BlueSweeper

Bluetooth Frame Injection for Existing Peripheral Connections

Emilie Leavitt & Wesley Newsam
Advised by Dr. Dean Sullivan
Department of Electrical and Computer Engineering
University of New Hampshire

Motivation

This project was inspired by Samy Kamkar's KeySweeper, a functional USB wall charger containing a concealed keylogger running on an Arduino. KeySweeper targets only keyboards using a proprietary 2.4 GHz RF protocol, which has been largely replaced by Bluetooth in modern peripheral devices. This raised the question:

Could Bluetooth connections be exploited in the same way?

Preliminary Research

Mirage → Python-based modular framework designed to facilitate security analysis of wireless communications.
InjectaBLE → Attack that allows for the injection of malicious traffic into existing BLE connections using the custom "ButteRFly" firmware.
Zephyr → Real-Time Operating System that provides a BLE stack that can be used alongside Mirage to perform MitM attacks.

Acknowledgements

Special thanks to Romain Cayre for designing Mirage and InjectaBLE, without which BlueSweeper would not have been possible, and to Sarah Mader, for the custom Zephyr firmware and her crucial additions to Mirage.

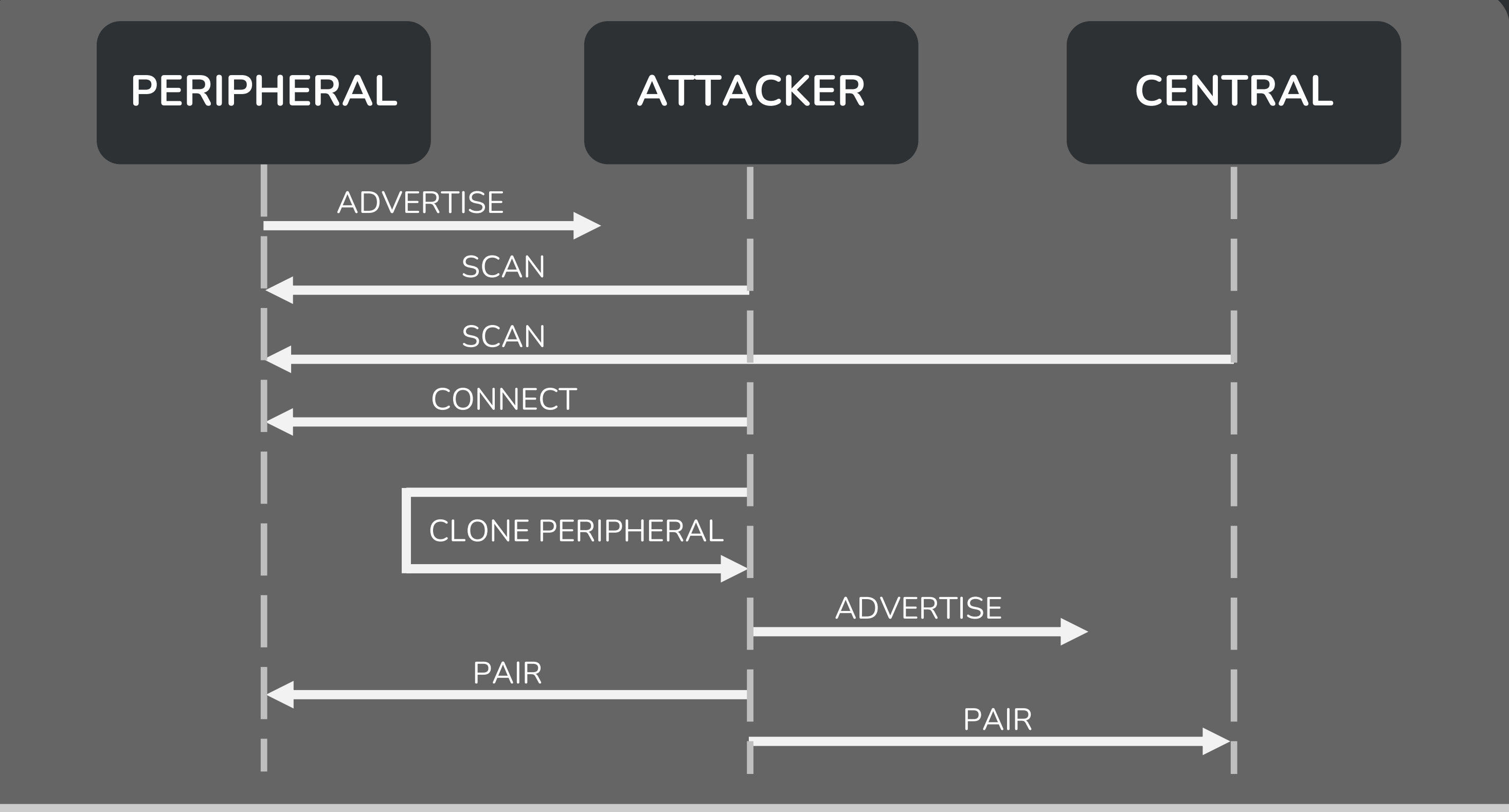
B luetooth **L** ow **E** nergy

BLE was introduced in Bluetooth 4.0 (2009) to reduce power consumption without sacrificing effective range.

BLE connections consist of two device roles:

Central → Scans for available peripherals and initiates the connection (typically a phone, laptop, or PC)

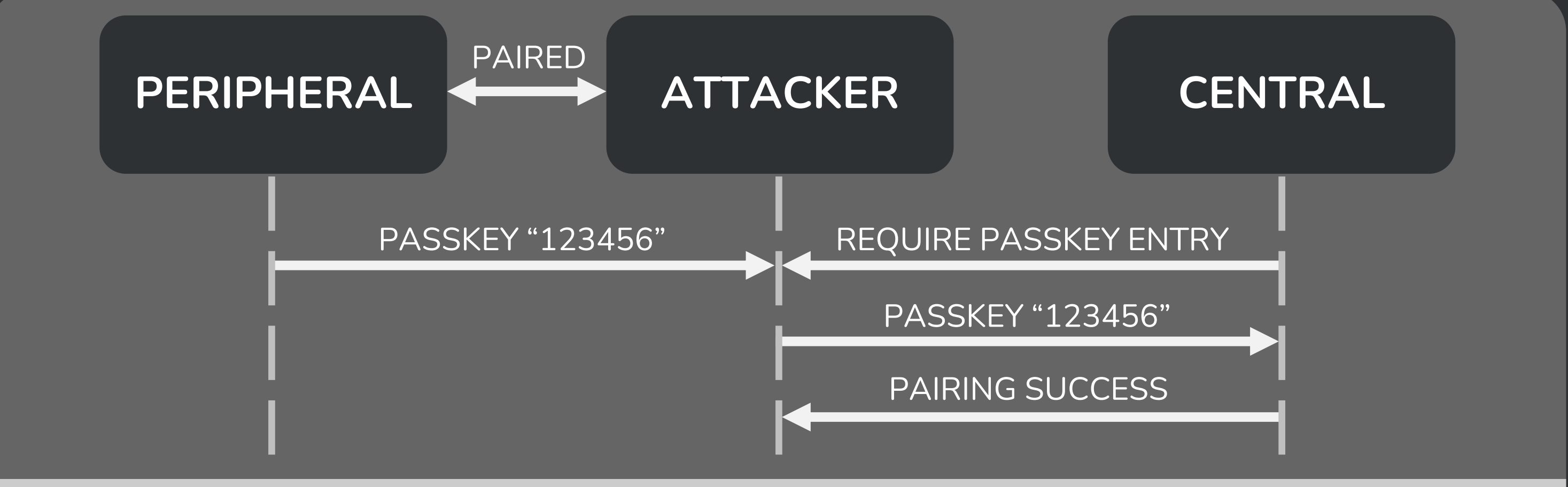
Peripheral → Broadcasts advertisement packets and waits for central to connect (typically a keyboard, mouse, or audio device)



Pairing Process

Pairing Methods

JustWorks	No user interaction required, though confirmation on one device may be requested.
Numeric Comparison	Each device displays a 6-digit number and requires the user to confirm that the value displayed on each device is the same.
Passkey Entry	The user either enters a 6-digit passkey on each device or one device displays a number, and the user enters it on the other.
OOB	One device exchanges a 128-bit value to the other using a non-BLE method. Out-of-band methods are vendor-specific.



Passkey Entry Bypass

You're working with a Bluetooth keyboard, maybe alone in your office or at a cafe. Suddenly, your keystrokes stop going through; your keyboard has disconnected. You check your Bluetooth settings and try to reconnect, but it fails. You press a button on the keyboard to re-pair it. You see the device listed, so you connect. Finally, your keyboard is working again.

Nothing more than a quick interruption, right?

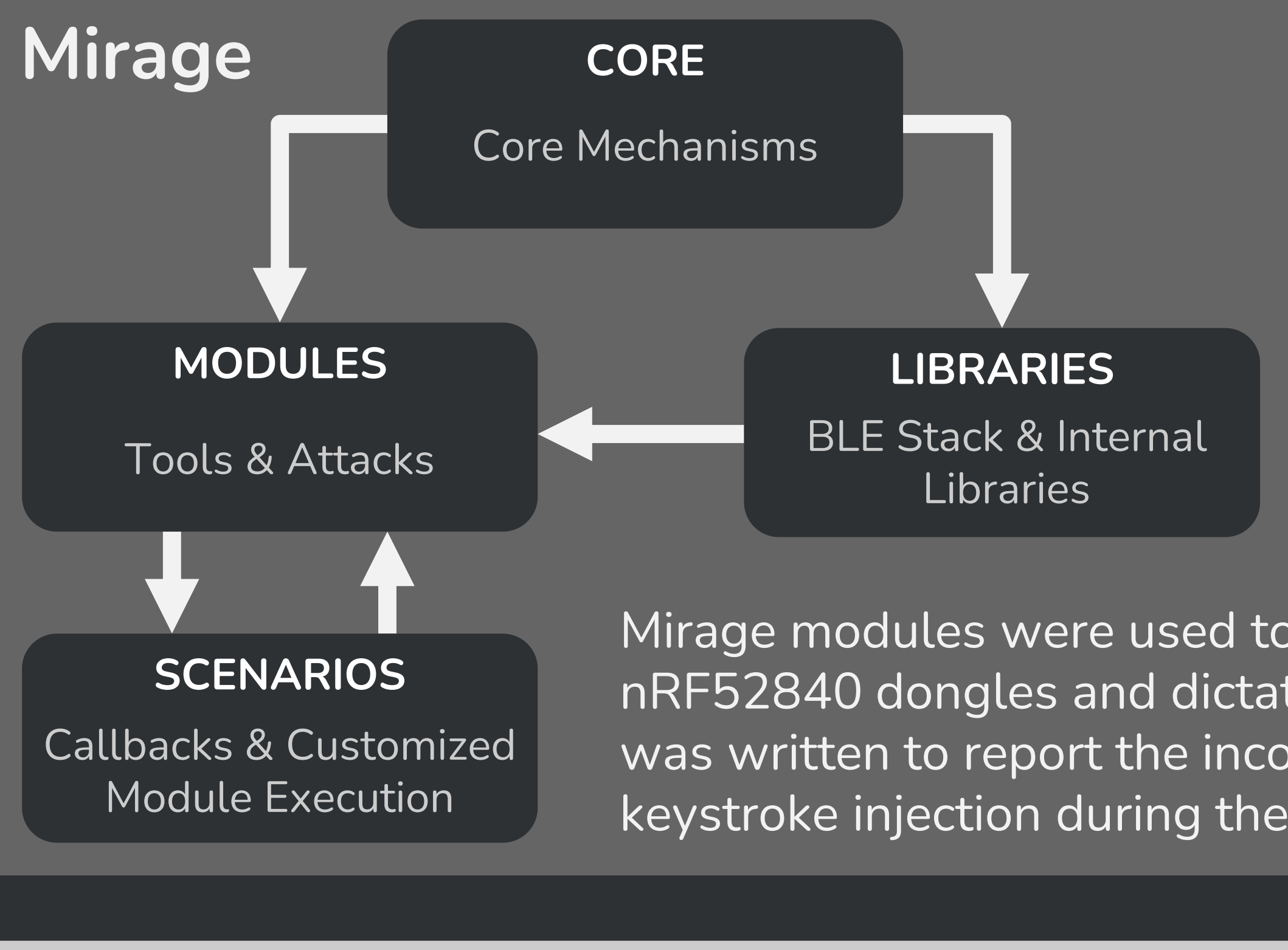
In reality, someone is now connected between your keyboard and your laptop. They can see and record everything that you type.

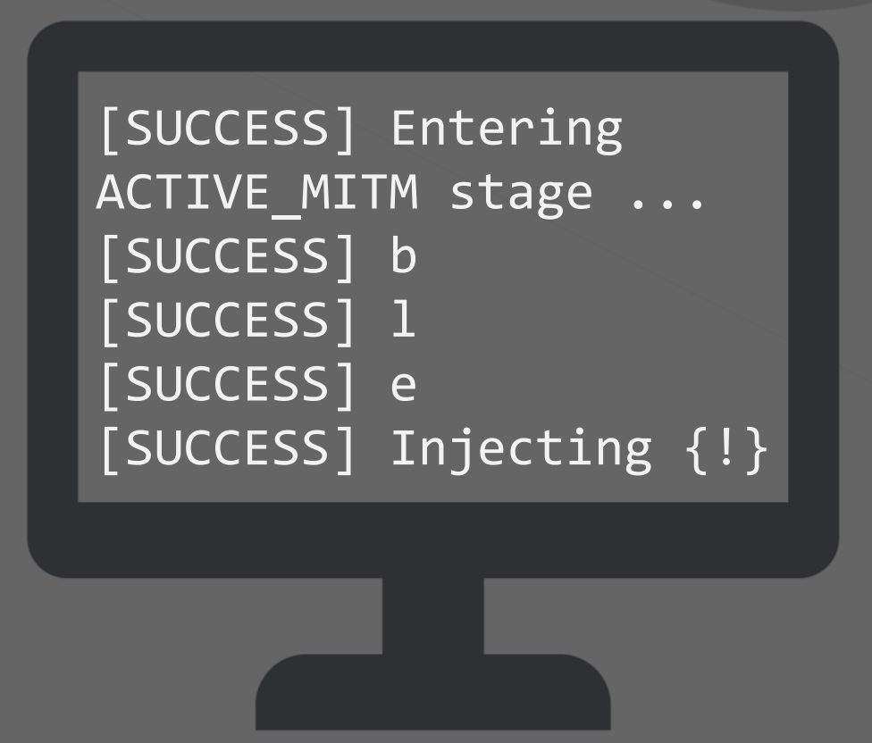
Did you do anything wrong? No. What is the alternative? To stop using the keyboard? This would be a very strange reaction to your keyboard disconnecting. You may think that you'd be smart enough to notice this attack, but you might not notice anything out of the ordinary.

Attacks were also performed on a BLE lightbulb, which had exposed **GATT characteristics,** allowing full control and monitoring of color, brightness, and power.

```
[SUCCESS] Master connected : 70:BC:10:62:28:AC
[INFO] << BLE - Security Request Packet | authentication=0x8 >>
[INFO] Exchange MTU Request (from master) : mtu = 527
[INFO] Redirecting to slave ...
[INFO] Exchange MTU Response (from slave) : mtu = 23
[INFO] Redirecting to master ...
[INFO] << BLE - Pairing Request Packet | outOfBand=no |
inputOutputCapability=0x4 | authentication=0x2d | maxKeySize=16 |
initiatorKeyDistribution=0xe | resp[illegible]istribution=0xf >>
[INFO] << BLE - Pairing Respon[illegible]=no |
inputOutputCapability=0x[illegible]
authentication=0x8 | m[illegible]ion=0x5
| responderK[illegible]
[INFO] [illegible] 0x1
/endHa[illegible]
[INFO[illegible]
[IN[illegible]
key_x=[illegible]d0\xe5k\xac\xd7\xad\x18
\x1[illegible]y=b'\xfc\xf0\x0b\xcc,\xde
\xa[illegible]\x7f~Ww\x89\xa0\xef\xc5\xb6\x9a
\xc[illegible] >>
[INFO] << BLE - Public Key Packet | key_x=b"\xf7[\x1e\x9d\x06\x9c
\xbc\x9fp\xae\x06\xa3\x13*ZG\x03g\xbb\xa7\x13I\x00\xb1\x9b\xfa\x17
\xbe\xa62n'" | key_y=b'\xd4\x95\x1c\x11\x88\x13J\x91\x1e\tVJ\xf0N
\x83Br\xf0\x81\xd9\xb5EB$\xf4i\x1ao\x01\xc0[\xd2' >>
[INFO] DH Key generated: d23e0b48f9a03a41812732e74cbf81ec11d84f8389
ac5cad24ee4cc02554503c
[INFO] << BLE - Pairing Confirm Packet | confirm=63f6bb9095ab6af15
73fccfa3dc947c1 >>
```

Mirage

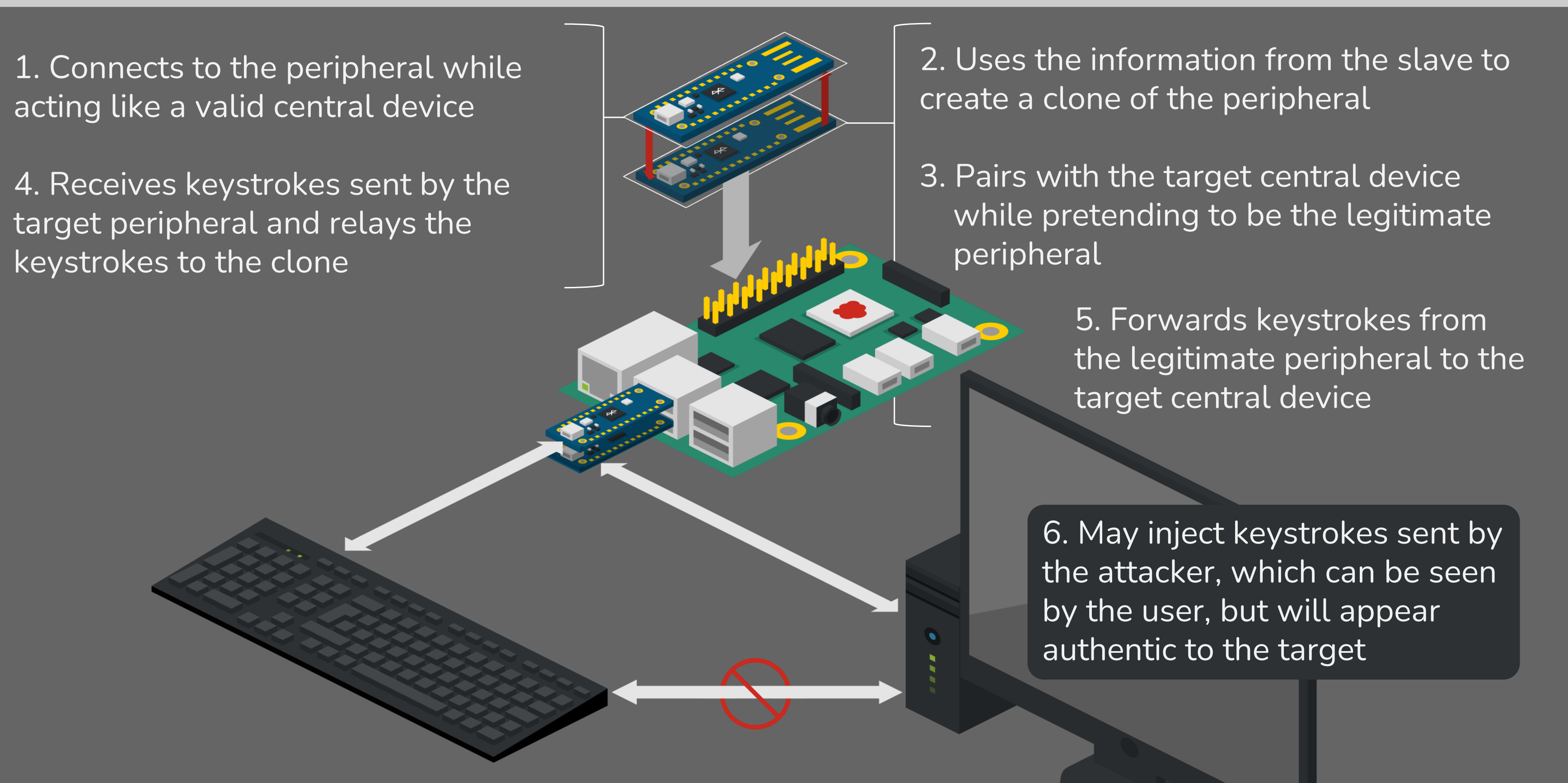

CORE
Core Mechanisms
MODULES
Tools & Attacks
LIBRARIES
BLE Stack & Internal Libraries
SCENARIOS
Callbacks & Customized Module Execution



```
[SUCCESS] Entering
ACTIVE_MITM stage ...
[SUCCESS] b
[SUCCESS] l
[SUCCESS] e
[SUCCESS] Injecting {!}
```

Mirage modules were used to configure the behavior of the nRF52840 dongles and dictate each step of the attack. A scenario was written to report the incoming keystrokes and enable active keystroke injection during the attack.

Man-in-the-Middle Attack



1. Connects to the peripheral while acting like a valid central device

2. Uses the information from the slave to create a clone of the peripheral

3. Pairs with the target central device while pretending to be the legitimate peripheral

4. Receives keystrokes sent by the target peripheral and relays them to the clone

5. Forwards keystrokes from the legitimate peripheral to the target central device

6. May inject keystrokes sent by the attacker, which can be seen by the user, but will appear authentic to the target

At any point, keystrokes can easily be injected by the attacker, enabling a plethora of other attacks via a DuckyScript.

Conclusion

- The attacks shown can pose a legitimate threat to the consumer and should be addressed by the Bluetooth SIG.
- Features with names like "Secure Connections" and "MitM protection flag" incorrectly suggest that the Bluetooth Low Energy protocol is protected against MitM attacks.
- BLE connections are targeted here, but legacy Bluetooth connections are equally vulnerable if not more so.
- When security features are made optional, many vendors will opt out, especially if the feature impedes user experience.
- Performing this attack is remarkably inexpensive, as only the nRF52840 dongles are required ($10 each).

Future Work

- Connection jamming has not been implemented, but there are many methods of achieving this that require minimal effort.
- Creating a housing similar to that of KeySweeper would enable long-term monitoring if the targets remain within range.
- The Bluetooth 5.4 Core Specification was adopted in February 2023, including new security features like the LE GATT Security Levels Characteristic and Encrypted Advertising Data. Unfortunately, these additions have little impact on the demonstrated vulnerabilities.