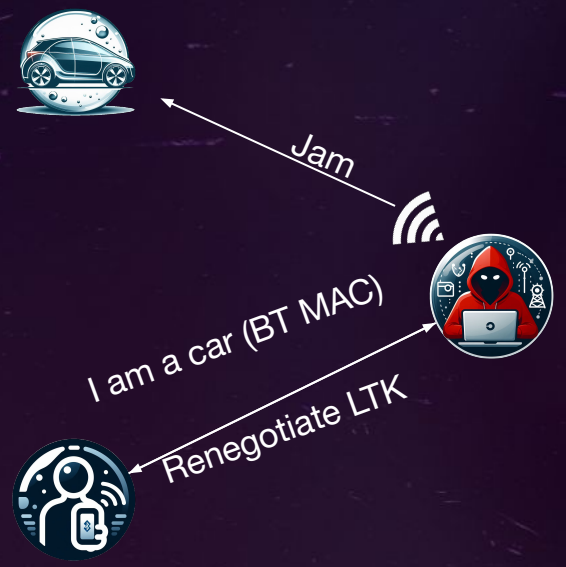
- Implementation-specific
  - IoT hacking
    - https://www.youtube.com/watch?app=desktop&v=YPcOwKtRuDQ
  - Reverse Engineering and Binary exploitation (ideally)
    - from hardcoded keys to buffer and heap overflows

# Demo

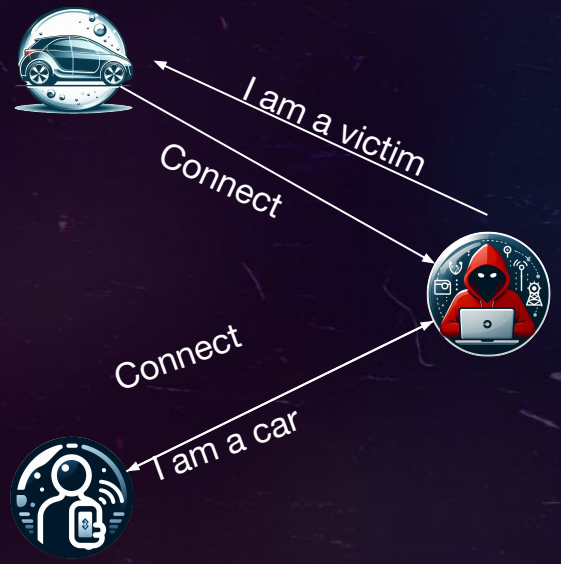

# Understanding an Impact

# DoS = MitM

1. Selective Jamming




2. Pre-commit with different MAC addr.
Or exploit other MitM vulnerabilities

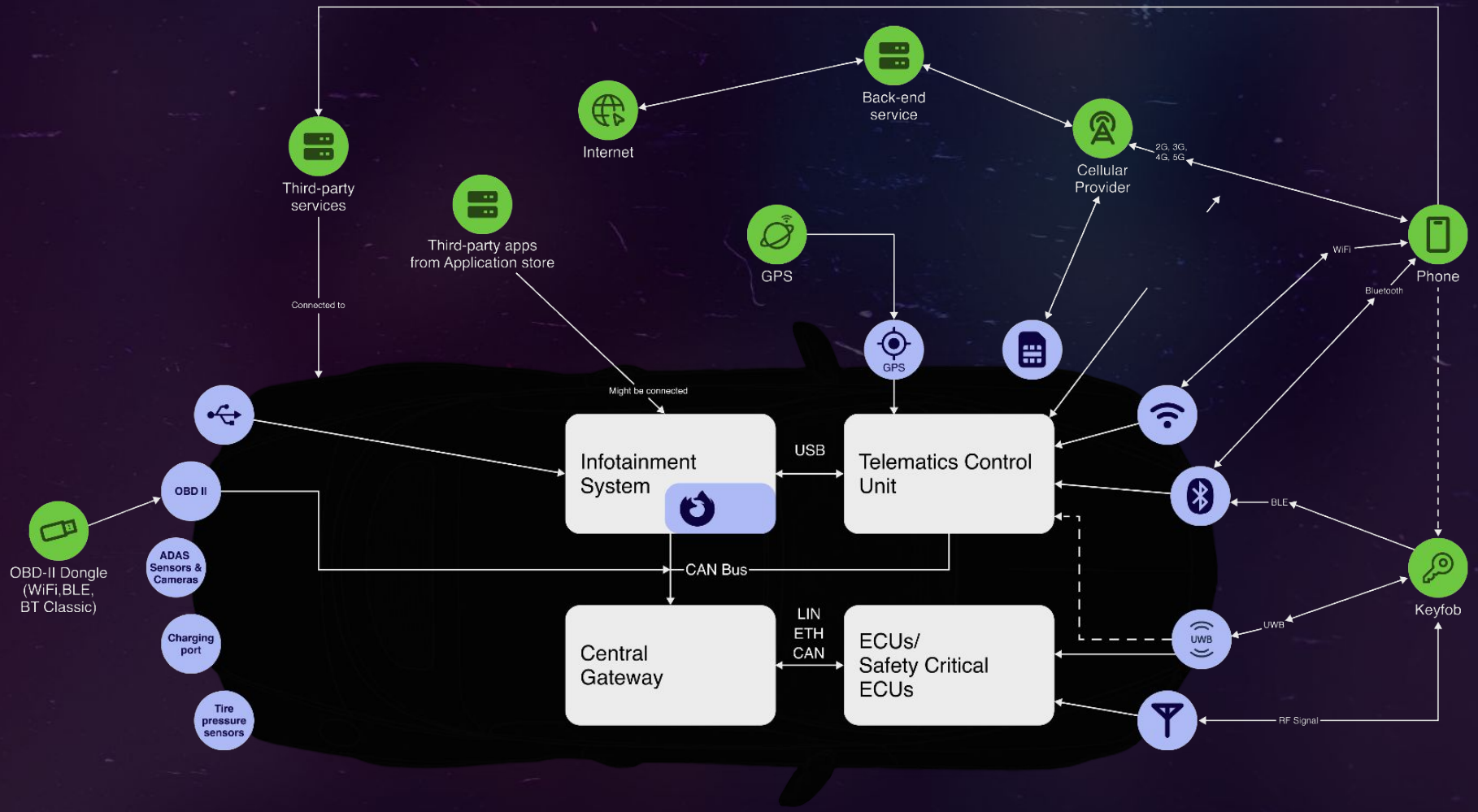



3. DoS and advertise as a car


Link key is renegotiated after 2-3 failed attempts.

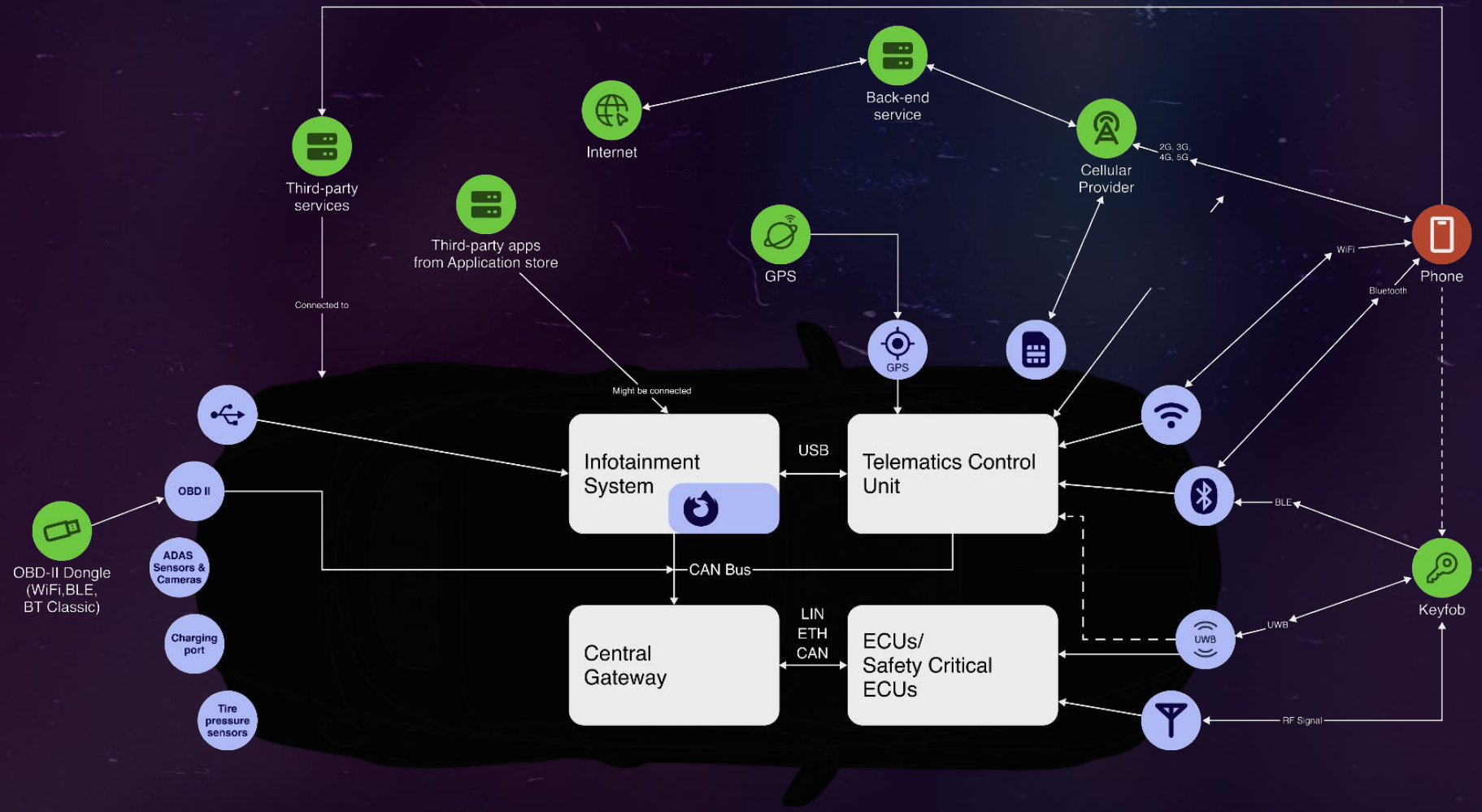
# MitM = High Severity attack

The Android security team has conducted an initial severity assessment on this report. Based on our published severity assessment matrix (1) it was rated as High severity and High quality.



# MitM = High Severity attack

The Android security team has conducted an initial severity assessment on this report. Based on our published severity assessment matrix (1) it was rated as High severity and High quality.



# MitM = High Severity Attack - Example

- Exploits typical to cars functionality
- Adversaries may spoof the MAC address of an already connected device or simply connect to a device and get access to SMS messages
  - Hijack account
  - Leak OTPs, etc.

- Might remain after it will be hardened in Android, Apple - ???* - not going to fix a 1-click attack
  - Why? - if an adversary controls IVI they may already have access to MAP
  - If they control Bluetooth link (MitM) they may try to get access to MAP

- For more information:
  - https://www.youtube.com/watch?v=JL7a_oLoXHY


COMMON VULNERABILITY SCORING SYSTEM

# Physical attacks matter in vehicle security

- Is hijacking a car a 6.4 or 6.8 medium vulnerability?
- Be careful with CVSS when evaluating vulns in car context.

Base Score **6.4 (Medium)**

**Attack Vector (AV)**

Network (N) | Adjacent (A) | Local (L) | **Physical (P)**

**Attack Complexity (AC)**

Low (L) | **High (H)**

**Privileges Required (PR)**

**None (N)** | Low (L) | High (H)

**User Interaction (UI)**

**None (N)** | Required (R)

**Scope (S)**

**Unchanged (U)** | Changed (C)

**Confidentiality (C)**

None (N) | Low (L) | **High (H)**

**Integrity (I)**

None (N) | Low (L) | **High (H)**

**Availability (A)**

None (N) | Low (L) | **High (H)**

# Going beyond

- BlueToolkit and techniques shown can of course be used for any Bluetooth (Classic) system
- We also tried it out on CYD avionics lab with real aircraft hardware


| Index | Exploit | Result | Data |
|---|---|---|---|
| 1 | au_rand_flooding | Not vulnerable | 0 |
| 2 | feature_response_flooding | Not vulnerable | 0 |
| 3 | braktooth_knob | Vulnerable | vulnerable |
| 4 | feature_req_ping_pong | Not vulnerable | 3 |
| 5 | reconnaissance_SC_supported | Vulnerable | No Secure Connections supported |
| 6 | reconnaissance_possible_BLUR | Not vulnerable | No LE supported, Cross transport attacks are not going to work |
| 7 | reconnaissance_SSP_supported | Vulnerable | SSP supported, weak cryptography is used |

# Conclusion

- Doing security research into cars is not that hard
  - Use your existing specialization or learn a new one
- Report to
  - Bug Bounty Programs, Pwn2Own, etc.
  - Directly to a manufacturer
  - National CERTs
- Why
  - Harvest CVEs
  - Get rewards

# Q&A


## Scan to discover more

https://github.com/sgxgsx

Contact: ysoschwytz@protonmail.com

Design: ux.uidesigns@protonmail.com