Bluetooth Mouse Data Sniffing and Attacking

IoT Lab, SEMO

Project Supervisor: Dr. George Li

Team:

Yash Harikrishna Barot

Sartaj Jamal Chowdhury

Date: 2-21-2024

2-25-2024

Equipment and Software Used:

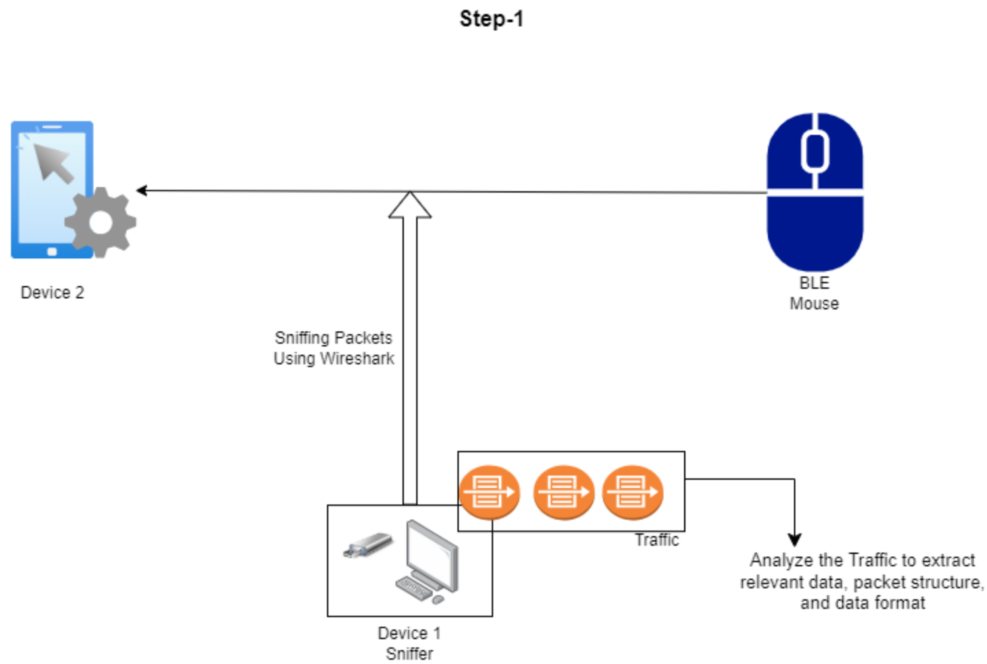
1. nRF52840 Dongles (1 for Sniffing Packets; 1 for trying to Attack)
2. Wireless Bluetooth (BLE) + 2.4G Mouse
3. Wireshark
4. 1 device to connect with the mouse and create scenarios.
5. 1 device to capture and analyze Bluetooth packets. Later, try to attack the Mouse.
6. nRF Connect for Desktop
7. ESP32

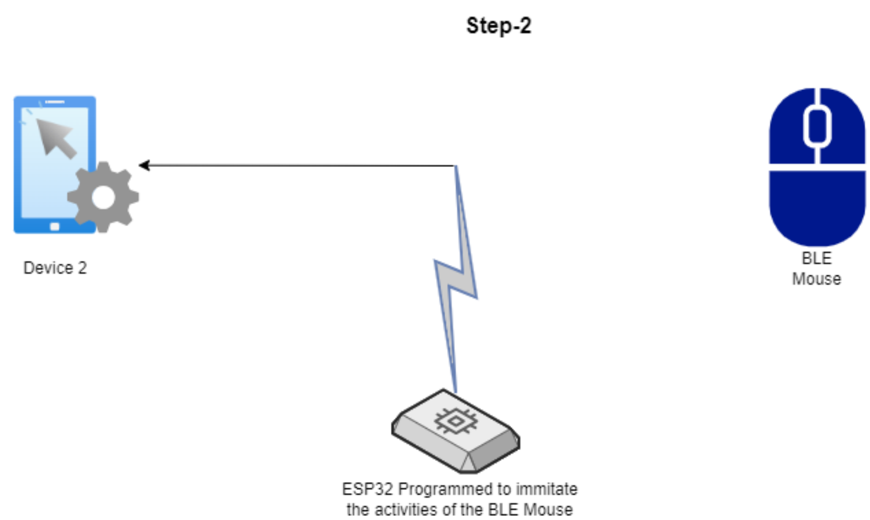
Summary:

1. Setting up an nRF52840 Dongle as a sniffer to help capture Bluetooth traffic using Wireshark.
2. - Creating certain scenarios, such as – only Left-Click, only Right-Click, only moving the mouse upwards, only moving mouse downwards, only scrolling upwards, only Scrolling downwards, mixtures of two or more of the above scenarios.
   - Also, capturing the Bluetooth packets in Wireshark after doing so, and stopping the capture right after each action was completed. This helped understand the values correlated with each of these actions.
   - After repeating each granular steps several times, we were certain about the Value that was generated for each specific actions (such as Right-Click is always represented by the value: 020000000000).
3. - Next, we setup another nRF52840 Dongle as nRF Connect for Desktop BLE Standalone, just to see what happens when we perform the Mouse Actions (Left-Click, Right-Click, etc.).
   - And we also find out on which portion of the Human Interface Device (HID) stack changes are made when we perform those actions. This actually also helped us further verify that the values for each of these mouse actions are fixed (such as Right-Click is always represented by the value: 020000000000).
   - Then we tried to use the different action values on that portion of the stack to see what happens. This was an effort to try to imitate the Mouse, for those specific actions.
   - We were able to change the attribute value of the mouse. But in doing so, the output we found was not fully similar to the output we got after performing the actions from the mouse itself.
4. We try to understand the sequence of Bluetooth packets that we captured for the Bluetooth Mouse.
   We try to compare it with the sequence of Bluetooth packets we sniffed from the Bluetooth LED Bulb.
5. Explore the concept of replicating the Bluetooth mouse using an ESP32 and perform a “mouse emulation attack” or “wireless mouse spoofing”.

In next section we provide further details on the above findings with screenshots.

0. Simplified Architecture





We sniff and capture the wireless communication packets between the Bluetooth mouse, and the device that it is connected to (Device-2). We use our sniffer setup – Device-1, nRF52840 Dongle, and Wireshark to capture and analyze the packets.

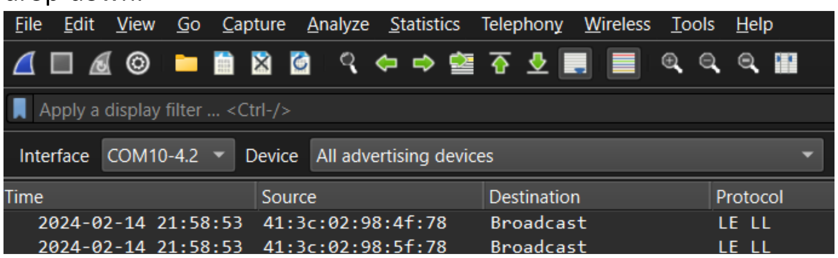
We are working on our Attacker setup. We are exploring the possibility of reverse-engineering the BLE mouse and leveraging an ESP32 to create a copy of it. This way we may try to perform a “mouse emulation attack” or “wireless mouse spoofing”.

1. Setting Up the Sniffer

The steps for setting up the nRF52840 Dongle as a Bluetooth packet sniffer is documented in the following Word file named: "2. *Setting Up nRF52840 Dongle as a Bluetooth Packet Sniffer.docx*"

2. Creating the Scenarios with the Mouse

For each of the different scenarios, we maintained a constant procedure, which is mentioned below:

1. First, we make sure Bluetooth in all our devices are off. Even for Device-1.
2. Start capturing Bluetooth traffic in Wireshark from Device-1.
3. Turn on the Mouse in Bluetooth mode.
4. While sniffing select the Bluetooth Mouse named "BT5.1 Mouse" from "All advertising devices" drop-down.


5. Turn-on the Bluetooth of Device-2. Scan till we find the BT5.1 Mouse.
6. Pair with BT5.1 Mouse.
7. We make sure the mouse-bottom is not touching any surface, to avoid generating any unwanted cursor movement, unless we want that specific action. In that case, we just use a finger to drag over the sensor to move the cursor a little.
8. Perform the specific action or the set of actions with the Mouse. (e.g. Right-Click, or cursor moving).
9. Stop capturing packets in Wireshark.
10. Turn-off mouse.
11. Unpair mouse from Device-2.
12. Unplug and again plug in the sniffer dongle for upcoming captures.
13. Save and analyze the captured packets in Wireshark.

At first, we captured packets for some Random mouse activities. It was the first time we were able to capture some ATT (attributes) packets. However, segregating the packets to understand which packet resembles what Mouse-Actions was not easy to understand from this big jungle of ATT packets. Hence, decided to granularly capture packets for only one action each attempt. Later, we also tried combinations to verify the values are not changing. For each action, we repeated the whole process a number of times too, for strong verification. We created the following scenarios, and captured the packets for each of these instances separately:

a. Only Right-Click twice.

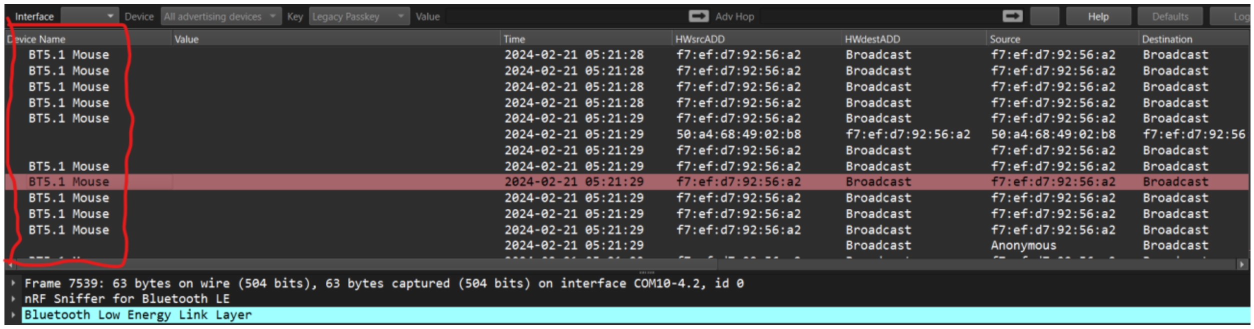
b. Only Left-Click twice.
c. Left-Right-Left-Right-Right click.
d. Right-Left-Right-Left-Left click.
e. Cursor Movement: Moving mouse up.
f. Cursor Movement: Moving mouse right.
g. Cursor Movement combined with One Left-Click.
h. Scrolling Up.
i. Scrolling Down.
j. Scrolling Up and then Down.
k. Scrolling Up-Up and Down-Down.
l. Only pressing the Scroller Button three times.
m. Pressing the "M" Button 3 times.

We saved all the packets and moved on to packet-analysis:

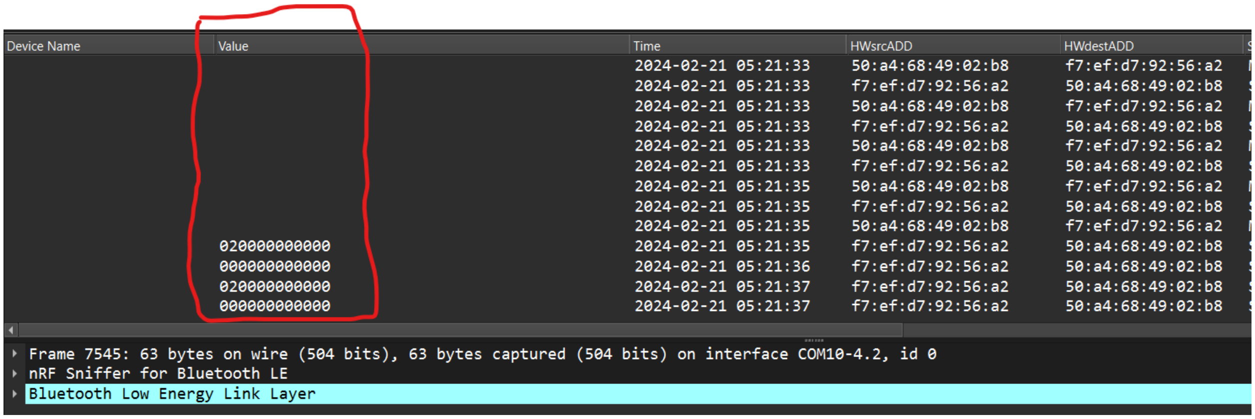
Name	Date modified	Type	Size
_Random_First_Capture_2_15_24.pcapng	2/15/2024 12:00 PM	Wireshark capture ...	807 KB
1_OnlyRightClickTwice_pt1.pcapng	2/20/2024 11:25 PM	Wireshark capture ...	708 KB
2_OnlyRightClickTwice_pt2.pcapng	2/20/2024 11:37 PM	Wireshark capture ...	407 KB
3_OnlyLeftClickTwice_pt1.pcapng	2/20/2024 11:42 PM	Wireshark capture ...	342 KB
4_OnlyLeftClickTwice_pt2.pcapng	2/20/2024 11:48 PM	Wireshark capture ...	645 KB
5_LRLRRclicks_pt1.pcapng	2/20/2024 11:54 PM	Wireshark capture ...	437 KB
6_RLRLLclicks_pt2.pcapng	2/20/2024 11:57 PM	Wireshark capture ...	513 KB
7_1_MoveMouseRight.pcapng	2/21/2024 12:27 PM	Wireshark capture ...	466 KB
7_MoveMouseUp.pcapng	2/21/2024 12:04 AM	Wireshark capture ...	556 KB
8_RandomMouseMoving_OneLeftClick.pc...	2/21/2024 12:11 AM	Wireshark capture ...	596 KB
9_Scrolling_UP.pcapng	2/21/2024 12:16 AM	Wireshark capture ...	605 KB
10_Scroll_DOWN.pcapng	2/21/2024 12:19 AM	Wireshark capture ...	675 KB
11_ScrollingUP_DOWN.pcapng	2/21/2024 12:14 AM	Wireshark capture ...	556 KB
12_Scroll_UPUP_DOWNDOWN.pcapng	2/21/2024 12:23 AM	Wireshark capture ...	577 KB
13_ScrollButton_Press_ThreeTimes.pcapng	2/21/2024 12:25 AM	Wireshark capture ...	728 KB
14_M_Button_3Times.pcapng	2/21/2024 12:40 AM	Wireshark capture ...	544 KB

2.1 Wireshark Customized Columns

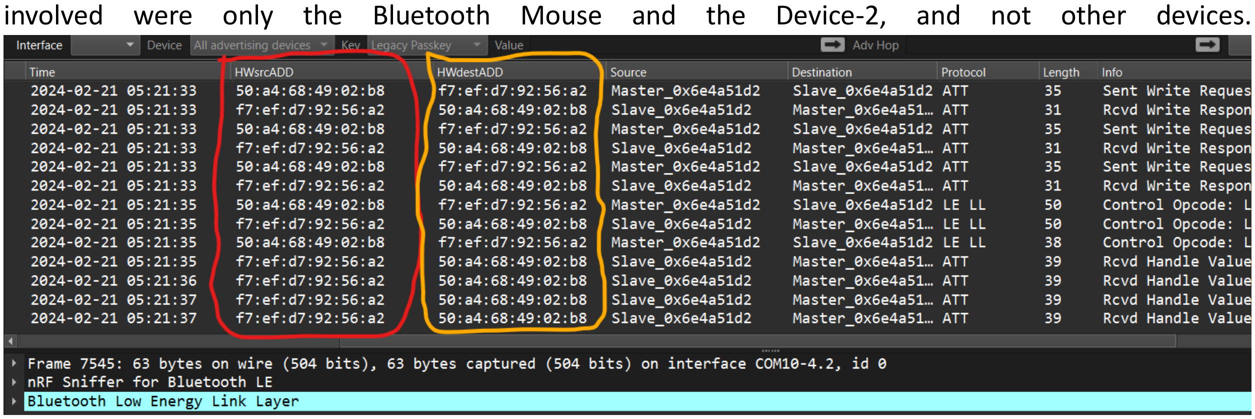
We added the device name column to quickly find where the Bluetooth Mouse packets start from:



We added the Value column to see the values generated for specific Mouse-Actions:



Built-in Source and Destination columns only show if the packet source or destination was the “Slave…” or the “Master…”. But to see what are the Bluetooth MAC Addresses of these slave and master devices in every step, we added the “HWsrcADD” and “HWdestADD” columns. This helped us verify the devices involved were only the Bluetooth Mouse and the Device-2, and not other devices.



Note: For each demonstration, we also filtered-out the packets with “Empty PDU” under the Info column.

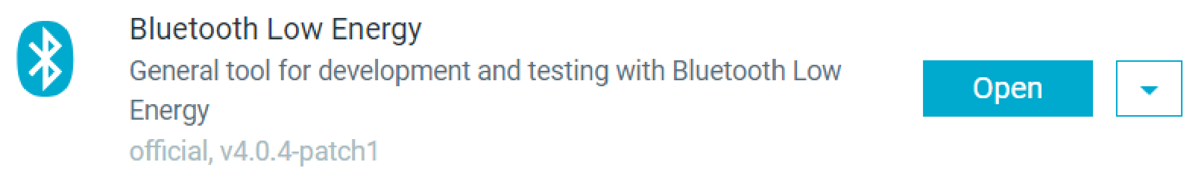
2.2 Findings from the different scenarios

We have analyzed the sequence of packets triggered during the steps of Advertising, Scanning, Pairing, and Information Exchange. We delve deep into that in the later sections of this report. In this segment we only compare and contrast the piece of information that we can verify to be the data or value that is unique for each Mouse-Actions. From our findings below are the Attribute data of the mouse which map to the following actions:

Mouse Action	Value	Direction
Right Click	020000000000	Mouse -> Device-2
	000000000000	Mouse -> Device-2
Left Click	010000000000	Mouse -> Device-2
	000000000000	Mouse -> Device-2
Scroll-Button Press	040000000000	Mouse -> Device-2
	000000000000	Mouse -> Device-2
"M" Button Press	0800070000000000	Mouse -> Device-2
	0000000000000000	Mouse -> Device-2
Scrolling Up	000000000100 000000000200 000000000300	Mouse -> Device-2
Scrolling Down	00000000ff00 00000000fe00	Mouse -> Device-2
Cursor Movements (up)	00fedfff0000 00fa3fff0000 00fb1fff0000 00ff9fff0000 00ff7fff0000 000090fe0000 000010ff0000 0000f0ff0000	Mouse -> Device-2
Cursor Movements (right)	00fe1f000000 00f7ffff0000 00f50f000000 00f3ffff0000 00fd1f000000 00f62f000000 0002f0ff0000 00fd1f000000	Mouse -> Device-2

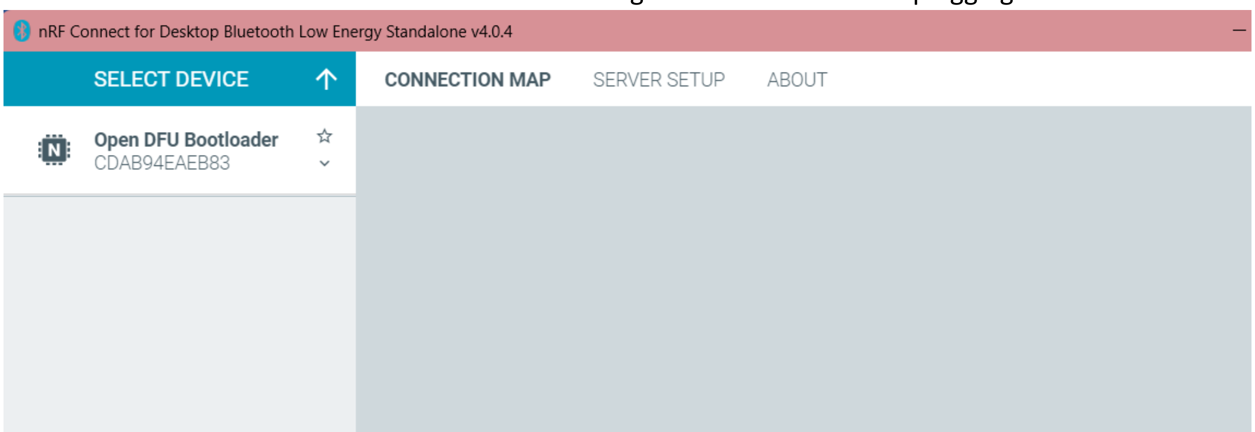
3. Verifying the attribute values using the nRF Connect for Desktop BLE Standalone

Either we use a different Dongle or reset the one we used as sniffer and work with it.

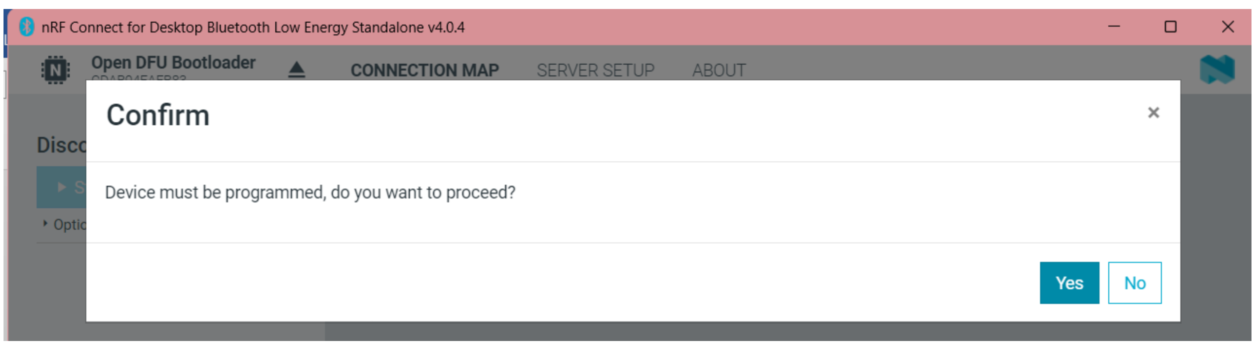


We open the above tool from nRF Connect for Desktop.

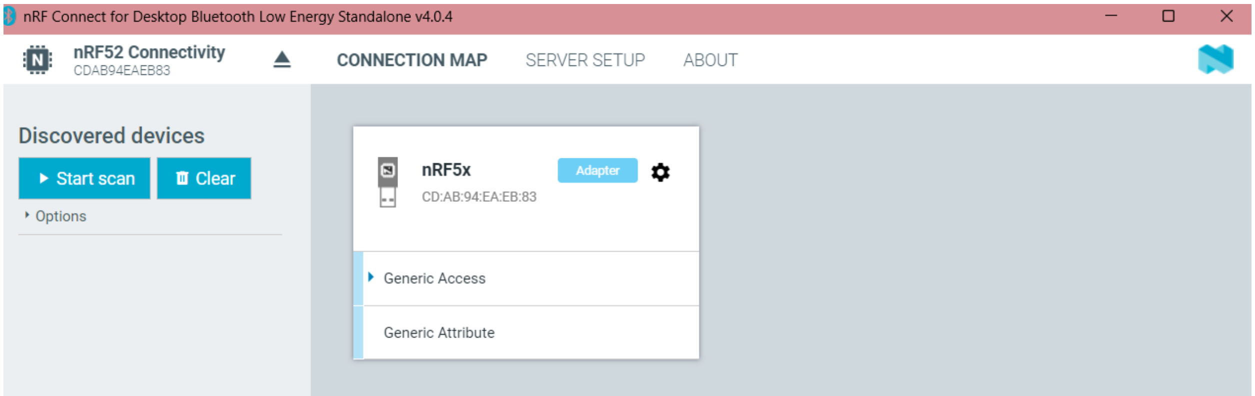
We select our nRF Dongle after plugging it in:



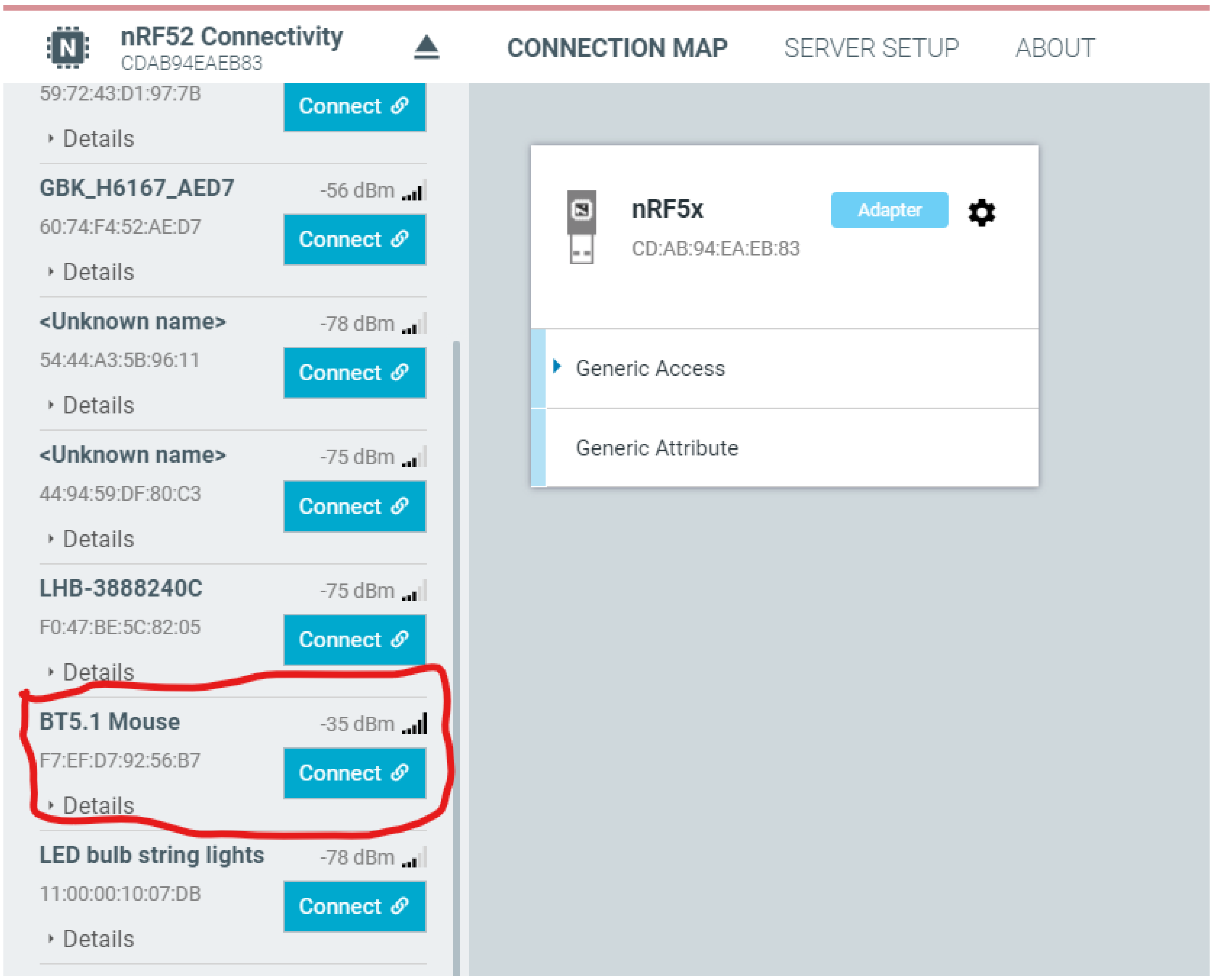
We select Yes to confirm.



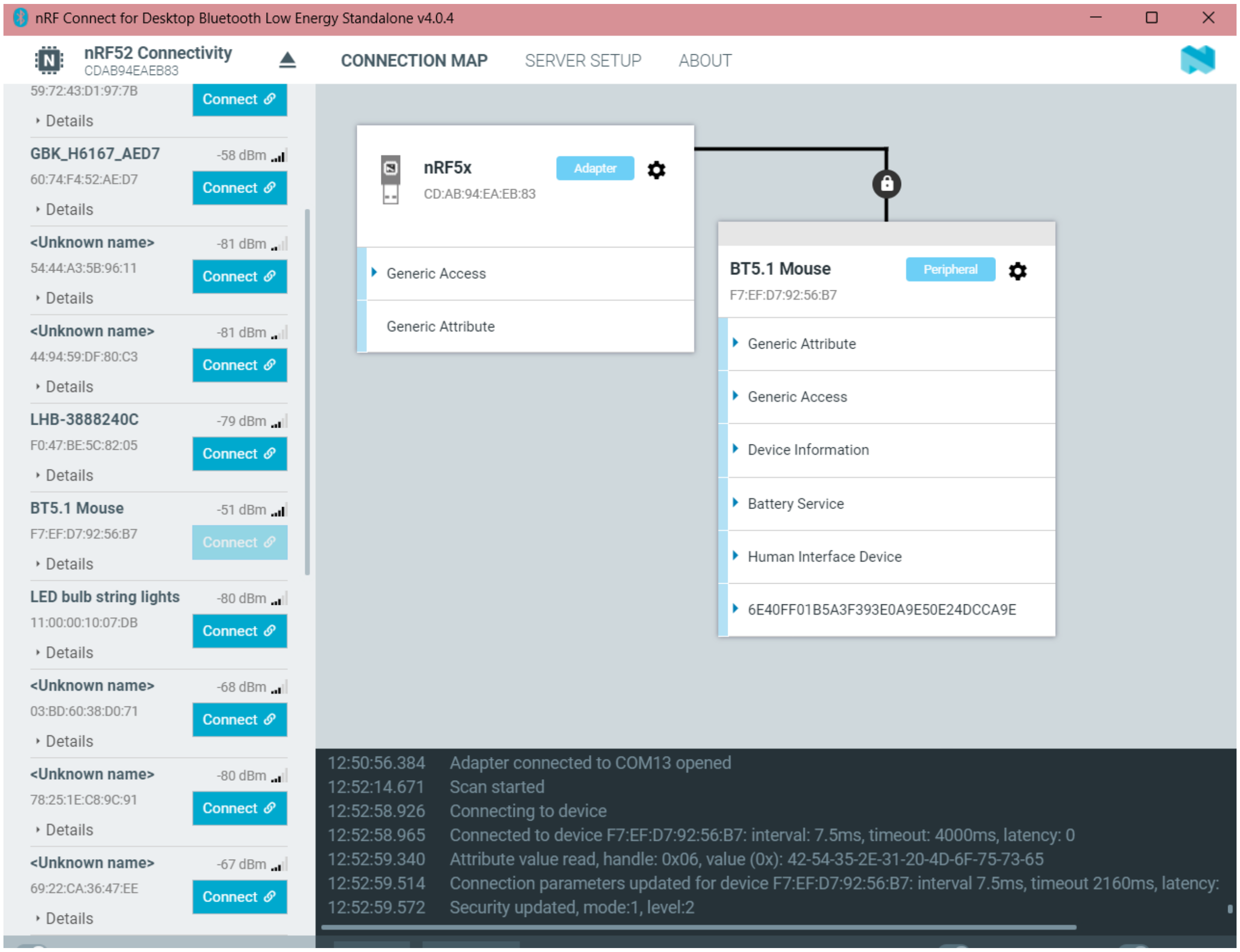
When the image is completed uploading-



We turn on our Bluetooth Mouse > Start Scan from the above.

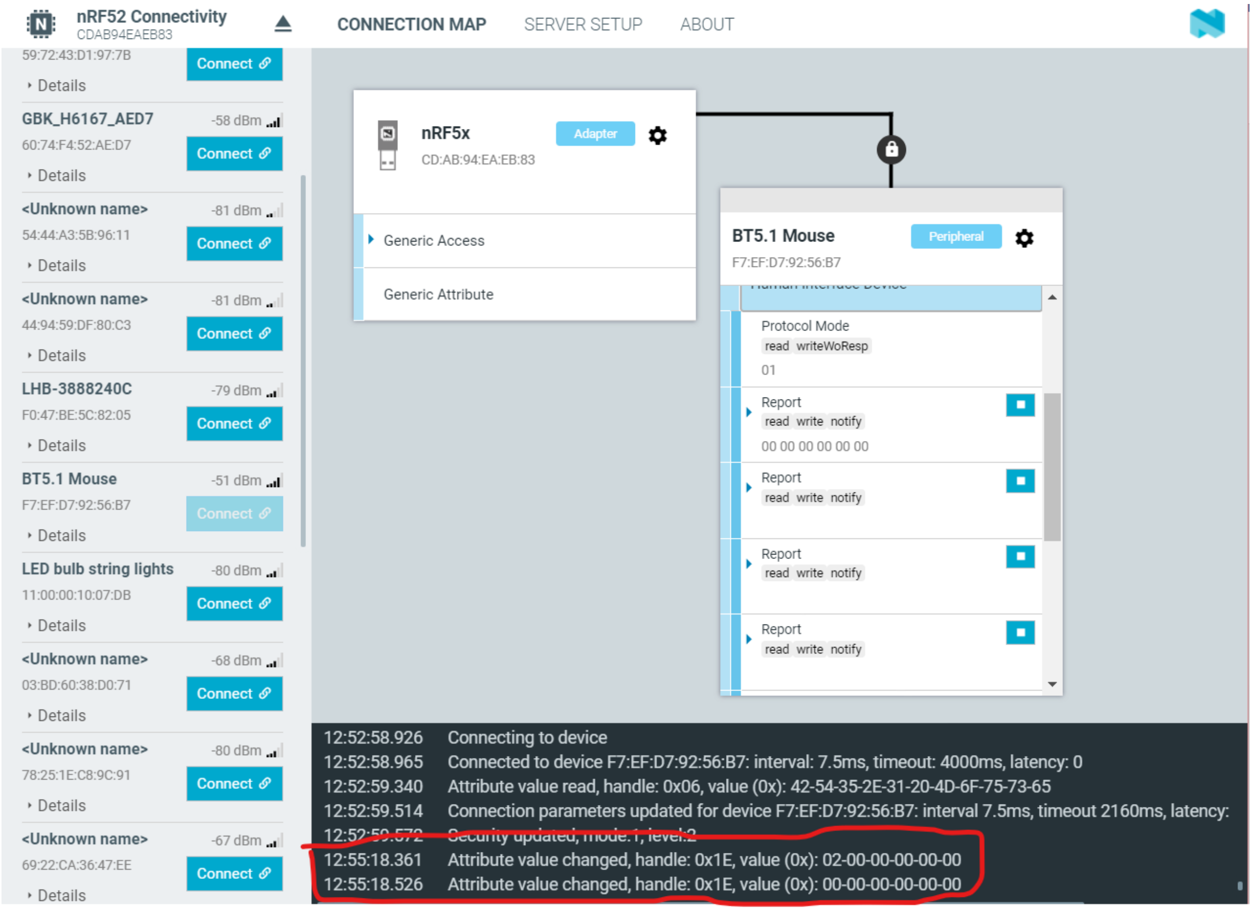


We connect with it.



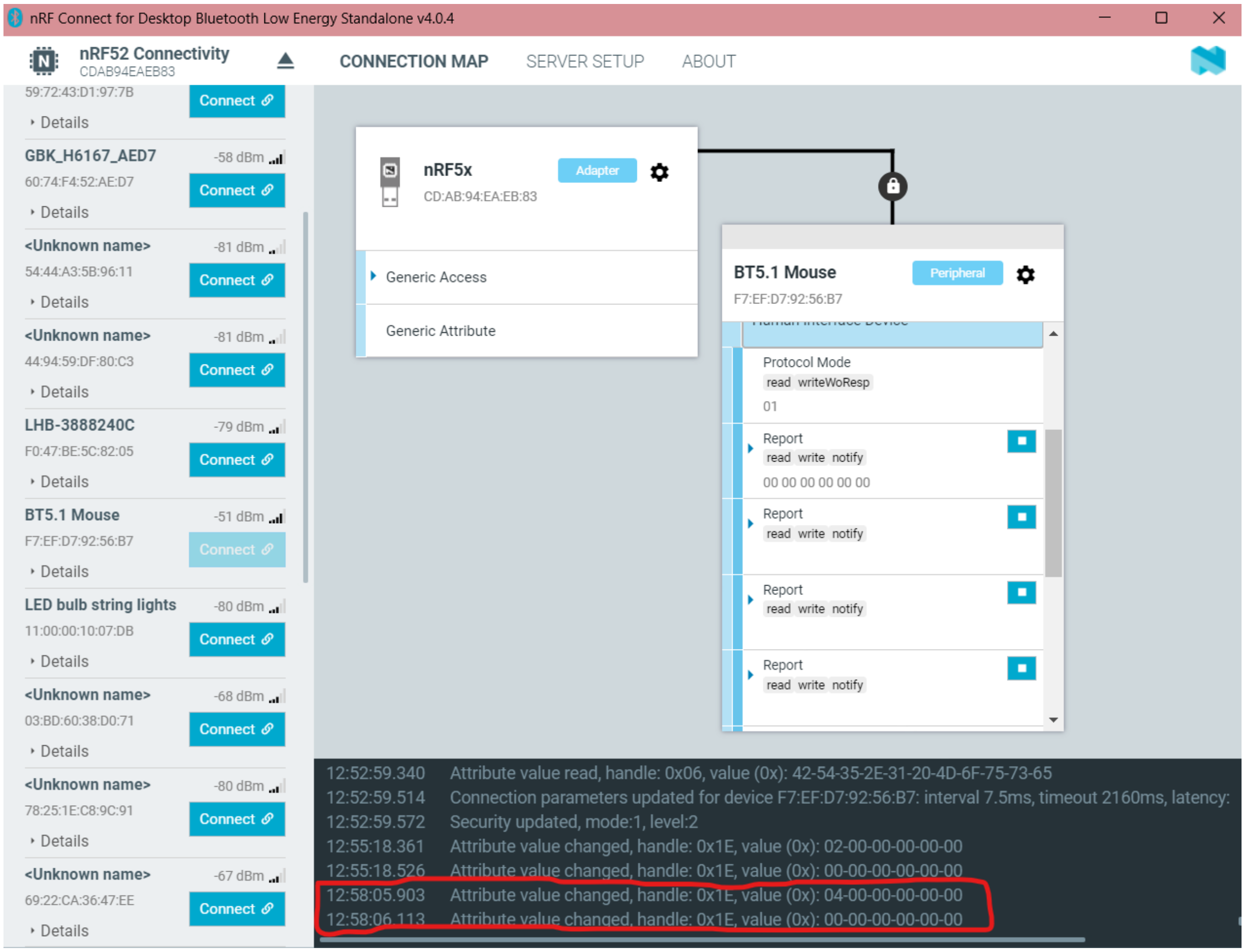
Now, we can verify our attribute values. We can, for example, press the “Right-Click” button on our mouse. And see what values are triggered.

For example, after pressing right-click we get the following two attribute value changes:



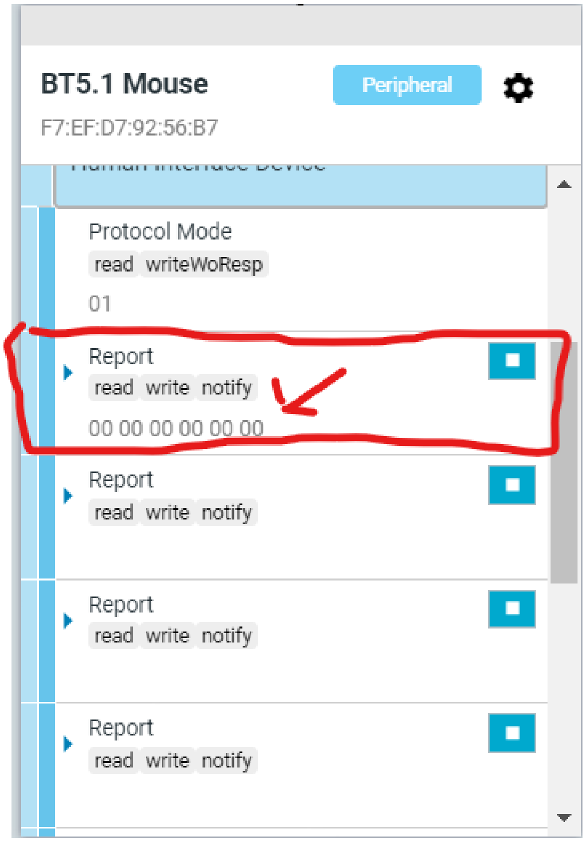
It does match with the values we found after capturing the data for right-click action.

Next, we also try clicking the scroll button:

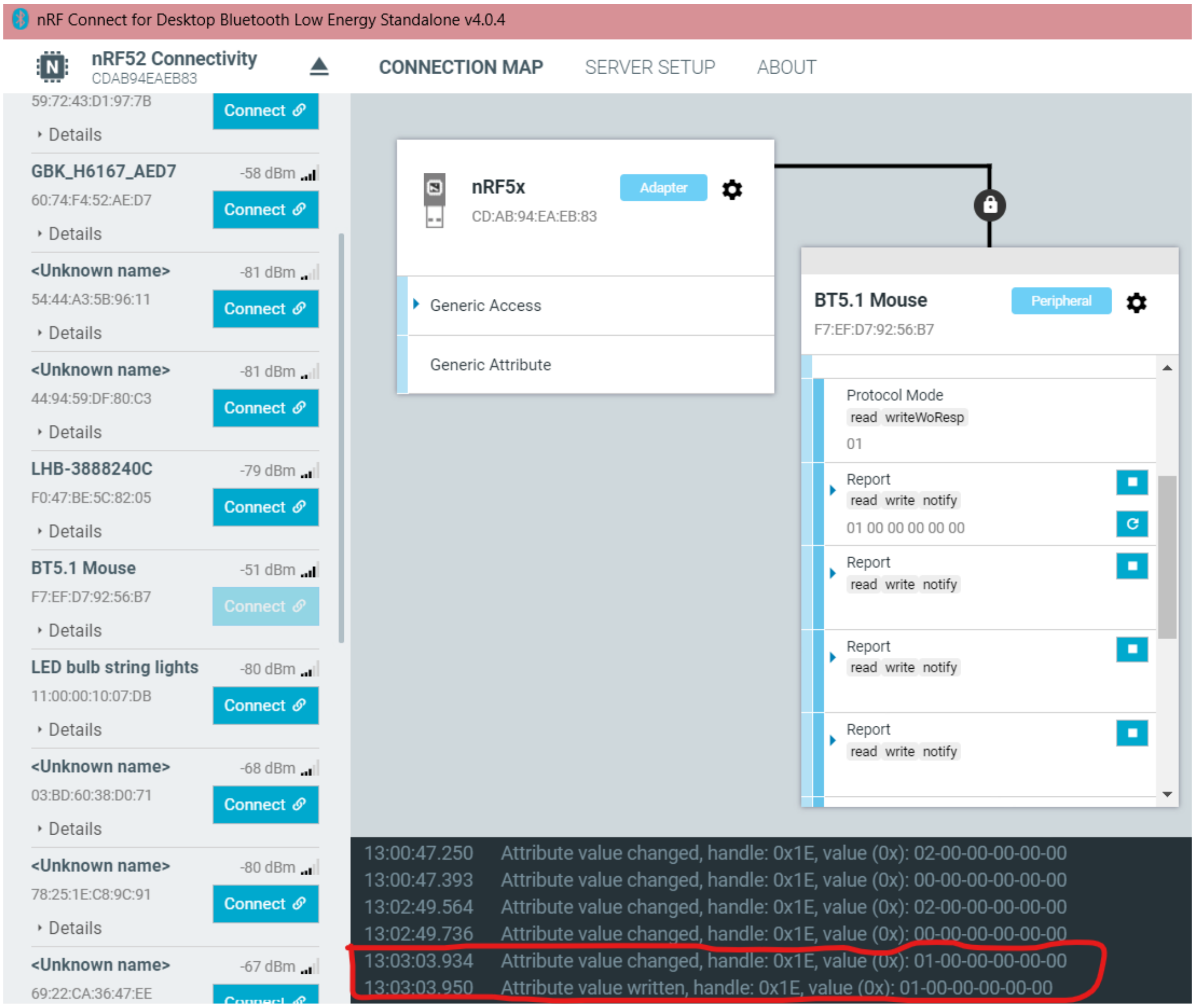


As expected, we get the same attribute value changes we found from Wireshark for pressing the Scroll-Button.

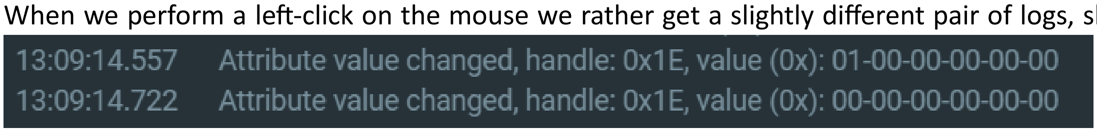
Next, we try to enter the attribute value of Left-Click (i.e. 010000000000) into the following field to see what happens. We chose this particular field because when we were clicking the mouse buttons, the following field was being updated.



When we input the attribute value of Left-Click (010000000000) we get the following to logs. First one says the “Attribute value changed…010000000000”, followed by: “Attribute value written…010000000000”.



When we perform a left-click on the mouse we rather get a slightly different pair of logs, shown below-



“Attribute value changed…010000000000”, followed by: “Attribute value changed…000000000000”.

4. Packet Sequence for the Bluetooth Mouse traffic

1. Firstly, the Bluetooth Mouse (named: BT5.1 Mouse) generates a bunch of Advertisement Packet with Inquiry Scan:

Device Name	Value	Time	HWsrcADD	HWdestADD	Source	Destination	Protocol	Length	Info
BT5.1 Mouse		2024-02-21 05:38:22	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:22	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:22	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:22	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:22	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
		2024-02-21 05:38:22	5e:83:34:3c:1b:e8	f7:ef:d7:92:56:a4	5e:83:34:3c:1b:e8	f7:ef:d7:92:56:…	LE LL	38	SCAN_REQ
BT5.1 Mouse		2024-02-21 05:38:22	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:22	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
		2024-02-21 05:38:22	40:d4:58:99:f9:43	f7:ef:d7:92:56:a4	40:d4:58:99:f9:43	f7:ef:d7:92:56:…	LE LL	38	SCAN_REQ
BT5.1 Mouse		2024-02-21 05:38:22	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
		2024-02-21 05:38:22					LE LL	38	Unknown
BT5.1 Mouse		2024-02-21 05:38:25	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:25	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:25	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:25	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:25	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:25	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:25	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse		2024-02-21 05:38:25	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND

ADV: This likely refers to an Advertisement packet, which is a type of packet used by Bluetooth LE devices to advertise their presence and capabilities to other devices.
INV: This might stand for Inquiry Scan, which is a type of scan performed by a device searching for nearby Bluetooth LE devices.

2. Next, we get CONNECT_IND events:

BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
	2024-02-21 05:38:31	5e:81:34:3c:1b:f8	f7:ef:d7:92:56:a4	5e:81:34:3c:1b:f8	f7:ef:d7:92:56:…	LE LL	38	SCAN_REQ
	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	32	SCAN_RSP
BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
	2024-02-21 05:38:31	7e:81:34:3c:1a:f2	f7:ef:d7:92:56:a4	7e:81:34:3c:1a:f2	f7:ef:d7:92:56:…	LE LL	38	SCAN_REQ
BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
BT5.1 Mouse	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
	2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:…	LE LL	60	CONNECT_IND
	2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	35	Control Opcode:
	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	35	Control Opcode:
	2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	32	Control Opcode:
	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	32	Control Opcode:
	2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	SMP	37	Sent Pairing Req
	2024-02-21 05:38:31	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	SMP	37	Rcvd Pairing Res

Connection Indication (CONNECT_IND): This is a standard Bluetooth LE event that indicates a device has received a connection request from another device and is accepting the connection. It's part of the LE Link Layer establishment process.

In the image we see that f7:….:a4 is the Mouse and it has received a connection request from 4f:…:f5 (which is likely the Device-2).

3. Then there is a sequence of Link Layer data transfers:

2024-02-21 05:38:31	f7:ef:d7:92:56:a4	Broadcast	f7:ef:d7:92:56:a4	Broadcast	LE LL	63	ADV_IND
2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:…	LE LL	60	CONNECT_IND
2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	35	Control Opcode: LL_FEATURE_REQ
2024-02-21 05:38:31	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	35	Control Opcode: LL_FEATURE_RSP
2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	32	Control Opcode: LL_VERSION_IND
2024-02-21 05:38:31	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	32	Control Opcode: LL_VERSION_IND
2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	SMP	37	Sent Pairing Request: AuthReq: Bonding, MIT
2024-02-21 05:38:31	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	SMP	37	Rcvd Pairing Response: AuthReq: Bonding | I

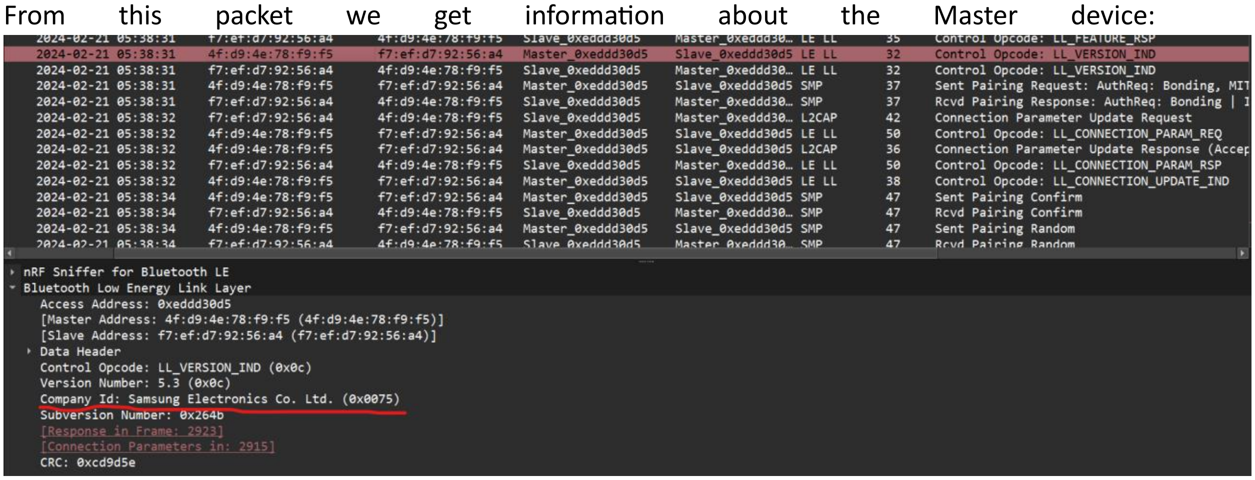
LL_FEATURE_REQ
Purpose: This opcode is used by a Bluetooth Low Energy (LE) Link Layer master device to request information about the supported features of a slave device.

Data Sent: The master device sends a packet containing the LL_FEATURE_REQ opcode and an identifier for the specific features it's interested in.
Response: The slave device responds with an LL_FEATURE_RSP packet containing information about its supported features.

LL_FEATURE_RSP
Purpose: This opcode is used by a Bluetooth LE slave device to respond to an LL_FEATURE_REQ from a master device.
Data Sent: The slave device sends a packet containing the LL_FEATURE_RSP opcode and a list of its supported features.
Response: The master device doesn't send a direct response, but it uses the information in the LL_FEATURE_RSP packet to determine how to proceed with the connection.

LL_VERSION_IND
Purpose: This opcode is used by a Bluetooth Low Energy (LE) Link Layer slave device to indicate its supported Bluetooth Low Energy version to the master device.
Data Sent: The slave device sends a packet containing the LL_VERSION_IND opcode and a single byte representing the BLE version it supports. The version numbers typically correspond to Bluetooth specifications (e.g., 0x08 for BLE 4.2).
Response: The master device should use this information to ensure compatibility with the slave device and potentially adjust its communication parameters accordingly.
Additional Notes:
The LL_VERSION_IND is typically exchanged during the connection establishment process, allowing devices to determine if they can communicate using compatible versions.
Some older devices might not support this opcode, relying on older methods for version negotiation.
In some cases, the slave device might send multiple LL_VERSION_IND packets during the connection if its supported versions change due to feature negotiations.

From this packet we get information about the Master device:



Since, it says “Samsung Electronics Co. Ltd.” in the Company ID it helps us verify that the Mouse is connected to the right device (Device-2).

4. Next the sequence in short is :

2024-02-21 05:38:31	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	32	Control Opcode: LL_VERSION_IND
2024-02-21 05:38:31	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	SMP	37	Sent Pairing Request: AuthReq: Bonding, MIT
2024-02-21 05:38:31	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	SMP	37	Rcvd Pairing Response: AuthReq: Bonding | I
2024-02-21 05:38:32	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	L2CAP	42	Connection Parameter Update Request
2024-02-21 05:38:32	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	50	Control Opcode: LL_CONNECTION_PARAM_REQ
2024-02-21 05:38:32	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	L2CAP	36	Connection Parameter Update Response (Accep
2024-02-21 05:38:32	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	50	Control Opcode: LL_CONNECTION_PARAM_RSP
2024-02-21 05:38:32	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	38	Control Opcode: LL_CONNECTION_UPDATE_IND
2024-02-21 05:38:34	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	SMP	47	Sent Pairing Confirm
2024-02-21 05:38:34	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	SMP	47	Rcvd Pairing Confirm
2024-02-21 05:38:34	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	SMP	47	Sent Pairing Random
2024-02-21 05:38:34	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	SMP	47	Rcvd Pairing Random

Device-2 sends Pairing request to Mouse.
Mouse receives it, and responds.
Connection parameter update request by Mouse to Device-2.
Device-2 accepts the request and responds.
Device-2 updates the connection parameter.

Then there are some SMP packets:
Device-2 sends pairing confirm
Mouse receives.

5. Some connection encryption related traffic:

f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	49	Control Opcode: LL_ENC_REQ
4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	39	Control Opcode: LL_ENC_RSP
4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	27	Control Opcode: LL_START_ENC_REQ
f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	27	Control Opcode: LL_START_ENC_RSP
4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	27	Control Opcode: LL_START_ENC_RSP

In Bluetooth LE, LL_ENC_REQ stands for Link Layer Encryption Request. It's a control opcode used by the master device to initiate the encryption process with a connected slave device.

Purpose:
To start encrypting the communication between the master and slave devices. This improves security by ensuring the data exchanged is confidential and cannot be intercepted by unauthorized parties.

6. After that,

```
SMP     47    Rcvd Pairing Random
LE LL   49    Control Opcode: LL_ENC_REQ
LE LL   39    Control Opcode: LL_ENC_RSP
LE LL   27    Control Opcode: LL_START_ENC_REQ
LE LL   27    Control Opcode: LL_START_ENC_RSP
LE LL   27    Control Opcode: LL_START_ENC_RSP
ATT     37    Sent Read By Type Request, Server Support
SMP     47    Rcvd Encryption Information
SMP     41    Rcvd Master Identification
SMP     47    Rcvd Identity Information
SMP     38    Rcvd Identity Address Information
ATT     53    Rcvd Find By Type Value Request, Primary
ATT     35    Rcvd Error Response - Attribute Not Found
ATT     35    Sent Error Response - Attribute Not Found
ATT     39    Sent Find By Type Value Request, Primary
ATT     39    Rcvd Find By Type Value Request, Primary
ATT     35    Rcvd Find By Type Value Response
```

The mouse sends acknowledgement that it has received the encryption data. And updated accordingly.

After that we start getting many ATT packets.

7. From amongst all the ATT packets:

Value	Time	HWsrcADD	HWdestADD	Source	Destination	Protocol	Length	Info
	2024-02-21 05:38:36	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	39	Rcvd Read Response, Handl
	2024-02-21 05:38:36	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	ATT	35	Sent Write Request, Handl
	2024-02-21 05:38:36	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	31	Rcvd Write Response, Hand
	2024-02-21 05:38:36	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	ATT	35	Sent Write Request, Handl
	2024-02-21 05:38:36	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	31	Rcvd Write Response, Hand
	2024-02-21 05:38:36	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	ATT	35	Sent Write Request, Handl
	2024-02-21 05:38:36	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	31	Rcvd Write Response, Hand
	2024-02-21 05:38:36	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	ATT	35	Sent Write Request, Handl
	2024-02-21 05:38:36	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	31	Rcvd Write Response, Hand
	2024-02-21 05:38:36	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	ATT	35	Sent Write Request, Handl
	2024-02-21 05:38:36	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	31	Rcvd Write Response, Hand
	2024-02-21 05:38:38	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	50	Control Opcode: LL_CONNEC
	2024-02-21 05:38:38	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	LE LL	50	Control Opcode: LL_CONNEC
	2024-02-21 05:38:38	4f:d9:4e:78:f9:f5	f7:ef:d7:92:56:a4	Master_0xeddd30d5	Slave_0xeddd30d5	LE LL	38	Control Opcode: LL_CONNEC
010000000000	2024-02-21 05:38:39	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	39	Rcvd Handle Value Notific
000000000000	2024-02-21 05:38:39	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	39	Rcvd Handle Value Notific
010000000000	2024-02-21 05:38:42	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	39	Rcvd Handle Value Notific
000000000000	2024-02-21 05:38:42	f7:ef:d7:92:56:a4	4f:d9:4e:78:f9:f5	Slave_0xeddd30d5	Master_0xeddd30…	ATT	39	Rcvd Handle Value Notific

The mouse-actions are the only last 4 packets in the above image. This is for the set of actions: Left-Click button pressed twice.

These 4 packets have the following as information:

```
Rcvd Handle Value Notification, Handle: 0x001e (Human Interface Device: Report)
Rcvd Handle Value Notification, Handle: 0x001e (Human Interface Device: Report)
Rcvd Handle Value Notification, Handle: 0x001e (Human Interface Device: Report)
Rcvd Handle Value Notification, Handle: 0x001e (Human Interface Device: Report)
```

5. Comparison amongst the Packet Sequence for the LED Lamp and the Bluetooth mouse traffics

BLE MOUSE TRAFFIC SEQUENCE	
ADV_IND, SCAN_REQ, SCAN_RSP	LE LL
CONNECT_IND	
Control Opcode: LL_FEATURE_REQ LL_FEATURE_RSP LL_VERSION_IND	LE LL
Sent Pairing Rqst... Recvd Pairing Response...	SMP
Connection Parameter Update Rqst Connection Parameter Update Rsp (Accepted)	L2CAP
Control Opcode: LL_CONNECTION_PARAM_REQ LL_CONNECTION_PARAM_RSP LL_CONNECTION_UPDATE_IND	LE LL
Sent Pairing Confirm Rcvd Pairing Confirm Sent Pairing Random Rcvd Pairing Random	SMP
Control Opcode: LL_ENC_REQ LL_ENC_RSP LL_START_ENC_REQ LL_START_ENC_RSP	LE LL
Server Supported Features	ATT
Rcvd Encryption Information Rcvd Master Information Rcvd Identity Information Rcvd Identity Address Information	SMP
Sent Read By Type Rqst Sent Read By Type Resp Sent Find Information Rqst Rcvd Find Information Resp Sent Read Blob Rqst Rcvd Read Block Resp Sent Write Rqst Sent Write Resp	ATT
Control Opcode: LL_CONNECTION_PARAM_REQ LL_CONNECTION_PARAM_RSP LL_CONNECTION_UPDATE_IND	LE LL
Rcvd Handle Value Notificaton (HID Report)	ATT

LED Bluetooth Lamp Traffic Sequence	
ADV_IND, SCAN_REQ, SCAN_RSP	ADV_IND
CONNECT_IND	LE LL
Control Opcode: LL_FEATURE_REQ LL_FEATURE_RSP LL_CONNECTION_UPDATE_IND LL_VERSION_IND	LE LL
Connection Parameter Update Request Connection Parameter Update Rsp (Accepted)	L2CAP
Sent Pairing Rqst Rcvd Pairing Rsp	SMP
LL_VERSION_IND	LE_LL
Sent Pairing Confirm Rcvd Pairing Confirm Sent Pairing Random Rcvd Pairing Random	SMP
Control Opcode: LL_ENC_REQ LL_ENC_RSP LL_START_ENC_REQ LL_START_ENC_RSP	LE LL
Sent Read By Type Rqst Sent Read By Type Resp	ATT
Rcvd Encryption Information Rcvd Master Information Rcvd Identity Information Rcvd Identity Address Information	SMP
Sent Find By Type Value Rqst	ATT
Sent Read By Type Rqst Sent Read By Type Resp Sent Read By Group Type Rqst Sent Read By Group Type Resp Sent Find Information Rqst Rcvd Find Information Rsp Sent Write Rqst Rcvd Write Resp	ATT
Sent Write Command, Handle Recvd Handle Value Notification, Handle	ATT
Sent Write Command, Handle	ATT

LED Lamp Packet Data for turning light Red:

```
7:1f:00:e2:05   Master_0x679ac498   Slave_0x679ac498 ATT   40   56ff000000f0aa   Sent Write Command, Handle: 0x0009 (Unknown: Unknown)

▸ Frame 7613: 40 bytes on wire (320 bits), 40 bytes captured (320 bits) on interface COM8-4.2, id 0
▸ nRF Sniffer for Bluetooth LE
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Attribute Protocol
  ▸ Opcode: Write Command (0x52)
  ▸ Handle: 0x0009 (Unknown: Unknown)
    Value: 56ff000000f0aa
```

BLE Mouse Packet Data for Left Click:

```
4e:78:f9:f5   Slave_0xeddd30d5   Master_0xeddd30… ATT   39   010000000000   Rcvd Handle Value Notification, Handle: 0x001e (Human Interface Device: Report)

▸ Frame 4301: 39 bytes on wire (312 bits), 39 bytes captured (312 bits) on interface COM10-4.2, id 0
▸ nRF Sniffer for Bluetooth LE
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Attribute Protocol
  ▸ Opcode: Handle Value Notification (0x1b)
  ▸ Handle: 0x001e (Human Interface Device: Report)
  ▸ Value: 010000000000
```

LED Lamp Packet Frame data for turning light Red:

```
▾ Frame 7613: 40 bytes on wire (320 bits), 40 bytes captured (320 bits) on interface COM8-4.2, id 0
    Section number: 1
  ▾ Interface id: 0 (COM8-4.2)
      Interface name: COM8-4.2
      Interface description: nRF Sniffer for Bluetooth LE COM8
    Encapsulation type: nRF Sniffer for Bluetooth LE (186)
    Arrival Time: Feb 25, 2024 10:33:07.137060000 Central Standard Time
    UTC Arrival Time: Feb 25, 2024 16:33:07.137060000 UTC
    Epoch Arrival Time: 1708878787.137060000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.000230000 seconds]
    [Time delta from previous displayed frame: 1.687971000 seconds]
    [Time since reference or first frame: 64.005039000 seconds]
    Frame Number: 7613
    Frame Length: 40 bytes (320 bits)
    Capture Length: 40 bytes (320 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: nordic_ble:btle:btl2cap:btatt]
```

BLE Mouse Packet Frame Data for Left Click:

```
▾ Frame 4301: 39 bytes on wire (312 bits), 39 bytes captured (312 bits) on interface COM10-4.2, id 0
    Section number: 1
  ▾ Interface id: 0 (COM10-4.2)
      Interface name: COM10-4.2
      Interface description: nRF Sniffer for Bluetooth LE COM10
    Encapsulation type: nRF Sniffer for Bluetooth LE (186)
    Arrival Time: Feb 20, 2024 23:38:42.391236000 Central Standard Time
    UTC Arrival Time: Feb 21, 2024 05:38:42.391236000 UTC
    Epoch Arrival Time: 1708493922.391236000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.000229000 seconds]
    [Time delta from previous displayed frame: 3.150021000 seconds]
    [Time since reference or first frame: 36.139975000 seconds]
    Frame Number: 4301
    Frame Length: 39 bytes (312 bits)
    Capture Length: 39 bytes (312 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: nordic_ble:btle:btl2cap:btatt]
```

Bluetooth Attribute Protocol (For LED Lamp):

```
▾ Bluetooth Attribute Protocol
  ▾ Opcode: Write Command (0x52)
      0... .... = Authentication Signature: False
      .1.. .... = Command: True
      ..01 0010 = Method: Write Request (0x12)
  ▾ Handle: 0x0009 (Unknown: Unknown)
      [Service UUID: Unknown (0xffd5)]
      [UUID: Unknown (0xffd9)]
    Value: 56ff000000f0aa
```

Bluetooth Attribute Protocol (For BLE Mouse):

```
▾ Bluetooth Attribute Protocol
  ▾ Opcode: Handle Value Notification (0x1b)
      0... .... = Authentication Signature: False
      .0.. .... = Command: False
      ..01 1011 = Method: Handle Value Notification (0x1b)
  ▾ Handle: 0x001e (Human Interface Device: Report)
      [Service UUID: Human Interface Device (0x1812)]
      [UUID: Report (0x2a4d)]
  ▾ Value: 010000000000
    ▾ [Expert Info (Note/Undecoded): Undecoded]
        [Undecoded]
        [Severity level: Note]
        [Group: Undecoded]
```

6. Active Attack Using ESP32 / Mouse Emulation Attack

The data packets we captured using Wireshark between the actual Bluetooth mouse and Device-2 can be incredibly valuable for simulating pre-defined data packets when developing our ESP32-based mouse.

Analyzing captured packets:

We identify relevant packets: Open the captured data file in Wireshark. Look for packets related to the mouse's functionality. These might be labeled with terms like "HID," "Mouse," or specific vendor IDs for our mouse brand.

<u>Packet structure:</u> Within those packets, we pay close attention to the payload section. This section typically carries the actual data about the mouse events, like click type (left, right, scroll), movement delta (X and Y movement distances), and button press/release status.

<u>Data format:</u> We note the format of the data within the payload. It might be binary, hexadecimal, or even text-based depending on the specific protocol used.

Simulating packets with ESP32:

<u>Extract data:</u> Based on our analysis, we extract the relevant data from the payload of specific events (clicks, movement) we want to simulate. This data represents the specific values the ESP32 needs to send to replicate those events.

Code implementation: In our ESP32 code, utilize libraries like BLEPeripheral to establish a BLE connection and create characteristic values that represent different mouse events.

<u>Data payload:</u> Within the characteristic values, we use the extracted data from the captured packets to define the payload content. This ensures our ESP32 sends data that aligns with the format and information expected by the computer.

Benefits of using captured packets:

Provides a concrete reference for what your ESP32 needs to send to replicate actual mouse events.

Reduces the need for guessing or reverse-engineering the protocol entirely.

Helps ensure our simulated data is correctly formatted and interpreted by the computer.

Limitations to consider:

Captured packets might not be fully comprehensive. We might need to experiment with different data combinations to achieve desired behavior.

Some data within the packets might be specific to our original mouse and might not be universally compatible.

Conclusion:

Using Wireshark and captured data packets can significantly simplify and streamline the initial development phase of our ESP32-based Bluetooth mouse. By analyzing the communication between our existing mouse and computer, we gain valuable insights into the data format and structure required to replicate its functionality. We need to remember to consider the potential limitations and be prepared to adjust and experiment as we develop our attacker mouse.