

Etiqueteuse Supvan E10

Bonjour :)

Il y a quelque temps, j'ai acheté une simple étiqueteuse Bluetooth sur [Amazon](#) et elle fonctionne avec une application Android nommée "[Katasymbol](#)".

Cette application limite grandement ce qu'il est possible de faire avec l'imprimante, par exemple, il n'est pas possible de répéter x fois un texte (pour faire une étiquette autour d'un câble par exemple).

Je me suis donc lancé dans le reverse de cette application mobile ainsi que la communication Bluetooth.

Exploration

J'ai d'abord commencé par récupérer l'APK et le lire avec [Jadx](#), c'est un outil permettant de décompiler le byte code java.

Je cherche notamment où sont envoyés les paquets Bluetooth pour ensuite remonter aux fonctions liées à l'affichage, l'impression, etc.

```
292     public boolean sendCmdStartTrans(byte b, int i, int i2, byte[] bArr) {
293         int i3 = 0;
294         Arrays.fill(bArr, (byte) 0);
295         byte[] bArr2 = new byte[16];
296         bArr2[0] = 126;
297         bArr2[1] = 90;
298         bArr2[2] = 12;
299         bArr2[3] = 0;
300         bArr2[4] = UnionPtg.sid;
301         bArr2[5] = 1;
302         bArr2[6] = -86;
303         bArr2[7] = b;
304         bArr2[10] = 0;
305         bArr2[11] = 1;
306         bArr2[12] = (byte) i;
307         bArr2[13] = (byte) (i >> 8);
308         bArr2[14] = (byte) i2;
309         bArr2[15] = (byte) (i2 >> 8);
310         for (int i4 = 10; i4 < 16; i4++) {
311             i3 += bArr2[i4] & UByte.MAX_VALUE;
312         }
313         bArr2[8] = (byte) i3;
314         bArr2[9] = (byte) (i3 >> 8);
315         if (this.mBluetoothMode == this.CLASSIC_BLUETOOTH) {
316             return classicBluetoothStatus(bArr2, bArr);
317         }
318         return BLEBluetoothStatus(b, bArr2, bArr);
319     }
```

La fonction sendCmdStartTrans contient un tableau bArr2 qui correspond à un paquet Bluetooth (RFCOMM) qui est une simple liaison série en Bluetooth.

Les nombres dans le tableau sont la représentation des bytes.

J'ai maintenant voulu analyser les trames Bluetooth faites par mon téléphone quand il communique avec l'étiqueteuse.

En utilisant un téléphone rooté et avec la partie Bluetooth HCI snoop activé dans le mode développeur, il faut taper cette commande en ADB :

```
adb shell su -c "'nc -s 127.0.0.1 -p 8872 -L system/bin/tail -f -c +0  
/data/log/bt/btsnoop_hci.log' &"
```

Cela permet ensuite de voir les trames dans Wireshark.

8163	867.204302000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	RFCOMM	13 Sent SABM Channel=0
8164	867.209950000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	13 Rcvd UA Channel=0
8165	867.209355000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	RFCOMM	23 Sent UIH Channel=0 -> 1 MPX_CTRL DLC Parameter Negotiation (PN)
8167	867.213912000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	23 Rcvd UIH Channel=0 -> 1 MPX_CTRL DLC Parameter Negotiation (PN)
8168	867.214192000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	RFCOMM	13 Sent SABM Channel=1 (Serial Port)
8169	867.220183000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	13 Rcvd UA Channel=1
8170	867.220537000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	RFCOMM	17 Sent UIH Channel=0 -> 1 MPX_CTRL Modem Status Command (MSC)
8172	867.225131000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	17 Rcvd UIH Channel=0 -> 1 MPX_CTRL Modem Status Command (MSC)
8173	867.226913000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	17 Rcvd UIH Channel=0 -> 1 MPX_CTRL Modem Status Command (MSC)
8174	867.227182000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	RFCOMM	17 Sent UIH Channel=0 -> 1 MPX_CTRL Modem Status Command (MSC)
8185	867.960584000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	SPP	30 Sent UIH Channel=1 UID
8186	867.968228000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	SPP	73 Rcvd UIH Channel=1
8187	867.972929000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	14 Rcvd UIH Channel=1 UID
8188	867.984126000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	SPP	30 Sent UIH Channel=1 UID
8190	867.990679000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	SPP	36 Rcvd "Z\023\000\020\003U\030\004\000\000\000\000\000\000\004\000\000\000\000\000"
8191	867.994161000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	14 Rcvd UIH Channel=1 UID
8192	868.032089000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	SPP	30 Sent UIH Channel=1 UID
8193	868.039022000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	SPP	39 Rcvd UIH Channel=1
8194	868.042634000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	14 Rcvd UIH Channel=1 UID
8196	868.316447000	SamsungElect_c7:b4:...	SupvanInform_73:f8:...	SPP	30 Sent UIH Channel=1 UID
8197	868.322080000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	SPP	39 Rcvd UIH Channel=1
8198	868.326448000	SupvanInform_73:f8:...	SamsungElect_c7:b4:...	RFCOMM	14 Rcvd UIH Channel=1 UID

Grace au code de l'application mobile ainsi que des trames bluetooth, j'ai commencé à reproduire certaines fonctions via un petit programme python (récupération du nom de l'appareil, du firmware, etc).

Sur le [dépôt Git](#).

J'ai fait des fonctions qui permettent de convertir une trame en hexadécimal vers des bytes puis vers la représentation des bytes en nombres.

Revision #1

Created 2 June 2025 19:57:01 by Matthieu

Updated 2 June 2025 20:18:32 by Matthieu