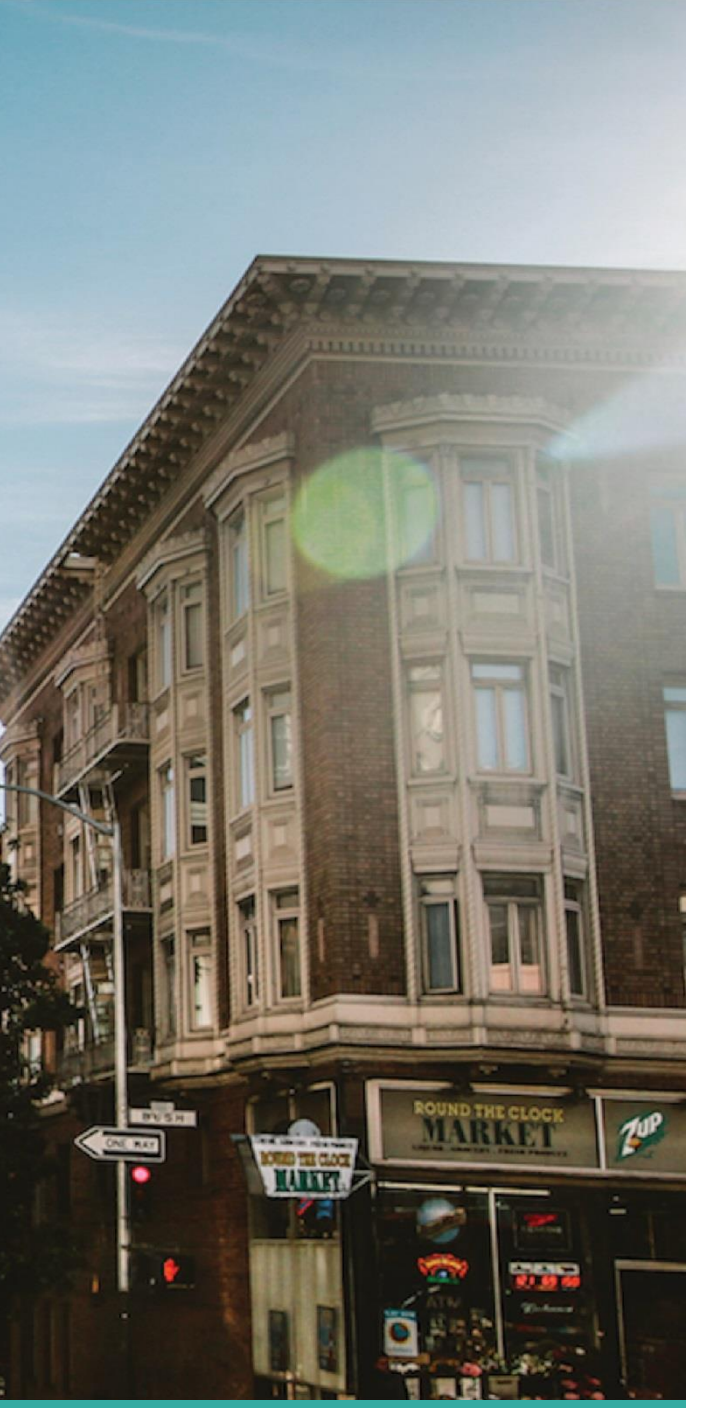
Sécurité Matérielle USB1_2



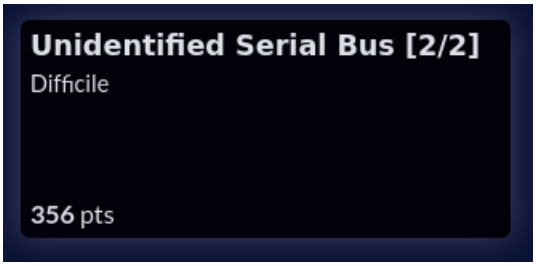
22 MAI

404CTF2025

Créé par : JOL

Unidentified Serial Bus 2_2

USB Partie 2 sur 2



Voici la méthode appliquée par **USB2_flag_recover_v2.py**, étape par étape, avec le “pourquoi” derrière chaque choix :

1. **Chargement & pré-traitement**

- On lit les deux traces **float32** D+ et D-.
- On fabrique un signal logique **state = sign(D+ − D−)** (et on teste aussi l’inverse **sign(D− − D+)** au cas où la polarité serait inversée).

2. **Estimation de la période de bit**

- On fait les longueurs de runs sur state (durées où l’état ne change pas).
- La **période de bit** est estimée comme le **mode** des runs dans une fenêtre plausible (≈ 5–60 échantillons). Dans ce challenge elle tombe ≈ **20 échantillons/bit**.

3. **Repérage des trames (SYNC → EOP)**

- On détecte les débuts de trame via le motif **SYNC** observé dans le domaine des runs : **7 runs ≈ T** puis **1 run ≈ 2T** (représentant le KJKJKJKK de l’USB 1.1).
- On repère la fin de trame à l’**EOP** grâce à **SE0** (les deux lignes à bas niveau), via un simple seuil (ex. D+ < ~0.15 ET D- < ~0.15).
- Pour chaque début SYNC, la trame est **[SYNC sauté] → jusqu’au prochain SE0** (on ne coupe pas sur un faux SYNC au milieu des données).

4. **Décodage binaire**

- On **échantillonne au centre** de chaque cellule de bit : indices start + phase + T//2, step T.
- On applique **NRZI** : **pas de transition = 1**, **transition = 0** (en comparant l’état échantillonné au pas précédent).
- **Bit stuffing** : **désactivé** ici (UNSTUFF=False) car inutile sur ce challenge (les données ASCII sortent déjà propres).
- On assemble les bits **LSB-first** par octets.

5. **Interprétation USB minimale**

- Le **premier octet** d'une trame est le **PID** ; on vérifie la cohérence (nibble et son complément) pour classer : **DATA0/DATA1**, **IN/OUT/SETUP**, etc.
- Pour les **DATA0/DATA1**, on prend le **payload** = octets[1:] **moins les 2 derniers** (supposés **CRC16**).
  On ne recalcule pas les CRC : on s'en moque pour l'extraction du flag.

6. **Récupération du texte & sélection robuste**

- On **concatène tous les octets ASCII imprimables** de **toutes** les trames **DATA0/DATA1** (erreur classique évitée : ne pas se limiter à 1 octet par trame).
- On balaie **toutes les phases** 0..T−1 **et** les **deux polarités** ; pour chaque combinaison on reconstruit le texte et on **score** :
  - priorité 1 : présence d'un **flag complet** 404CTF{[0-9a-f]{64}},
  - priorité 2 : longueur de l'HEXA **partiel** visible après 404CTF{,
  - priorité 3 : **longueur totale** du texte ASCII.
- On retient la combinaison (phase, polarité) au **meilleur score** et on sort le texte + le flag.

7. **Pièges évités (les causes du "Lf s 4CT{…}")**

- **Ne prendre qu'1 octet par trame** → texte tronqué. Corrigé en parcourant **tous** les octets du payload.
- **Mauvaise phase** → bits faux → lettres manquantes. Corrigé par **balayage complet** des phases.
- **Polarité inversée** → tout le flux "décalé". Corrigé en testant **les deux polarités**.
- **Bit-stuffing inutile** et **CRC** : laissés de côté (juste **retirer 2 octets** de fin pour le CRC16 des DATA), conformément au retour des solveurs.

En pratique, on est sur du **Full-Speed (12 Mb/s)** tel que défini par **USB 1.1** — et **repris à l'identique** dans la base spec **USB 2.0** (mêmes paquets IN/OUT/SETUP, PIDs, NRZI, SYNC KJKJKJKK, bit stuffing, etc.). Donc tu peux te référer **sans risque** à la **USB 2.0 Base Specification** pour la couche protocolaire (notamment le chapitre « Protocol Layer », qui décrit bien SYNC et le format des paquets), ou à la **USB 1.1** si on préfère l'originale : les définitions FS/LS y sont équivalentes. usb.orgu.dianyuan.comesd.cs.ucr.eduni.com

Docs à utiliser (officielles / primaires)

- **USB 2.0 Base Specification** – page de la spec (le ZIP contient la spec et les ECN ; voir le chap. 8 « Protocol Layer » pour les paquets, NRZI, SYNC/EOP). usb.org
- **USB 1.1 Specification (1998)** – révision historique couvrant FS/LS (mêmes règles de trame, NRZI, SYNC).

(À noter : USB 2.0 **remplace** USB 1.1 et ajoute simplement le mode **High-Speed** ; tout ce qui est **Full-Speed** reste valide et inchangé.)

Au final, sans connaissance profonde du protocole, cette procédure “signal → SYNC/EOP → NRZI → bytes → DATA0/1 ASCII” suffit à reconstruire la phrase et à extraire le flag :

404CTF{9f993d54e688927dbfad50d6980c4b3dbf61991ba06fbe707409d699c724116b}

En pratique, À partir des traces brutes D+/D−, on estime la période de bit, on balaye phase et polarité pour échantillonner au centre, on découpe chaque trame de SYNC→EOP (SE0), on décode NRZI puis assemble en octets ASCII(LSB-first), on lit tous les payloads DATA0/DATA1 (en ignorant bit-stuffing/CRC dans ce défi)