HTB Challenge: MiTM- Difficulty: Easy - Misc

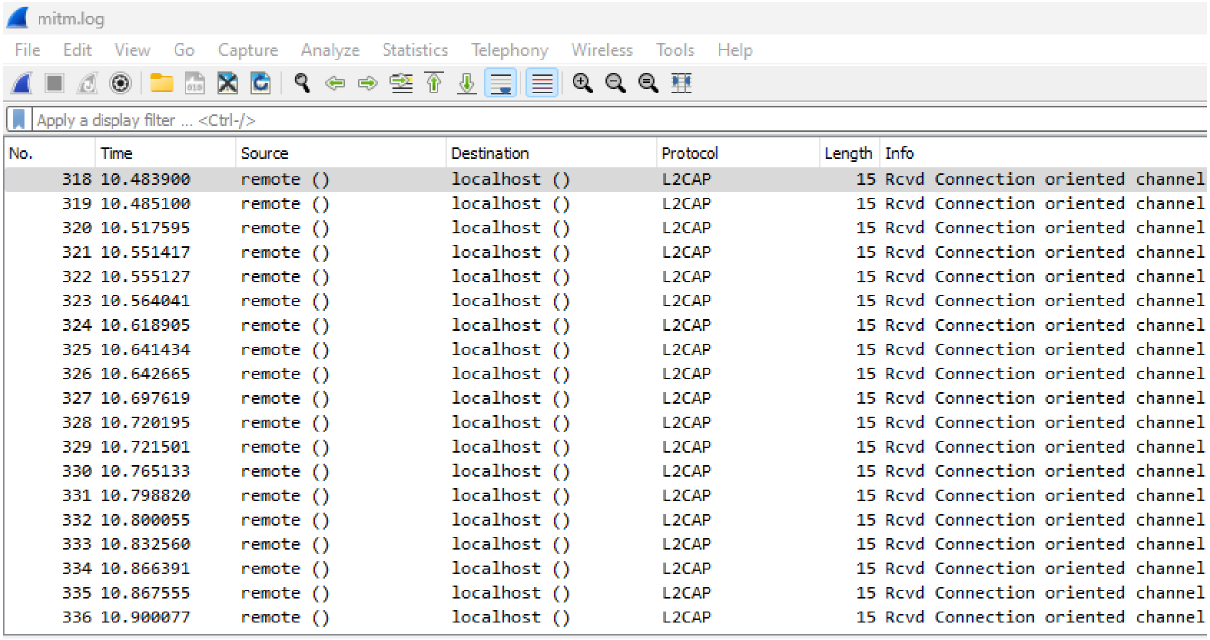
Erel Regev

# Table of Contents



# Intro

I received a log file called mitm.log. I opened it with wireshark:




| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 318 | 10.483900 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 319 | 10.485100 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 320 | 10.517595 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 321 | 10.551417 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 322 | 10.555127 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 323 | 10.564041 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 324 | 10.618905 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 325 | 10.641434 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 326 | 10.642665 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 327 | 10.697619 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 328 | 10.720195 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 329 | 10.721501 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 330 | 10.765133 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 331 | 10.798820 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 332 | 10.800055 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 333 | 10.832560 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 334 | 10.866391 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 335 | 10.867555 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 336 | 10.900077 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |

It looks like a Bluetooth communication.

# L2CAP

L2CAP frames consist of a header followed by payload data.

The header typically includes fields like Length, Channel ID, and more.

Header Fields:

Length Field: Indicates the total length of the L2CAP frame (header + payload).

Channel ID Field: Specifies the logical channel for multiplexing different protocols.

Payload:

The payload contains the actual data being transported, which could be from higher-layer protocols such as RFCOMM or SDP in the case of Bluetooth.

Refer to Bluetooth Specification:

Erel Regev

Consult the Bluetooth Core Specification for the version relevant to your application (e.g., Bluetooth 4.0, 5.0, etc.).

The Bluetooth Core Specification will provide detailed information about the structure of L2CAP frames and the meaning of each field.

Use a Protocol Analyzer:

Protocol analyzers, such as Wireshark with Bluetooth support or dedicated Bluetooth analyzers, can simplify the decoding process.

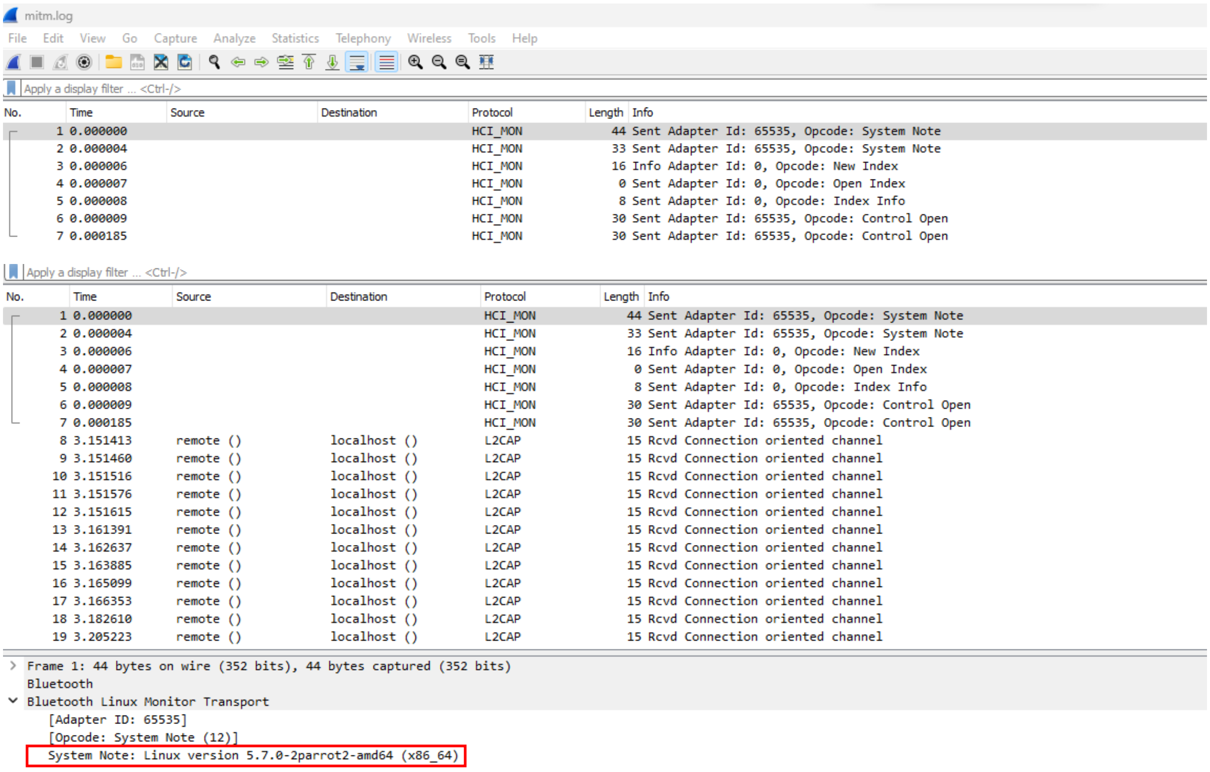
Analyzers often provide a user-friendly interface and interpret the frames for you.

Programming Tools:

If you're dealing with this programmatically, use programming languages like Python and libraries such as PyBluez.

Parse the header and payload fields according to the Bluetooth specification.

While investigating the packets, and focusing on the HCI_MON packets (first packets), system information and commands can be seen:



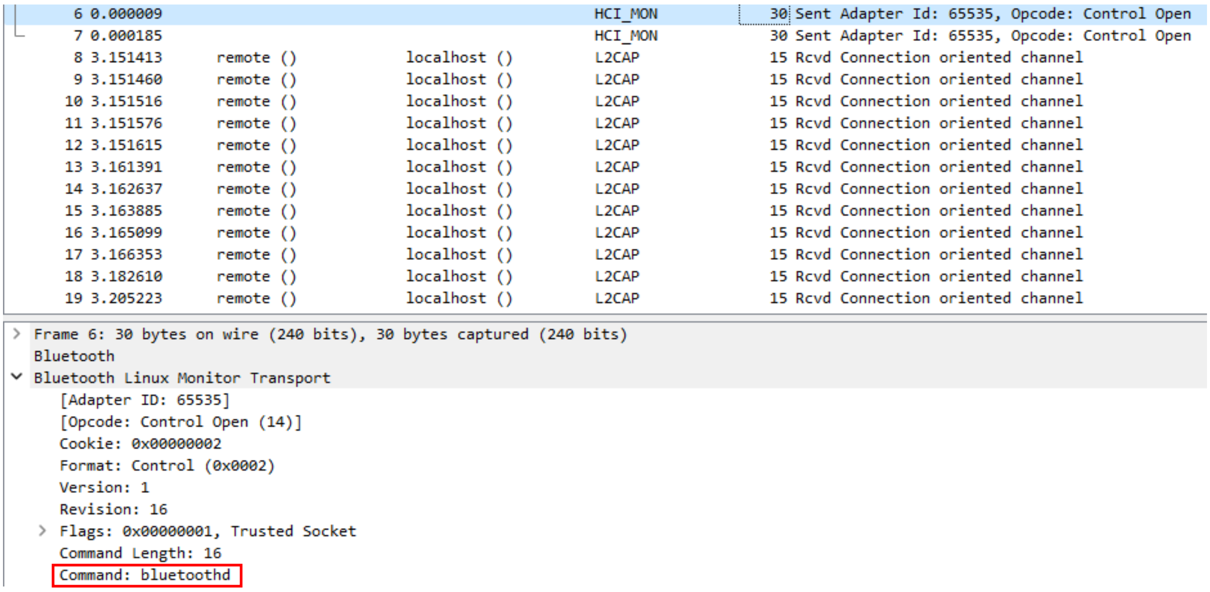
| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | | | HCI_MON | 44 | Sent Adapter Id: 65535, Opcode: System Note |
| 2 | 0.000004 | | | HCI_MON | 33 | Sent Adapter Id: 65535, Opcode: System Note |
| 3 | 0.000006 | | | HCI_MON | 16 | Info Adapter Id: 0, Opcode: New Index |
| 4 | 0.000007 | | | HCI_MON | 0 | Sent Adapter Id: 0, Opcode: Open Index |
| 5 | 0.000008 | | | HCI_MON | 8 | Sent Adapter Id: 0, Opcode: Index Info |
| 6 | 0.000009 | | | HCI_MON | 30 | Sent Adapter Id: 65535, Opcode: Control Open |
| 7 | 0.000185 | | | HCI_MON | 30 | Sent Adapter Id: 65535, Opcode: Control Open |
| 8 | 3.151413 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 9 | 3.151460 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 10 | 3.151516 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 11 | 3.151576 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 12 | 3.151615 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 13 | 3.161391 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 14 | 3.162637 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 15 | 3.163885 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 16 | 3.165099 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 17 | 3.166353 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 18 | 3.182610 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 19 | 3.205223 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |

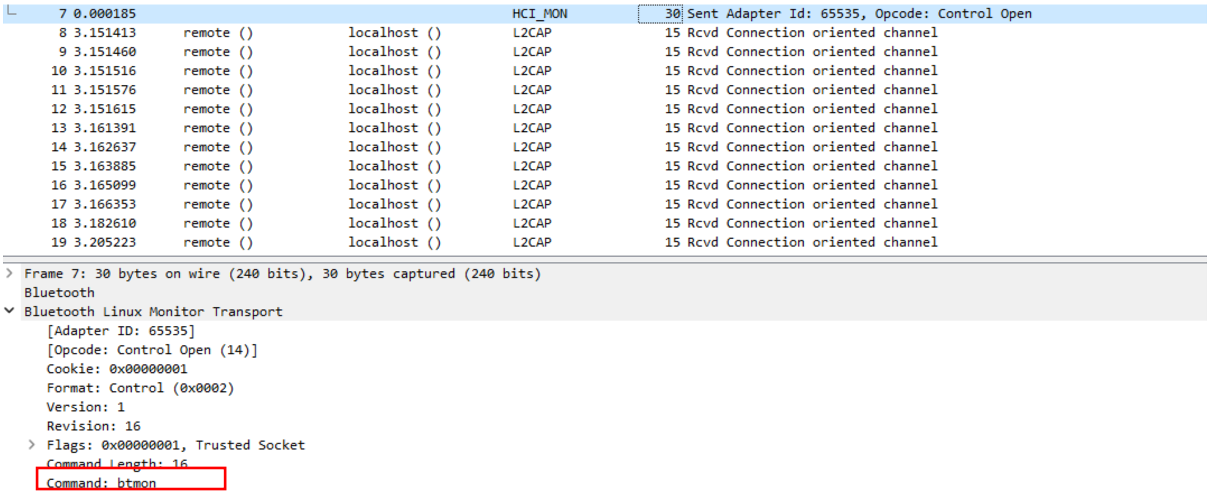
```
> Frame 2: 33 bytes on wire (264 bits), 33 bytes captured (264 bits)
  Bluetooth
v Bluetooth Linux Monitor Transport
    [Adapter ID: 65535]
    [Opcode: System Note (12)]
    System Note: Bluetooth subsystem version 2.22
```

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | | | HCI_MON | 44 | Sent Adapter Id: 65535, Opcode: System Note |
| 2 | 0.000004 | | | HCI_MON | 33 | Sent Adapter Id: 65535, Opcode: System Note |
| 3 | 0.000006 | | | HCI_MON | 16 | Info Adapter Id: 0, Opcode: New Index |
| 4 | 0.000007 | | | HCI_MON | 0 | Sent Adapter Id: 0, Opcode: Open Index |
| 5 | 0.000008 | | | HCI_MON | 8 | Sent Adapter Id: 0, Opcode: Index Info |
| 6 | 0.000009 | | | HCI_MON | 30 | Sent Adapter Id: 65535, Opcode: Control Open |
| 7 | 0.000185 | | | HCI_MON | 30 | Sent Adapter Id: 65535, Opcode: Control Open |
| 8 | 3.151413 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 9 | 3.151460 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 10 | 3.151516 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 11 | 3.151576 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 12 | 3.151615 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 13 | 3.161391 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 14 | 3.162637 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 15 | 3.163885 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 16 | 3.165099 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 17 | 3.166353 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 18 | 3.182610 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 19 | 3.205223 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |

```
> Frame 3: 16 bytes on wire (128 bits), 16 bytes captured (128 bits)
  Bluetooth
v Bluetooth Linux Monitor Transport
    [Adapter ID: 0]
    [Opcode: New Index (0)]
    Bus: BR/EDR (0x00)
    Type: USB (0x01)
    BD_ADDR: LiteonTe_e4:58:60 (f8:28:19:e4:58:60)
    Adapter Name: hci0
```

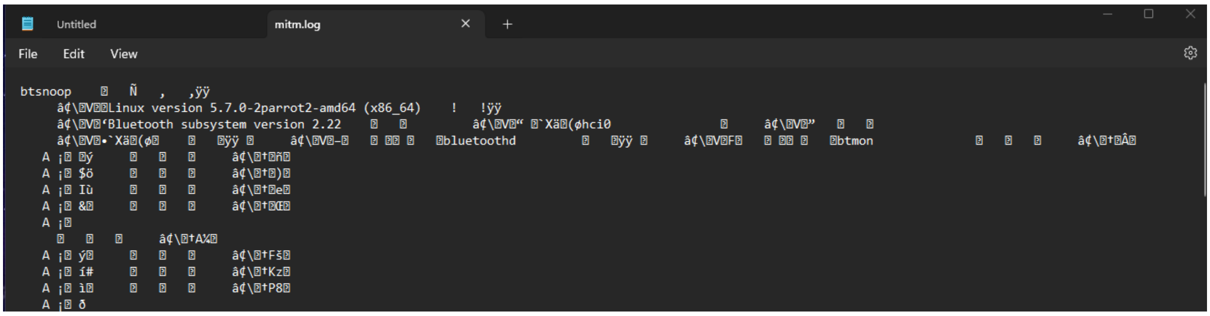
Erel Regev




Linux tool.

When viewing the file using notepad:




Btsnoop is a file format used to log Bluetooth communication between devices.

The btmon tool in Kali Linux is a Bluetooth monitor that captures Bluetooth HCI (Host Controller Interface) packets and displays them in a human-readable format. The output can be saved in a btsnoop file, which is a standard format for Bluetooth packet captures.

After understanding a little bit what is going on, its time to figure out how to work with the payloads transferred in the packets:

```
10 3.151516   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
11 3.151576   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
12 3.151615   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
13 3.161391   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
14 3.162637   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
15 3.163885   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
16 3.165099   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
17 3.166353   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
18 3.182610   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel
19 3.205223   remote ()   localhost ()   L2CAP   15 Rcvd Connection oriented channel

> Frame 10: 15 bytes on wire (120 bits), 15 bytes captured (120 bits)
> Bluetooth
v Bluetooth Linux Monitor Transport
    [Adapter ID: 0]
    [Opcode: ACL Rx Packet (5)]
> Bluetooth HCI ACL Packet
v Bluetooth L2CAP Protocol
    Length: 7
    CID: Dynamically Allocated Channel (0x0041)
    Payload: a1020049f90000
```

In order to do so, it is necessary to identify the devices within the file.

As mentioned before, each L2CAP packet has a Channel ID Field, which specifies the logical channel for multiplexing different protocols.

In Bluetooth communication, L2CAP provides logical channels for data transmission between two devices. These logical channels are identified by the Channel Identifier (CID). Each CID represents a unique communication channel between two devices, and multiple CIDs can exist simultaneously, allowing for concurrent communication streams.

L2CAP uses CIDs to multiplex and demultiplex different higher-layer protocols and services over a single Bluetooth connection. For example, you might have one CID for control messages and another for data transmission.

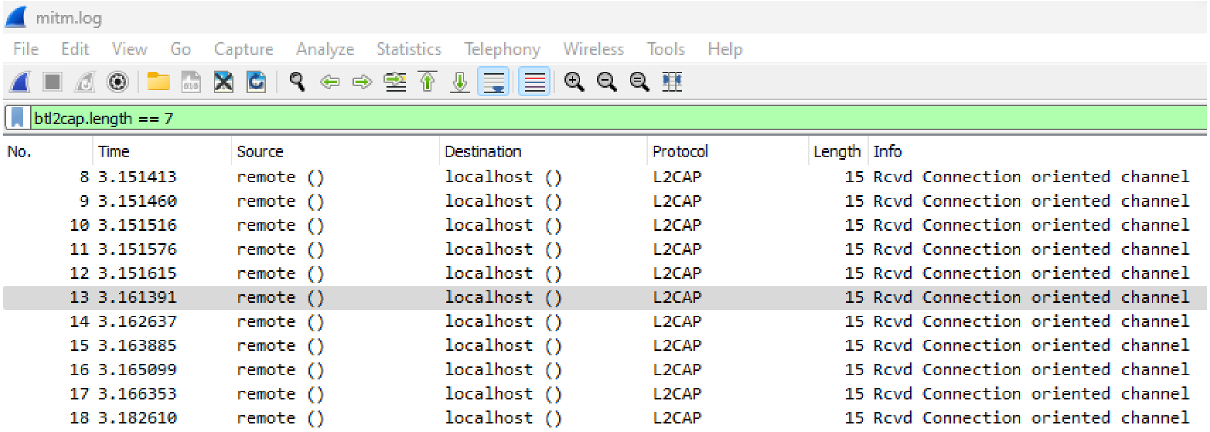
CID values are negotiated during the L2CAP connection establishment phase, and both devices agree on the CIDs that will be used for different purposes. The CIDs provide a way to distinguish between different types of data or protocols being transmitted over the Bluetooth link.

When inspecting the payloads, I was focusing on the length of the payload:

- 7 Bytes
- 9 Bytes

```
┌──(kali㉿kali)-[~/Desktop]
└─$ tshark -r mitm.log -Y btl2cap -T fields -e 'btl2cap.length' | sort | uniq
7
9
```

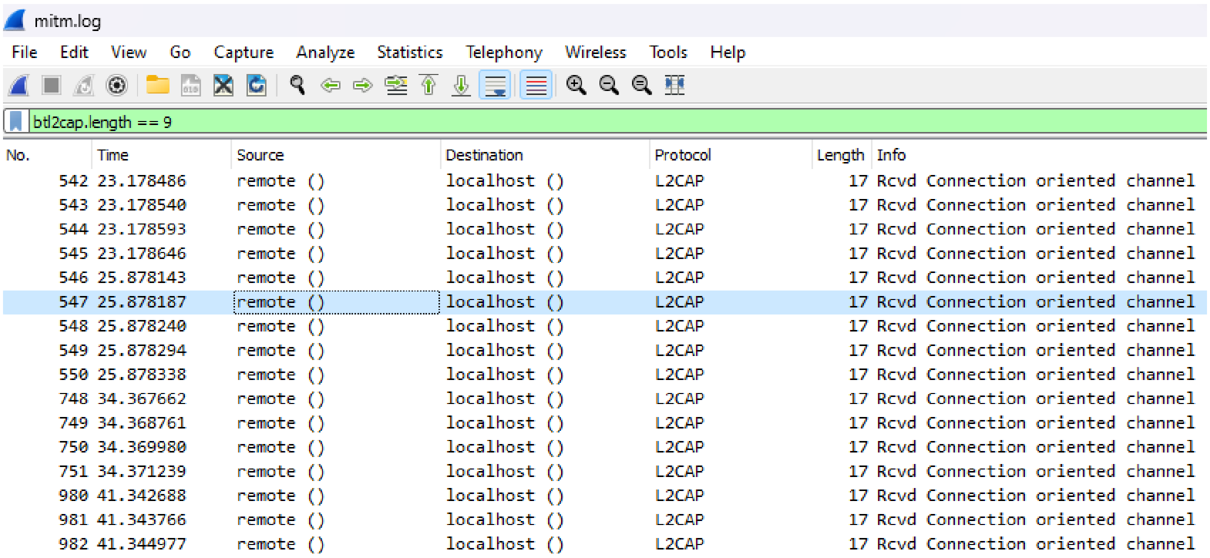
Erel Regev




| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 8 | 3.151413 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 9 | 3.151460 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 10 | 3.151516 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 11 | 3.151576 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 12 | 3.151615 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 13 | 3.161391 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 14 | 3.162637 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 15 | 3.163885 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 16 | 3.165099 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 17 | 3.166353 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |
| 18 | 3.182610 | remote () | localhost () | L2CAP | 15 | Rcvd Connection oriented channel |

When filtering the file for packets with 7 bytes payloads, we can see that packets with this length are 97.9% of the file:


When filtering the file for packets with 9 bytes payload:




| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 542 | 23.178486 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 543 | 23.178540 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 544 | 23.178593 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 545 | 23.178646 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 546 | 25.878143 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 547 | 25.878187 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 548 | 25.878240 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 549 | 25.878294 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 550 | 25.878338 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 748 | 34.367662 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 749 | 34.368761 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 750 | 34.369980 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 751 | 34.371239 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 980 | 41.342688 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 981 | 41.343766 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |
| 982 | 41.344977 | remote () | localhost () | L2CAP | 17 | Rcvd Connection oriented channel |

1.9% of the total packets, while the rest are the HCI_MON packets mentioned before.

Another thing that caught my eye is the time column. It is possible to see that it sends many packets within a very short period of time.

As well, L2CAP can hold up to 64Kb payloads, so why would it use only 7 or 9? What devices probably send short payloads within a very short time?

Mouse and keyboard.

Solving this had me connect BT keyboard and mouse and sniff the network using Wireshark.

Erel Regev

When testing the functionality, it was possible to see that:

When pressing the "a" key I get the following payload - "a1010000040000000000" and when I press "A" (with shift key) I get the payload "a1010200040000000000".

When pressing the "b" key I get the following payload - "a1010000050000000000" and when I press "B" (with shift key) I get the payload "a1010200050000000000".

When pressing the "c" key I get the following payload - "a1010000060000000000" and when I press "C" (with shift key) I get the payload "a1010200060000000000".

The conclusion is that the shift key being represented by the third byte (0x20), while the pressed key itself is represented by the 5th byte.

Note that letters are being represented as bytes starting from 4.

The keyboard encoding we are dealing with is Scancode.

https://deskthority.net/wiki/Scancode

## Decoding

First we must extract the payloads and save it to a new file:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ tshark -r mitm.log -T fields -e btl2cap.payload | grep -e "^a101"| grep -v "000000000000"

a10100002800000000
a10100002800000000
a10102000b00000000
a10102001700000000
a10102000500000000
a10102002f00000000
a10102000e00000000
a10100002000000000
a10100001c00000000
a10102001600000000
a10100001700000000
a10102001500000000
a10100002700000000
a10100000e00000000
a10100000800000000
a10102001600000000
a10102002d00000000
a10102000600000000
a10100002700000000
a10100001000000000
a10102001300000000
a10100001500000000
a10100002700000000
a10100001000000000
a10100001e00000000
a10100001600000000
a10100002000000000
a10100000700000000
a10102003000000000
```

```python
        '37':['.', '>'],
        '38':['/', '?'],
        '39':['⇪','⇪'],
        '4f':[u'→',u'→'],
        '50':[u'←',u'←'],
        '51':[u'↓',u'↓'],
        '52':[u'↑',u'↑']
}
flag = ""
with open('./keys.txt', "r") as f:
    for line in f:
        tab = line[4:6]
        key = line[8:10].upper()
        if tab == "20":
            flag += KEY[key][1]
        else:
            flag += KEY[key][0]
print(flag)
```

**Dictionary Definition (KEY)**

The dictionary KEY maps hexadecimal key codes to pairs of characters.

Each key code has two associated characters: the first one is used when the "tab" value is not equal to "20," and the second one is used when the "tab" value is equal to "20."

For example, the key code '04' corresponds to the pair ['a', 'A'].

**Flag Construction Loop**

The script opens the 'keys.txt' file for reading.

It iterates through each line in the file.

For each line, it extracts substrings corresponding to "tab" and "key" values.

If the "tab" value is equal to "20," it appends the second character of the corresponding key code pair to the flag.

Otherwise, it appends the first character of the corresponding key code pair to the flag.

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/Challenges/MiTM/misc_mitm]
└─$ python decode.py

htb[k[redacted]3d]
```