Lab 3 Template

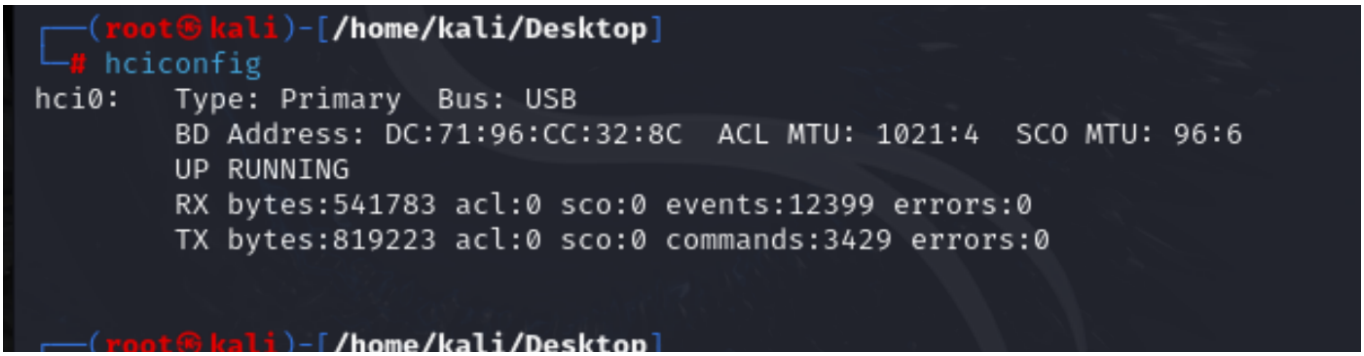
Summary

Read the entire lab manual and briefly state the objective of the lab in 4-5 sentences.

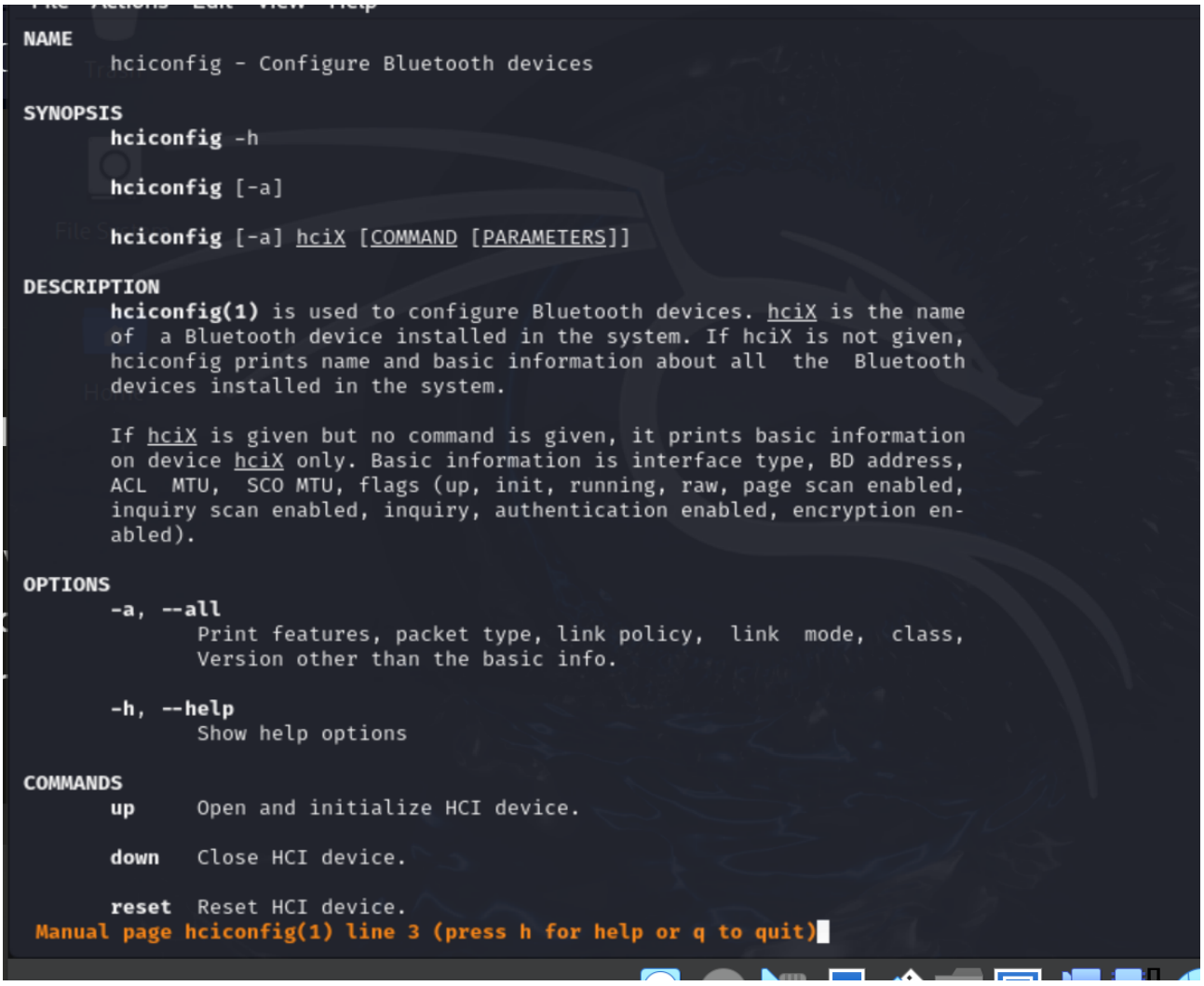
- The point of this lab is to learn about practical Bluetooth security reconnaissance with Kali Linux tools. The lab aims to teach us how to identify, scan, and analyze Bluetooth devices using tools like `hcitool`, `l2ping`, and `sdptool`. The lab demonstrates Bluetooth's security features including frequency hopping and key negotiation, while showing how devices can still be tracked and analyzed for vulnerabilities. Through the lab, both offensive testing and defensive security measures were of importance.

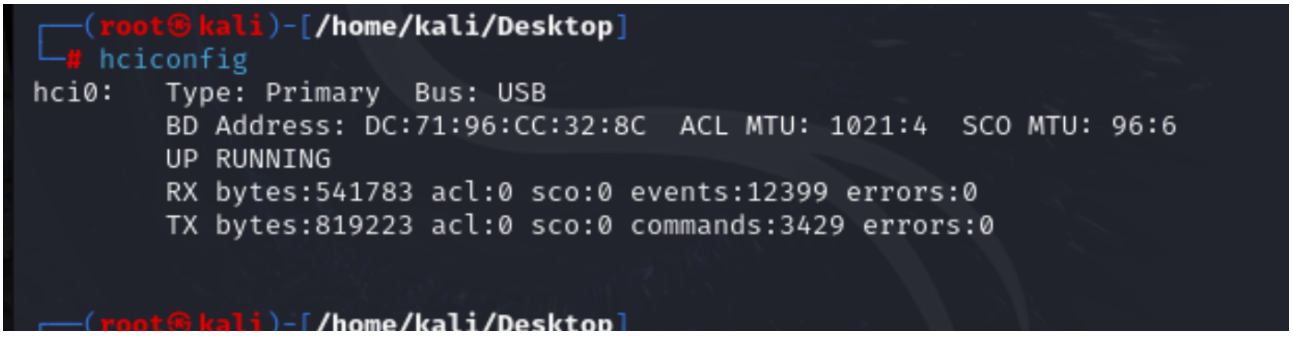
Step 1

Run the command `hciconfig`, and explain with the help of a screenshot, the output that you see including all parameters in the output



- The Type tells us whether its the main Bluetooth adapter or not. 'Bus' tells us that its connected via USB. The BD address is the MAC address of the adapter. the ACL MTU is the maximum transmission unit for ACL packets. SCO MTU is the same as ACL MTU but specifically for SCO packets. We see the device is 'Up and running'. Lastly, the RX/TX stats show received and transmitted bytes and the errors from the communication.
  Use the argument that you see in the man page to get the Bluetooth interface up and running and verify that this was successful by running `hciconfig`. Provide a screenshot to show this.

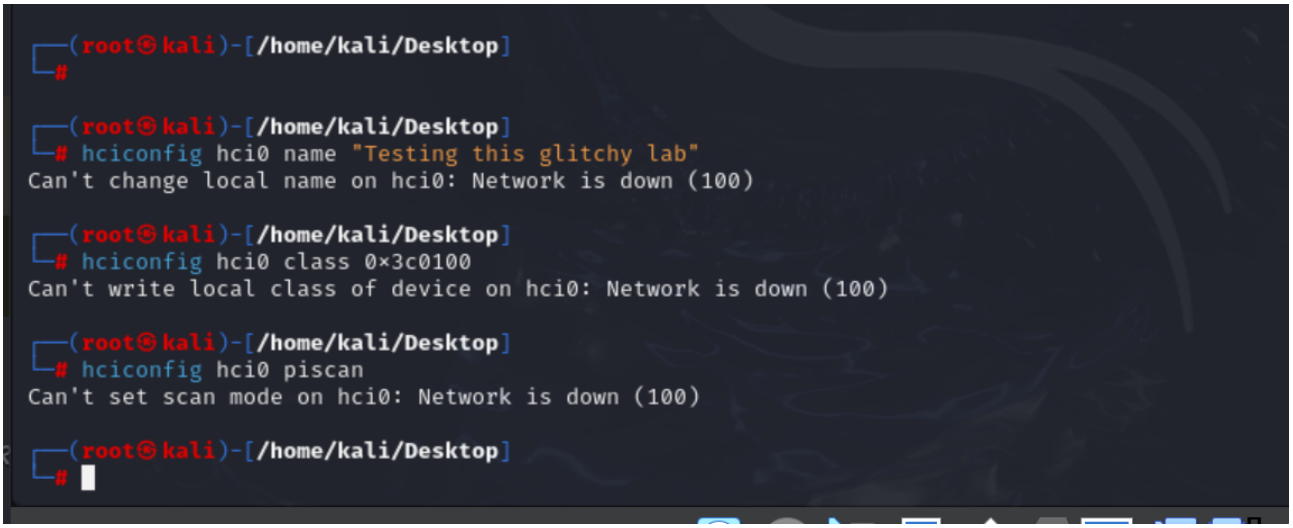




- A screenshot of the `hciconfig hci0 up` command missing. `hciconfig` after to confirm the device is up

Explore additional parameters of the `hciconfig` command. How can you modify the `hciconfig` command to change other settings of the Bluetooth adapter? Provide

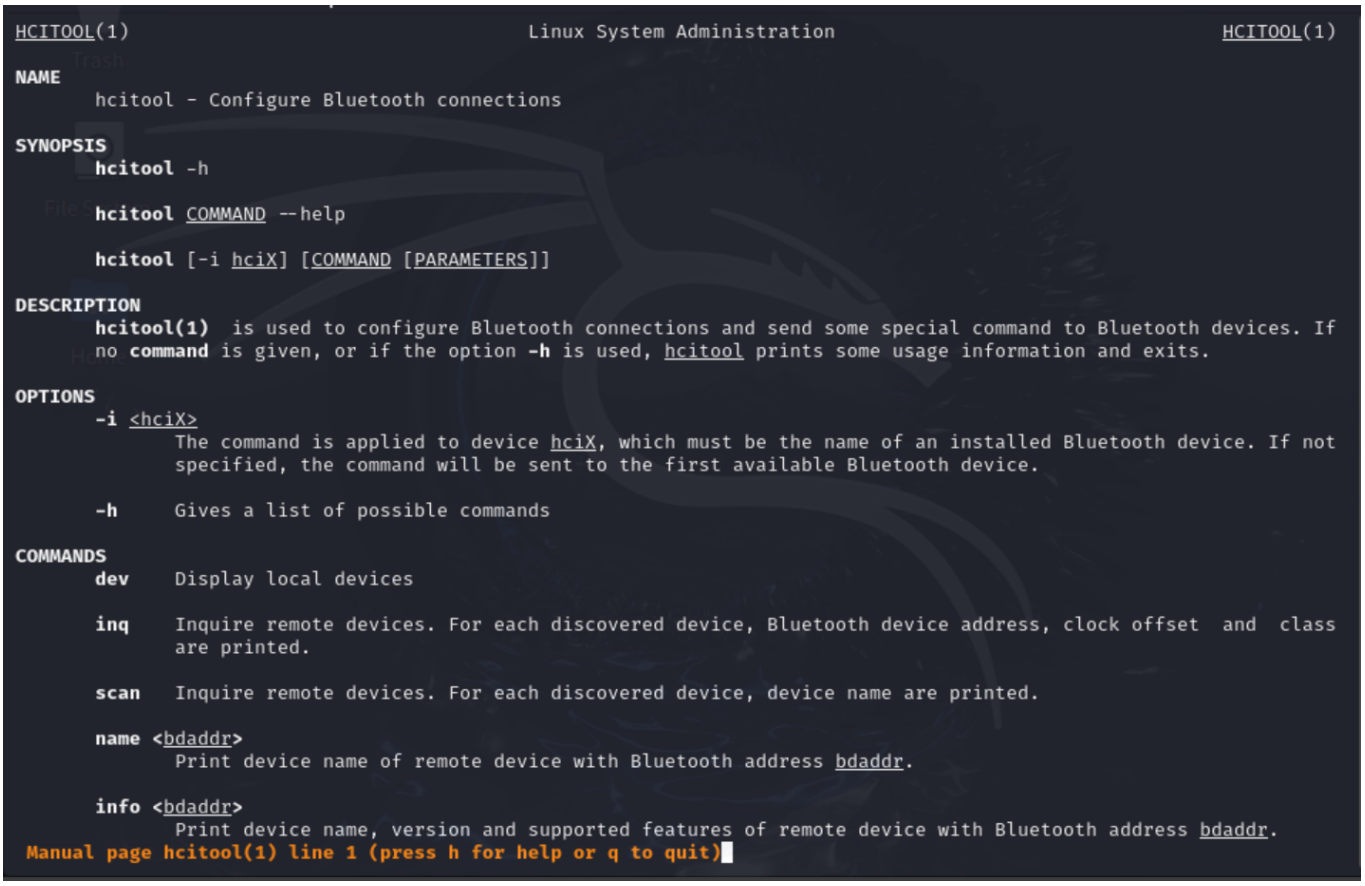
screenshots and explanations to back your exploration



- I was having major troubles getting the adapter connected and staying connected and was not able to get the working outputs of the above commands. Still, the first was meant to change the name of the device. The second command was meant to set the device class to 'desktop computer'. The third command is supposed t enable page and inquiry scans. These are all ways to modify the Bluetooth adapter settings.

Step 2

Check out its man page. What is the `hcitool` useful for?

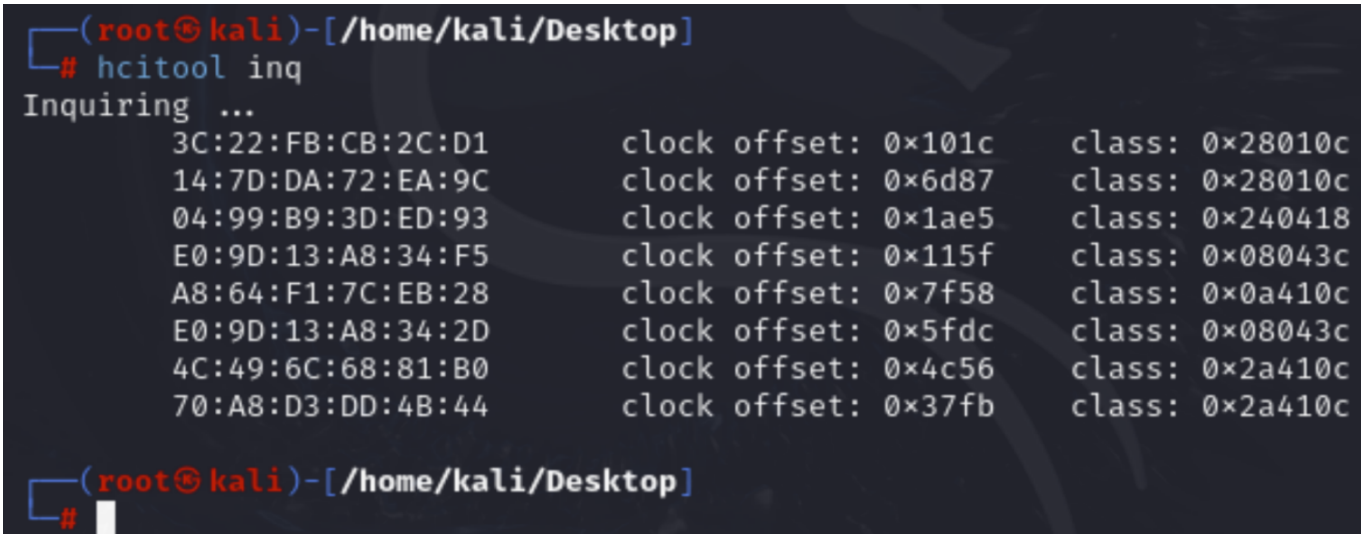


- The `hcitool` allows Bluetooth connections to be configued and is able to send special commands to Bluetooth deices. hcitool, by it self, simply prints usage information about the adapter.

First, run a scan to identify nearby Bluetooth devices and present their MAC addresses to do additional scans, inquiries, or attempt to get the name of the device.

- The scan was able to scan successfully and find names of devices nearby. I worked with my AirPods Pros. (inquire screenshot below)

To learn more, use the inq command. What additional information does the hcitool inq command provide compared to a basic scan and explain why this information may be useful



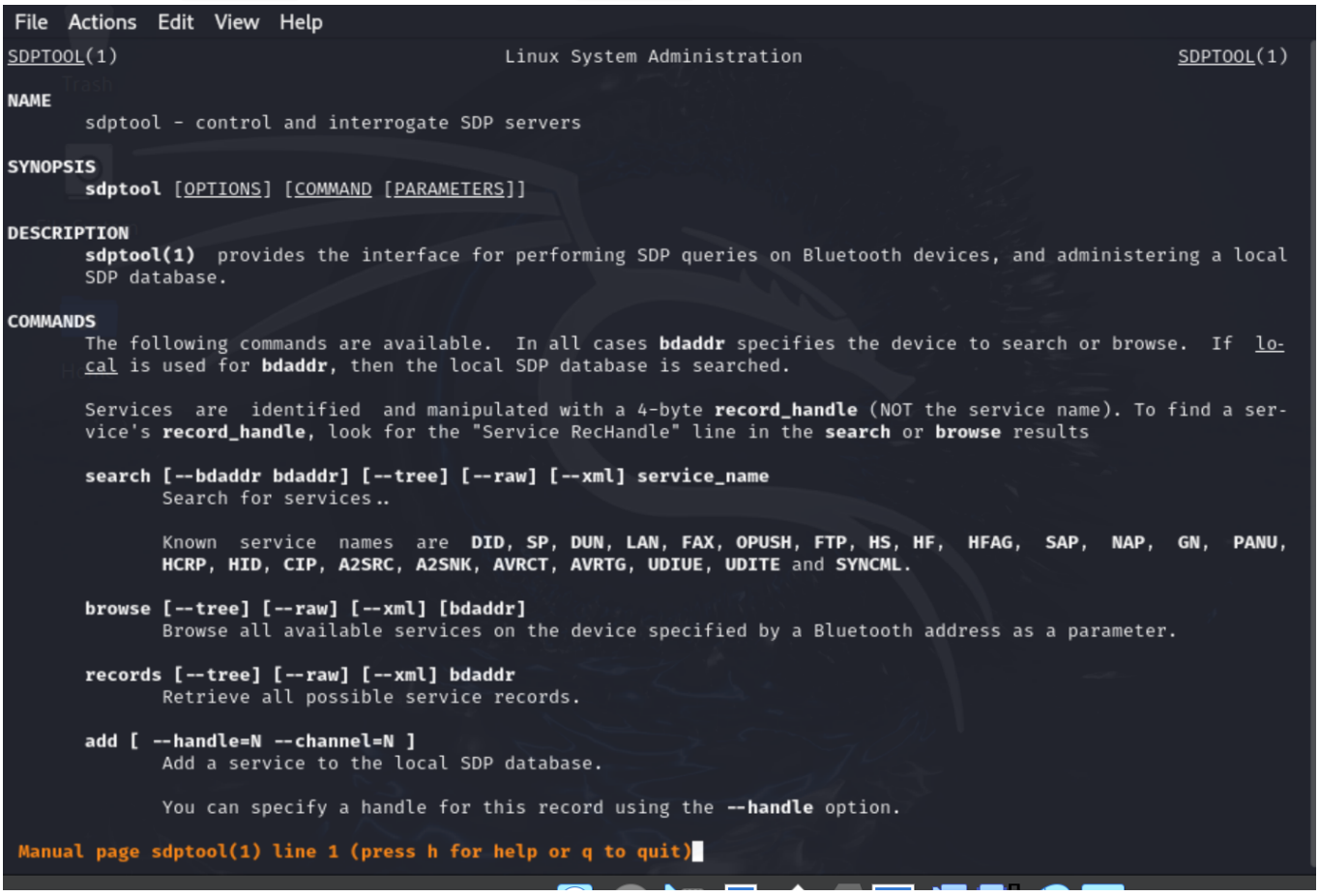
- The `hcitool inq` command gives two extra pieces of information compared to the basic scan. First is the clock offset which is timing information used for device synchronization. Second is its class which is the device type/capabilities in the form of a hexadecimal number (e.g., computer, phone, headphones).
  Explore other options for `hcitool` and provide insights into your findings, with explanations and screenshots

- My adapter stopped working again so I will just be explaining some `hcitool` options:
  - `hcitool dev` - Lists local Bluetooth devices
  - `hcitool con` - Shows active connections
  - `hcitool lescan` - Scans for BLE devices
  - `hcitool cc MAC` - Attempts connection
  - `hcitool auth MAC` - Authenticates with device
  - `hcitool info MAC` - Shows device info

    The all provide valuable recon data. `dev/con` provide system status, `lescan` scans for more modern IoT devices, `cc/auth` are for connection capabilities, and `info` shows extra device features and supported services.

Step 3

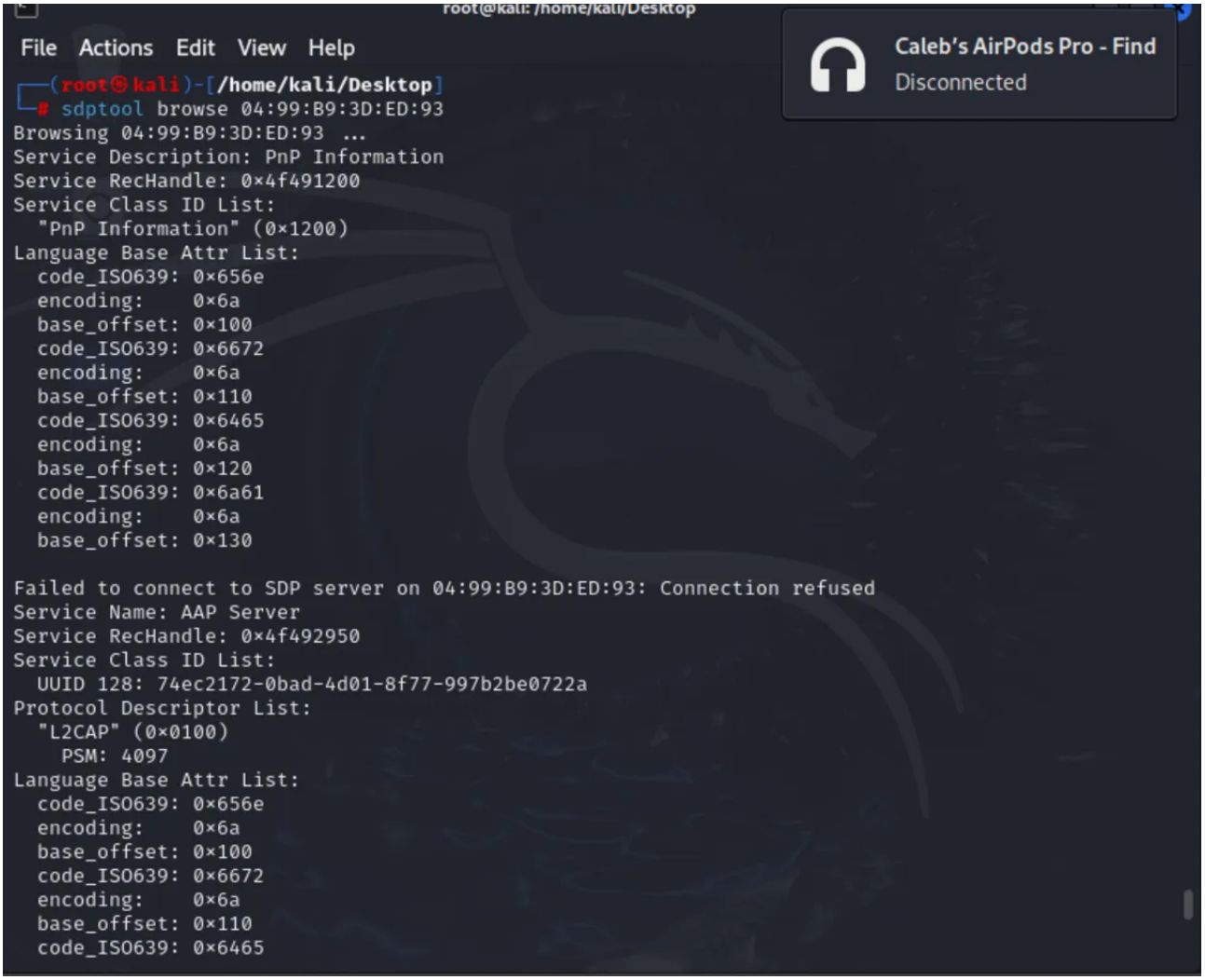
What is the `sdptool` used for? (Check `sdptool` man page)



- `sdptool` allows for performing SDP (Service Discovery Protocol) queries to have an interface on Bluetooth devices. It also will administer a local database on each device its used on.

  Quit the man page, and let's type sdptool then browse, followed by the MAC address we captured. Provide screenshots and explanations of what you see. Are you able to communicate with the device directly? Does the device use MAC

address randomization?



```
Service Name: Handsfree
Service RecHandle: 0x4f49111e
Service Class ID List:
  "Handsfree" (0x111e)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 7
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
  code_ISO639: 0x6672
  encoding:    0x6a
  base_offset: 0x110
  code_ISO639: 0x6465
  encoding:    0x6a
  base_offset: 0x120
  code_ISO639: 0x6a61
  encoding:    0x6a
  base_offset: 0x130
Profile Descriptor List:
  "Handsfree" (0x111e)
    Version: 0x0106

Service Name: AVRCP Target
Service RecHandle: 0x4f49110c
Service Class ID List:
  "AV Remote Target" (0x110c)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 23
  "AVCTP" (0x0017)
    uint16: 0x0104
Language Base Attr List:
```

```
Service Name: AVRCP Target
Service RecHandle: 0x4f49110c
Service Class ID List:
  "AV Remote Target" (0x110c)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 23
  "AVCTP" (0x0017)
    uint16: 0x0104
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
  code_ISO639: 0x6672
  encoding:    0x6a
  base_offset: 0x110
  code_ISO639: 0x6465
  encoding:    0x6a
  base_offset: 0x120
  code_ISO639: 0x6a61
  encoding:    0x6a
  base_offset: 0x130
Profile Descriptor List:
  "AV Remote" (0x110e)
    Version: 0x0105

Service Name: AVRCP Controller
Service RecHandle: 0x4f49110e
Service Class ID List:
  "AV Remote" (0x110e)
  "AV Remote Controller" (0x110f)
Protocol Descriptor List:
  "L2CAP" (0x0100)
```


PSM: 23
"AVCTP" (0×0017)
uint16: 0×0104
Language Base Attr List:


File Actions Edit View Help
base_offset: 0×120
code_ISO639: 0×6a61
encoding: 0×6a
base_offset: 0×130
Profile Descriptor List:
"AV Remote" (0×110e)
Version: 0×0105
Service Name: Audio Sink
Service RecHandle: 0×4f49110b
Service Class ID List:
"Audio Sink" (0×110b)
Protocol Descriptor List:
"L2CAP" (0×0100)
PSM: 25
"AVDTP" (0×0019)
uint16: 0×0103
Language Base Attr List:
code_ISO639: 0×656e
encoding: 0×6a
base_offset: 0×100
code_ISO639: 0×6672
encoding: 0×6a
base_offset: 0×110
code_ISO639: 0×6465
encoding: 0×6a
base_offset: 0×120
code_ISO639: 0×6a61
encoding: 0×6a
base_offset: 0×130
Profile Descriptor List:
"Advanced Audio" (0×110d)
Version: 0×0103
(root㉿kali)-[/home/kali/Desktop]
#

- All of my connection attempts failed with a "Connection refused" error. The out put still shows fruitful information like Pnp data and AAP server services. We can also see that the l2cap protocol is used (PSM: 4097). It looks like the the AirPods do use randomization after seeing the original MAC being different in previous scans.
  Do some research on the types of services typically listed by sdptool browse and which of these are most commonly associated with security vulnerabilities, and why?
- There are a few common vulnerable Bluetooth services identified by sdp tool:
  1. OBEX (Object Exchange):
     - File transfer vulnerabilities
     - Buffer overflow risks
     - Authentication bypass issues
  2. Serial Port Profile (SPP):
     - Unauthorized access to serial data
     - Man-in-the-middle attacks
     - Weak authentication
  3. Headset/Handsfree Profile:
     - Audio hijacking
     - Eavesdropping
     - Unauthorized connections
  4. Device ID Profile:
     - Information disclosure
     - Device fingerprinting
     - Version-specific exploits

     Of these services, OBEX seems to be the most critical due to the direct file system accessability and frequent implementation flaws. SPP follows in the same manner due to legacy authentication methods and use of clear-text data transmission.
     https://nvd.nist.gov/vuln/detail/CVE-2018-958
     https://www.bluetooth.com/learn-about-bluetooth/key-technologies/security/

Step 4

Just ping the Bluetooth device using its MAC address. Are any devices within range and reachable? Provide an explanation and screenshots to back this up

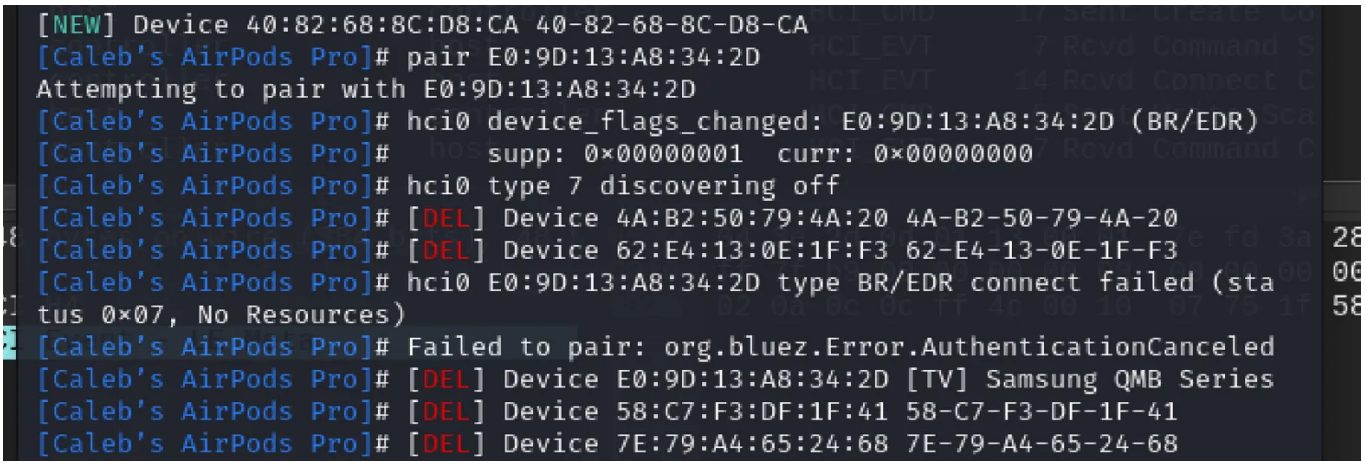
```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# l2ping 04:99:B9:3D:ED:93
Ping: 04:99:B9:3D:ED:93 from DC:71:96:CC:32:8C (data size 44) ...
0 bytes from 04:99:B9:3D:ED:93 id 0 time 19.74ms
0 bytes from 04:99:B9:3D:ED:93 id 1 time 15.01ms
0 bytes from 04:99:B9:3D:ED:93 id 2 time 23.61ms
0 bytes from 04:99:B9:3D:ED:93 id 3 time 15.60ms
0 bytes from 04:99:B9:3D:ED:93 id 4 time 14.91ms
0 bytes from 04:99:B9:3D:ED:93 id 5 time 18.75ms
^C6 sent, 6 received, 0% loss

┌──(root㉿kali)-[/home/kali/Desktop]
└─#
```

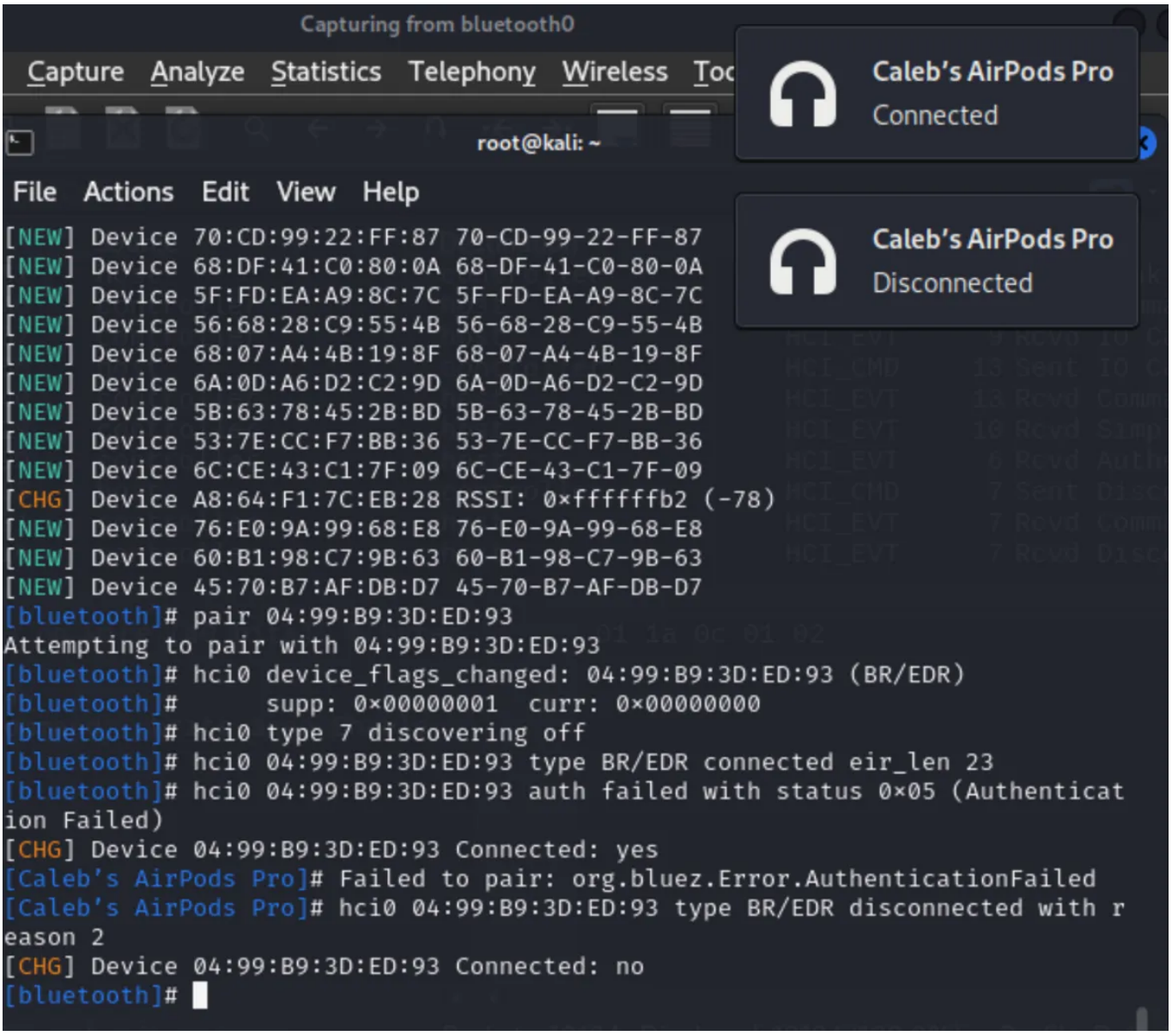
- The AirPods were found successfully. The earlier command `hcitool scan` was able to find any devices within range.

Step 5

When your chosen device appears in the list, use the MAC address to pair and connect to it (Screenshot)

```
┌──(root㉿kali)-[~]
└─# bluetoothctl
Waiting to connect to bluetoothd...[bluetooth]# Agent registered
[bluetooth]# power on
[bluetooth]# Changing power on succeeded
[bluetooth]# discoverable on
[bluetooth]# hci0 new_settings: powered connectable bondable ssp br/edr le se
cure-conn
[bluetooth]# hci0 new_settings: powered connectable discoverable bondable ssp
 br/edr le secure-conn
[bluetooth]# Changing discoverable on succeeded
[bluetooth]# [CHG] Controller DC:71:96:CC:32:8C Discoverable: yes
[bluetooth]# pairable on
[bluetooth]# Changing pairable on succeeded
[bluetooth]# scan on
[bluetooth]# SetDiscoveryFilter success
[bluetooth]# hci0 type 7 discovering on
[bluetooth]# Discovery started
[bluetooth]# [CHG] Controller DC:71:96:CC:32:8C Discovering: yes
[bluetooth]# [NEW] Device 64:FE:82:87:AE:6A 64-FE-82-87-AE-6A
[bluetooth]# [NEW] Device 0B:80:A1:E3:C7:DB 0B-80-A1-E3-C7-DB
[bluetooth]# [NEW] Device 64:D5:45:04:CF:99 64-D5-45-04-CF-99
[bluetooth]# [NEW] Device 4F:8E:F0:7B:E5:87 4F-8E-F0-7B-E5-87
[bluetooth]# [NEW] Device 44:18:4C:B8:C5:47 44-18-4C-B8-C5-47
[bluetooth]# [NEW] Device 65:1C:41:90:19:4F 65-1C-41-90-19-4F
```





1. **Observe the packets captured by wireshark during the pairing process, and provide screenshots and an explanation of what you see**
2. **Provide screenshots of notable packets, including explanations of any relevant details or identifiable information you see. What types of Bluetooth protocols did**

you observe in the captured traffic? Explain any security implications based on the captured data and any sensitive information you observed.

- Was not able to capture packets, but during a typical Bluetooth pairing process, wireshark would likely capture L2CAP, SDP, and HCI protocol data. Below are bits of information likely to be found in this scenario.
  Key Security-Related Packets:
  1. Pairing Request/Response
     - Pairing method capabilities
     - Security requirements
     - Key size preferences
  2. Authentication Stage
     - Random number challenges
     - Confirmation values
     - Link key creation
  3. Encryption Setup
     - Encryption key size negotiation
     - Start encryption sequence
     Security Implications:
- MITM attack opportunities during pairing
- Encryption key strength visible
- Device capability exposure
- Service enumeration possible
- Pairing method security level revealed

Research Question #1

Choose a specific Bluetooth vulnerability from a recent security report and outline a step-by-step attack plan, including reconnaissance, exploitation and post-exploitation. Assess the feasibility and ethical considerations of the attack. Use recent case studies or security bulletins to inform your scenario. Cite your sources.

I will be focusing on the Bluetooth Low Energy (BLE) BIAS Attack (2023). Starting with the Attack plan:

1. Reconnaissance
   - Scan for vulnerable BLE devices using `hcitool lescan`
   - Identify target's MAC and services via `sdptool`
   - Monitor pairing procedures with Wireshark
   - Verify device runs vulnerable firmware

2. Exploitation
    - Capture legitimate pairing attempt
    - Manipulate authentication responses
    - Force device into legacy pairing mode
    - Bypass authentication using BIAS technique
    - Establish unauthorized connection
3. Post-Exploitation
    - Access device services/data
    - Maintain persistent access
    - Potential for man-in-the-middle
    - Data exfiltration possibilities

This attack is much harder to carry out since there are a few requirements for the attack to be successful. First, the attacker must obviously be in physical proximity of the device and then the target must also be using a vulnerable BLE stack. This attack is not super feasible nowadays with it being detectable on many security monitoring devices, but with a success rate of ~80% on affected devices, its worth a shot!
Now, with every exploit comes ethical considerations. Like with many exploits & vulnerabilities, devices carrying them are at higher risk of unauthorized data access, potential service disruption, or long-term affects on critical systems. Privacy violations are very much a concern and more reason to negate such exploits.

Main Source: "BIAS: Bluetooth Impersonation AttackS" - IEEE Security & Privacy 2023 (https://ieeexplore.ieee.org/document/9152758)
https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/reporting-security/

Conclusion

Conclude your lab report with a brief 4-5 sentence summary of the lab that includes anything interesting you would like to mention, and what you learnt.

- Through lab 3, I gained experience with Bluetooth security analysis tools with Kali. though I encountered significant challenges with devices connectivity and authentication. I found the theoretical concepts of Bluetooth security to be insightful, particularly MAC randomization and frequency hopping, but the practical exercises were often inconsistent - especially when trying to connect to my AirPods which proved nearly impossible due to what I can only assume are strict security measures from Apple. The Wireshark packet capture part was also pretty problematic because establishing a stable connection for analysis was unreliable, but I guess this reminded me of how real-world security implementations can be. Despite all the challenges, I learned valuable information about

Bluetooth reconnaissance techniques and the need for robust security measures, even if some tools didn't work as smoothly as the lab manual suggested.