

HELMo Campus Guillemins

# Rapport – Firmware Extraction Tapo C100

UE46 : Sécurité de l'IOT – 3S1 B3 Cyber sécurité

Julien Diet – Thomas Huppe

17/12/2025

## Table des matières

1. Introduction.....	3
2. Accès au Shell root via le port série (UART) .....	4
2.1 Mise en place du matériel .....	4
2.1.1 Configuration et test préalable : .....	4
2.2 Ouverture de la caméra et repérage du port série.....	5
2.2.1 Démontage du dispositif : .....	5
2.2.2 Repérage de l'interface UART : .....	5
2.3 Lecture sur le port série .....	6
2.3.1 Câblage et Connexions :.....	6
2.3.2 Configuration logicielle et Démarrage : .....	7
2.4 Obtention du Shell root.....	7
2.4.1 Identification et Authentification : .....	7
2.4.2 Validation de l'accès : .....	7
2.5 Exploration du File System.....	8
2.5.1 Localisation des partitions :.....	8
2.5.2 Vérification : .....	8
2.6 Recherche de certificats et clés cryptographiques .....	9
2.6.1 Inventaire automatisé.....	9
2.6.2 Vérification du contenu.....	9
3. Extraction du Firmware via la mémoire flash .....	10
3.1 Mise en place du matériel .....	10
3.1.1 Connexion physique du programmeur .....	11
3.2 Identification et spécifications du composant mémoire .....	12
3.3 Configuration du logiciel d'extraction (IMSProg).....	12
3.4 Procédure de lecture et analyse du dump.....	13
3.4.1 Lecture physique du composant.....	13
3.4.2 Analyse de la structure avec Unblob .....	13
3.4.3 Identification du système de fichiers (SquashFS).....	14
3.5 Décompression et analyse du système de fichiers (RootFS).....	14
3.5.1 Extraction du contenu SquashFS.....	14
3.5.2 Exploration de l'arborescence .....	15
3.5.3 Recherche de certificats et clés cryptographiques.....	15
4. Conclusion .....	16
5. Bibliographie.....	17

# 1. Introduction

Ce rapport décrit les manipulations techniques réalisées lors du laboratoire consacré à l'analyse matérielle d'un dispositif IoT. Le sujet de ce laboratoire est la caméra de surveillance **TP-Link Tapo C100**.



L'objectif principal de ce travail est de parvenir à extraire le firmware de l'appareil afin d'en comprendre la structure et le fonctionnement interne.

Pour atteindre cet objectif, nous avons mis en œuvre deux méthodes d'extraction distinctes :

1. **L'accès via le port série (UART)** : Cette méthode consiste à se connecter à l'interface de communication de la carte mère pour interagir avec le système d'exploitation et localiser les fichiers du firmware.
2. **L'extraction physique de la mémoire (Flash Dump)** : Cette méthode implique l'utilisation d'un matériel spécifique pour lire directement le contenu brut de la puce de stockage.

Ce document détaille chronologiquement les étapes suivies, depuis le démontage de la caméra jusqu'à la récupération des données.

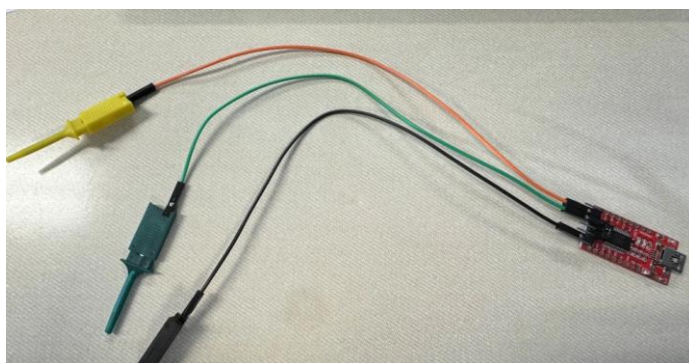
## 2. Accès au Shell root via le port série (UART)

Cette partie décrit la procédure suivie pour obtenir un accès privilégié (root) au système d'exploitation de la caméra en exploitant son interface série.

### 2.1 Mise en place du matériel

Pour établir la communication série avec la caméra, le matériel suivant a été utilisé :

- **Un adaptateur USB-TTL (FTDI FT232R)** : Ce module est indispensable pour convertir les signaux série (UART) de la caméra en signal USB interprétable par l'ordinateur de travail.
- **Câblage et connectique** : Un câble Mini USB vers USB-C pour relier l'adaptateur à l'ordinateur, ainsi que des sondes PCB (PCB probes). Ces sondes permettent de maintenir une connexion électrique stable sur les points de test de la carte mère sans nécessiter de soudure.
- **Logiciel de terminal** : L'outil picocom a été installé sur le poste de travail pour lire le flux de données.

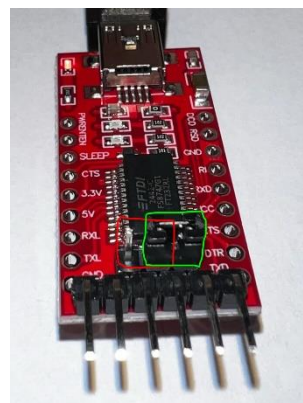


#### 2.1.1 Configuration et test préalable :

La caméra fonctionnant avec une tension logique de 3,3 V, le cavalier (jumper) de la carte FTDI a été positionné sur 3,3 V (et non 5 V) afin d'assurer la compatibilité des niveaux de tension.

On peut voir deux formes sur la photo :

- Le cadre rouge désigne l'emplacement pour 5 V
- Le cadre vert désigne l'emplacement pour 3,3V



Avant de connecter l'adaptateur à la cible, un test fonctionnel a été réalisé pour s'assurer de la bonne transmission des données (TX). Nous avons ouvert une session terminal avec la commande suivante :

```
picocom -b 115200 /dev/ttyUSB0
```

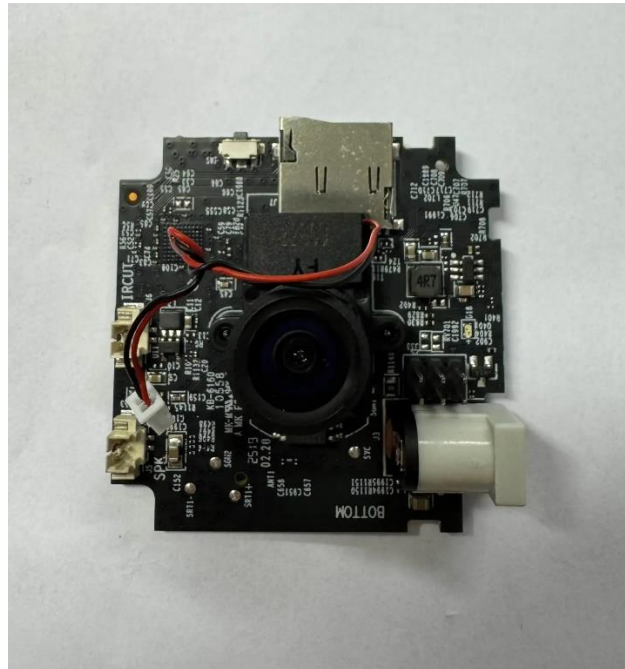
En saisissant des caractères arbitraires dans la console, nous avons observé le clignotement de la LED rouge sur le module FTDI. Cette confirmation visuelle valide que l'adaptateur reçoit bien les instructions de l'ordinateur et est capable d'émettre des signaux (Input).

## 2.2 Ouverture de la caméra et repérage du port série

Une fois le matériel de connexion prêt, la première étape a consisté à accéder physiquement à la carte électronique de la caméra.

### 2.2.1 Démontage du dispositif :

Le boîtier de la caméra étant clipsé, nous avons utilisé un outil d'ouverture en plastique (pry tool) pour écarter les fixations sans endommager le plastique. Une fois le boîtier ouvert, la carte mère (PCB) a été entièrement extraite de son logement afin de faciliter sa manipulation et l'accès aux composants.

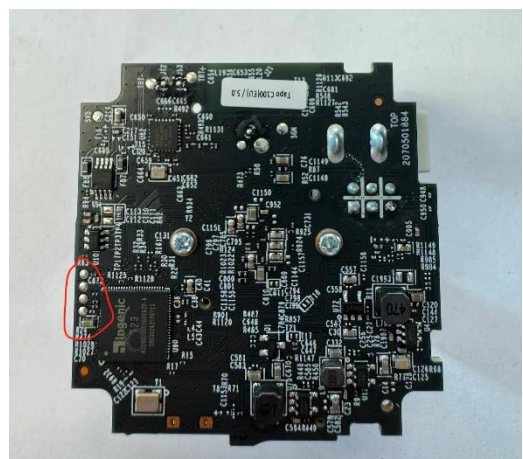


### 2.2.2 Repérage de l'interface UART :

Une inspection visuelle de la carte a été réalisée pour localiser l'interface série. Contrairement à certaines cartes de développement qui disposent de connecteurs standards (headers) clairement étiquetés (VCC, GND, RX, TX), la carte de la Tapo C100 ne présente pas de tels indicateurs évidents.

Nous avons identifié un groupe de quatre points de test (pads) situés à proximité du processeur principal. Pour identifier le rôle de chaque point :

- **GND (Masse)** : Plutôt que de chercher un pad spécifique, nous avons identifié que la cage métallique du lecteur de carte SD était reliée à la masse. Nous avons donc utilisé ce point comme référence fiable pour le Ground (GND).
- **RX et TX** : Les lignes de transmission (TX) et de réception (RX) ont été repérées parmi les pads identifiés sur le PCB. Ces points permettront d'intercepter les journaux de démarrage (logs de boot) et d'envoyer des commandes au système.







### 2.3.2 Configuration logicielle et Démarrage :

Une fois le montage physique réalisé, nous avons configuré l'outil de communication série sur l'ordinateur (picocom). Nous avons effectué des recherches pour déterminer la vitesse de transmission (baud rate) pour ce type de périphérique et on a pu déterminer que la vitesse est de 115200 bauds. Une configuration incorrecte de ce paramètre entraînerait l'affichage de caractères illisibles (garbage data).

La commande suivante a été exécutée pour ouvrir le port série en écoute :

```
picocom -b 115200 /dev/ttyUSB0
```

Avec le terminal actif, nous avons branché l'alimentation de la caméra. Immédiatement après la mise sous tension, les journaux de démarrage (*boot logs*) ont commencé à défiler dans la console, confirmant la réussite de la connexion série et le bon positionnement des sondes.

## 2.4 Obtention du Shell root

Une fois la connexion série établie, nous avons laissé le processus de démarrage de la caméra se dérouler complètement.

### 2.4.1 Identification et Authentification :

À la fin de la séquence de boot, le défilement des journaux système s'est interrompu pour laisser place à une invite de connexion (login:).

Pour franchir cette étape, nous avons effectué des recherches en ligne afin de trouver des identifiants constructeurs ou des vulnérabilités connues sur ce modèle. Nous avons identifié un article technique spécifique (<https://notes.landon.pw/notes/embedded/TP-LINK-Tapo-C100>) documentant les accès pour la Tapo C100.

Les identifiants récupérés sont les suivants :

- **Utilisateur :** root
- **Mot de passe :** slpingenic

### 2.4.2 Validation de l'accès :

Après la saisie de ces identifiants, l'authentification a réussi. L'invite de commande a changé pour afficher root@(c100v5):#, confirmant que nous disposons désormais d'un accès root sur le système d'exploitation embarqué (BusyBox Linux).

```
(c100v5) login: root
Password:
Jun 17 11:32:02 login[777]: root login on 'ttyS1'

BusyBox v1.19.4 (2024-06-17 19:22:15 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

  SSSSSSSSSSSSSSSSS  LLLLLLLLLLLLLL  PPPPPPPPPPPPPPPPPPP
  SSSSSSSSSSSSSSSSS  LLLLLLLLLLLLLL  PPPPPPPPPPPPPPPPPPP
  SSSSSS    SSSSSSSS  LLLLLL    PPPPPP    PPPPPPPP
  SSSSSS    SSSSSS    LLLLLL    PPPPPP    PPPPPP
  SSSSSSSS    SSSS    LLLLLL    PPPPPP    PPPPPP
  SSSSSSSSSSSSSSS  LLLLLL    PPPPPP    PPPPPPPP
  SSSSSSSSSSSSSSS  LLLLLL    PPPPPPPPPPPPPPPPPPP
  SSSS    SSSSSSSSSS  LLLLLL    LL    PPPPPP
  SSSSSS    SSSSSS    LLLLLL    LLLLLL  PPPPPP
  SSSSSSSS    SSSSSS    LLLLLL    LLLL  PPPPPP
  SSSSSSSS    SSSSSSSS  LLLLLL    LLLLLL  PPPPPP
  SSSSSSSSSSSSSSSSS  LLLLLLLLLLLLLLLLLLLLLL  PPPPPPPPPPPPPPP
  SSSS  SSSSSSSSSS    LLLLLLLLLLLLLLLLLLLLLL  PPPPPPPPPPPPPPP
-----
SMARTS, the power to be your best! (torchlight: svn unknown)
-----

root@(c100v5):~#
root@(c100v5):~#
root@(c100v5):~#
```

## 2.5 Exploration du File System

Disposant désormais des droits d'administration, nous avons pu explorer l'architecture du système pour localiser l'emplacement précis du firmware.

### 2.5.1 Localisation des partitions :

Nous avons utilisé la commande `cat /proc/mtd` pour afficher la table des partitions mémoire du système (Memory Technology Devices).

```
dev:   size  erasesize  name
mtd0: 0002d800 00001000 "factory_boot"
mtd1: 00002800 00001000 "factory_info"
mtd2: 00010000 00001000 "art"
mtd3: 00010000 00001000 "config"
mtd4: 00020000 00001000 "normal_boot"
mtd5: 0013fe00 00008000 "kernel"
mtd6: 00220000 00008000 "rootfs"
mtd7: 003a0000 00008000 "rootfs_data"
mtd8: 00080000 00008000 "user_record"
mtd9: 00010000 00008000 "verify"
mtd10: 00700000 00008000 "firmware"
mtd11: 00800000 00008000 "uitron"
mtd12: 00800000 00008000 "uitron_ext"
mtd13: 00800000 00008000 "ld"
mtd14: 00800000 00008000 "isp"
mtd15: 007d0000 00008000 "af"
```

L'exécution de cette commande a retourné la liste des zones mémoire mappées. Nous avons identifié la partition **mtd10**, explicitement étiquetée "firmware".

### 2.5.2 Vérification :

En explorant le dossier des périphériques (/dev), nous avons confirmé la présence du fichier de périphérique correspondant (/dev/mtd10).

```
CIW-I--I-- 1 root    root    90,  1 Jan 1 1970 mtd0ro
CIW-I--I-- 1 root    root    90,  2 Jan 1 1970 mtd1
CIW-I--I-- 1 root    root    90, 20 Jan 1 1970 mtd10
CIW-I--I-- 1 root    root    90, 21 Jan 1 1970 mtd10ro
CIW-I--I-- 1 root    root    90, 22 Jan 1 1970 mtd11
CIW-I--I-- 1 root    root    90, 23 Jan 1 1970 mtd11ro
CIW-I--I-- 1 root    root    90, 24 Jan 1 1970 mtd12
CIW-I--I-- 1 root    root    90, 25 Jan 1 1970 mtd12ro
CIW-I--I-- 1 root    root    90, 26 Jan 1 1970 mtd13
CIW-I--I-- 1 root    root    90, 27 Jan 1 1970 mtd13ro
CIW-I--I-- 1 root    root    90, 28 Jan 1 1970 mtd14
CIW-I--I-- 1 root    root    90, 29 Jan 1 1970 mtd14ro
CIW-I--I-- 1 root    root    90, 30 Jan 1 1970 mtd15
CIW-I--I-- 1 root    root    90, 31 Jan 1 1970 mtd15ro
CIW-I--I-- 1 root    root    90,  3 Jan 1 1970 mtd1ro
CIW-I--I-- 1 root    root    90,  4 Jan 1 1970 mtd2
CIW-I--I-- 1 root    root    90,  5 Jan 1 1970 mtd2ro
CIW-I--I-- 1 root    root    90,  6 Jan 1 1970 mtd3
CIW-I--I-- 1 root    root    90,  7 Jan 1 1970 mtd3ro
CIW-I--I-- 1 root    root    90,  8 Jan 1 1970 mtd4
```

Bien qu'il soit techniquement possible de copier ce fichier vers un support externe (comme une carte SD) pour l'exfiltrer logicielllement, l'objectif de ce laboratoire est d'effectuer une extraction matérielle. Cette localisation confirme simplement la cible pour la suite des opérations.



## 2.6 Recherche de certificats et clés cryptographiques

Disposant d'un accès *root* fonctionnel sur le système en cours d'exécution, nous avons profité de cette connexion privilégiée pour effectuer une première reconnaissance des données sensibles stockées sur l'appareil. L'objectif était d'identifier la présence de certificats et de clés privées potentiellement accessibles en clair dans le système de fichiers.

### 2.6.1 Inventaire automatisé

Pour localiser ces artefacts, nous avons exécuté une commande de recherche à la racine du système, filtrant les résultats sur les extensions courantes de fichiers cryptographiques (.crt, .pem, .key). La commande utilisée est la suivante :

```
find / -maxdepth 6 -type f \( -name '*.crt*' -o -name '*.pem*' -o -name '*.key*' \) 2>/dev/null
```

L'exécution de cette commande a retourné une liste exhaustive de certificats et clés dispersés dans l'arborescence. Parmi les résultats les plus critiques, nous avons noté :

- **Serveur Web (uhttpd)** : Les fichiers /etc/uhttpd.key et /etc/uhttpd.crt, utilisés pour sécuriser l'interface web de la caméra.
- **Protocole ONVIF** : Une paire de clés située dans /rom/mnt/www/cert/onvif/ (private\_key.pem et public\_key.pem), utilisée pour l'interopérabilité avec les enregistreurs vidéo tiers.
- **Services Cloud** : Des certificats liés à la communication avec le cloud TP-Link, trouvés dans /rom/mnt/www/cert/firmware et /rom/mnt/www/cert/cloud.crt.

### 2.6.2 Vérification du contenu

Afin de confirmer la nature critique de ces fichiers, nous avons affiché le contenu de la clé privée du serveur web via la commande `cat /etc/uhttpd.key`.

L'en-tête -----BEGIN RSA PRIVATE KEY----- visible dans le terminal confirme qu'il s'agit bien d'une clé privée RSA stockée sans chiffrement (au repos) sur la mémoire de la caméra.

Cette découverte via l'UART valide l'intérêt d'effectuer une extraction complète du firmware, car elle confirme que les secrets ne sont pas stockés dans un Secure Element inaccessible, mais bien dans le système de fichiers standard.

```
root@(c100v5):~# cat /etc/uhttpd.key
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDGDW+qIF3yx9bJ4y9ktQ++qJ4k1N+kDckj12+EkWwOzdVIZxQN
aKIEQR5pKfLLa1Nuq0v0jNutyGCoc5/ssn10C25wfU8w9v8Lg3LdXH20RPMxrwLO
wHxY9EJmWiuSUSdHf+aJvLTZ36sEm77+5sp1b1vtEFBybHFNyifOP4Y9QIDAQAB
AoGAbriH0w/tZbKus0Lc1fIBET1226Re0nqyfeUMpRIbspADsmxd23kpWLyS4bvf
zjT7+Zyt49NzZCSrpveg6lSebI2KUI5/+ffY0K8Met7nRhm0KyDZODR9wscA8CGM
lFHJGZP2xzJWwU2+wAFVSAGYTY7RVPTdYcx+/afUim8cQUECQDrlEcQeMcVNILT
ZgFfVjngIWUzSxbrsZyW0r9lUSSYcIECLaSHIBs1tQGPTXpeKZXK5gKrVYUGf/L
ookakWZNAKEA1zhn1HD0e6GyUaFFDc7VxvJwG6lRIlnaushgozrifu/eIn0vLLy
TM2GTGkKui/q1+5LDIsVjM8/x0np4LghSQJBAI60n+fmgHLiYG3W2PX7scuH70pH
JndYfFa098RURyrg3AWJyKTiWhlKsMtXT9Nctb4rUusE3aQHq+CbcRpbAFECQCS
xQehy8oeib3Gh1lF0ll1T84B7yzCU6RypH2qs1yEvmXZVU+XjZ5A+vJLfs/z6+y
w/HwQvWwNudePui4U7pAkeAw5N8tWPCD5wP1pnp2MpSVFp24s+Js+6W50QG8Ss2
piZ8/TQdnpS666B99EI+Y7cuIfKvNXuAXDmS9wE+6Wfwqw=
-----END RSA PRIVATE KEY-----
```

```
root@(c100v5):~# find / -maxdepth 6 -type f \( -name '*.crt*' -o -name '*.pem*' -o -name '*.key*' \) 2>/dev/null
/etc/data/dss_certificates/StarfieldClass2CertificationAuthority.pem
/etc/data/dss_certificates/dpss_public_key.pem
/etc/uhttpd.crt
/etc/uhttpd.key
/proc/sys/net/ipv4/tcp_fastopen_key
/rom/mnt/etc/data/dss_certificates/StarfieldClass2CertificationAuthority.pem
/rom/mnt/etc/data/dss_certificates/dpss_public_key.pem
/rom/mnt/etc/uhttpd.crt
/rom/mnt/etc/uhttpd.key
/rom/mnt/www/cert/cloud.crt
/rom/mnt/www/cert/firmware/firmware-pub-key.pem
/rom/mnt/www/cert/firmware/firmware-pub-key.rsa2048key
/rom/mnt/www/cert/onvif/private_key.pem
/rom/mnt/www/cert/onvif/public_key.pem
/rom/mnt/www/cert/tmpd/priv-key.pem
/rom/mnt/www/cert/tmpd/server-cert.pem
/rom/sp_rom/etc/data/dss_certificates/StarfieldClass2CertificationAuthority.pem
/rom/sp_rom/etc/data/dss_certificates/dpss_public_key.pem
/rom/sp_rom/etc/uhttpd.crt
/rom/sp_rom/etc/uhttpd.key
/rom/www/cert/cloud.crt
/rom/www/cert/firmware/firmware-pub-key.pem
/rom/www/cert/firmware/firmware-pub-key.rsa2048key
/rom/www/cert/onvif/private_key.pem
/rom/www/cert/onvif/public_key.pem
/rom/www/cert/tmpd/priv-key.pem
/rom/www/cert/tmpd/server-cert.pem
/sys/devices/platform/gpio-keys/keys
/sys/devices/platform/gpio-keys/disabled_keys
/tmp/base-files/etc/encrypt_key
/www/cert/cloud.crt
/www/cert/firmware/firmware-pub-key.pem
/www/cert/firmware/firmware-pub-key.rsa2048key
/www/cert/onvif/private_key.pem
/www/cert/onvif/public_key.pem
/www/cert/tmpd/priv-key.pem
/www/cert/tmpd/server-cert.pem
root@(c100v5):~#
```

### 3. Extraction du Firmware via la mémoire flash

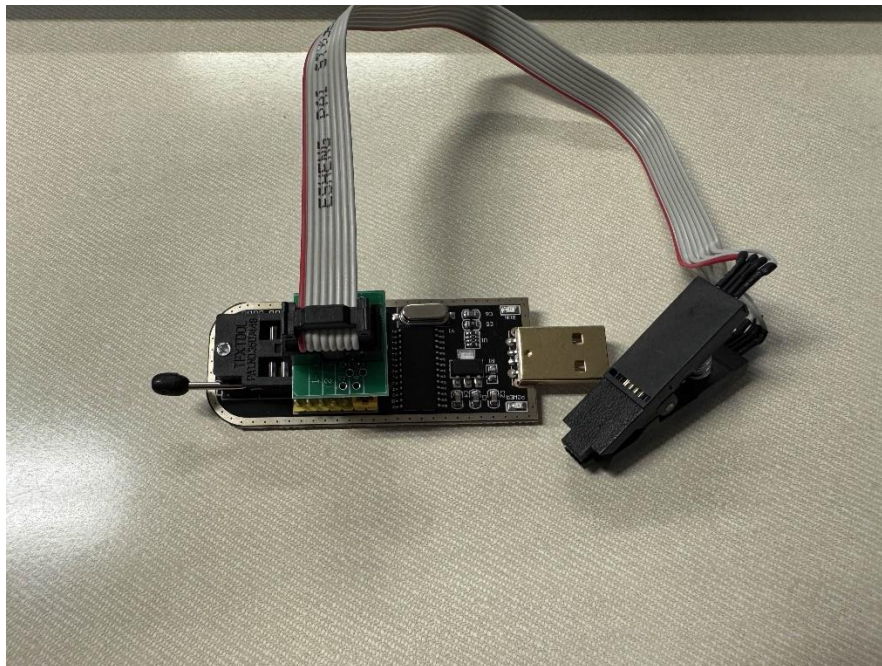
Cette seconde méthode d'extraction vise à contourner les protections logicielles potentielles en lisant directement le contenu de la puce mémoire (SPI Flash) située sur la carte mère.

#### 3.1 Mise en place du matériel

Pour réaliser cette opération sans dessolder le composant, nous avons utilisé un kit de programmation spécifique comprenant :

- **Un programmeur CH341A (Mini Programmer) :** Ce dongle USB permet de s'interfacer avec des puces mémoires de type EEPROM et Flash SPI (séries 24 et 25) pour effectuer des opérations de lecture et d'écriture brutes.
- **Une pince de test SOIC8/SOP8 (Test Clip) :** Cet outil est essentiel pour se connecter physiquement aux 8 broches de la puce mémoire directement sur le circuit imprimé (*In-Circuit Serial Programming*).
- **Un adaptateur SOP8/DIP8 :** Ce petit circuit imprimé sert d'interface entre la nappe de la pince et le support ZIF du programmeur CH341A, assurant la correspondance correcte du câblage.

Ce matériel permet d'établir une liaison directe entre l'ordinateur et l'espace de stockage de la caméra, indépendamment du processeur principal.

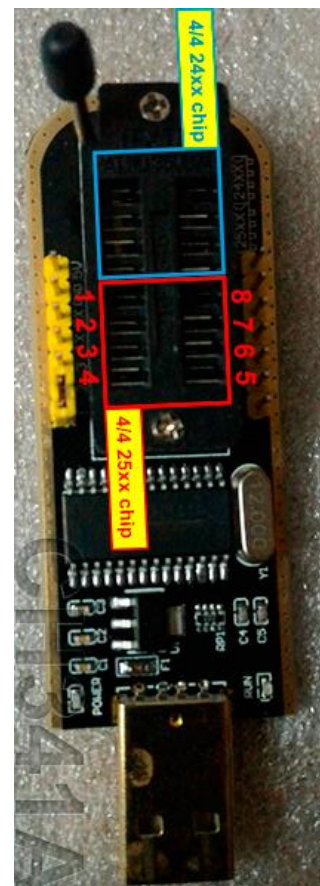


### 3.1.1 Connexion physique du programmeur

La liaison entre la pince de test (connectée à la puce de la caméra) et l'ordinateur est assurée par le programmeur CH341A via l'adaptateur PCB vert (SOP8 vers DIP8).

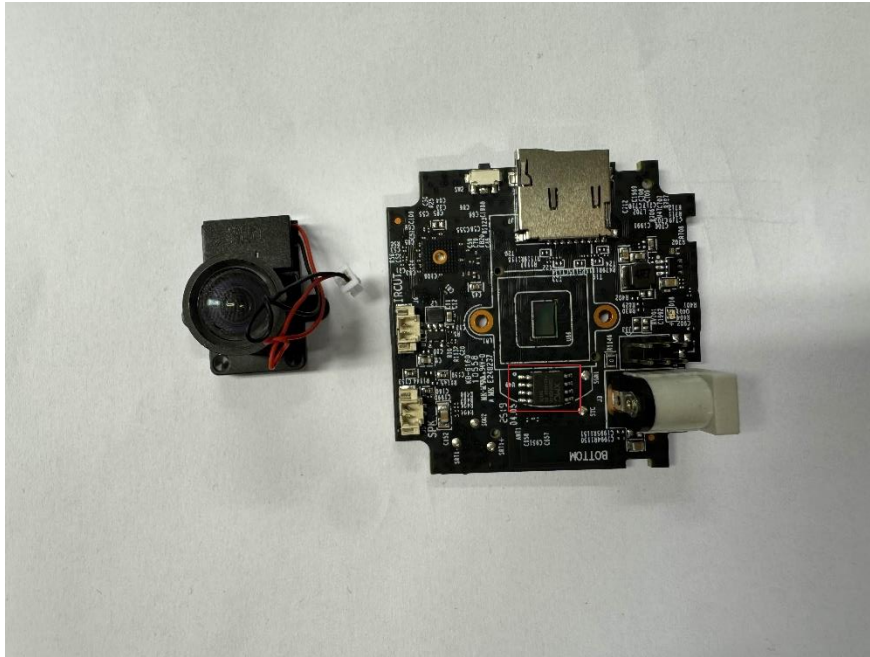
Le branchement sur le support ZIF du programmeur a été réalisé selon la logique suivante, justifiée par la sérigraphie du matériel et les standards techniques du CH341A :

1. **Sélection de l'interface (Zone 25 SPI)** : La puce cible étant une mémoire Flash SPI (Série 25), elle nécessite un protocole de communication différent des EEPROMs (Série 24). Nous avons donc utilisé exclusivement la zone identifiée "**25 SPI**" au dos du circuit.
2. **Positionnement physique** : Sur ce modèle de programmeur, cette zone correspond aux **8 emplacements supérieurs** (côté levier de verrouillage), par opposition à la zone inférieure réservée au protocole I2C.
3. **Orientation (Pin 1)** : L'adaptateur a été inséré en alignant son repère "Pin 1" (correspondant au fil rouge de la nappe) avec l'encoche indiquée sur le schéma du programmeur (en haut à gauche). Cet alignement est critique pour garantir la correspondance des broches de données (MOSI, MISO) et d'horloge (CLK).



### 3.2 Identification et spécifications du composant mémoire

Avant de procéder à la lecture, il est impératif d'identifier précisément le modèle de la puce pour configurer correctement le protocole de communication SPI. Nous avons d'abord démonté la caméra pour accéder à la puce mémoire. Une inspection minutieuse à la loupe nous a ensuite permis de relever la référence **XMC25QH64DH1Q**, information nécessaire pour configurer le protocole SPI avant la lecture



L'analyse de cette référence nous donne les spécifications techniques suivantes :

- **Fabricant** : XMC (Wuhan Xinxin Semiconductor Manufacturing Corp).
- **Série** : 25QH (Mémoire Flash SPI, 3.3V).
- **Capacité** : 64 Mbits, ce qui correspond à **8 Mo** (Mégaoctets).
- **Interface** : SPI (Serial Peripheral Interface).

### 3.3 Configuration du logiciel d'extraction (IMSProg)

Pour l'interface logicielle, nous avons utilisé l'outil **IMSProg** (compatible Linux/Windows), reconnu pour sa fiabilité avec les programmeurs basés sur la puce CH341A.

La détection automatique (fonction *Detect*) couplée à une configuration manuelle a permis de définir les paramètres de lecture optimaux pour éviter toute corruption de données.

**Paramètres retenus pour l'extraction :**

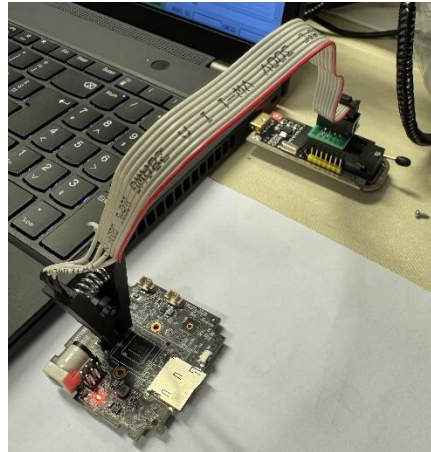
- **Chip Type** : SPI\_FLASH (Protocole série standard).
- **Manufacture** : XMC (Fabricant identifié).
- **Name** : XM25QH64C (Variante compatible détectée pour la version 64Mbit).
- **Size** : 8 M (La taille totale du dump sera de 8 388 608 octets).
- **Page size** : 256 (Unité standard de programmation pour les mémoires Flash NOR).
- **Block size** : 64 K (Taille du bloc d'effacement standard).
- **4byte Addr.** : No.



## 3.4 Procédure de lecture et analyse du dump

### 3.4.1 Lecture physique du composant

Une fois la pince de test correctement positionnée sur la puce XMC et le programmeur configuré, l'ensemble a été connecté au port USB de la station d'analyse. Comme illustré ci-dessous, la LED d'état du programmeur confirme l'alimentation du circuit.



L'opération de lecture a été lancée via le logiciel **IMSProg** avec les paramètres définis à l'étape précédente. Le processus a permis de copier l'intégralité du contenu de la mémoire Flash vers un fichier binaire brut nommé `firmware_camera.bin` sur notre machine locale.

### 3.4.2 Analyse de la structure avec Unblob

Disposant désormais de l'image brute du système, l'étape suivante a consisté à en extraire les composants logiques (systèmes de fichiers, noyaux, fichiers de configuration). Pour cela, nous avons utilisé l'outil `unblob`, un extracteur automatisé spécialisé dans l'analyse de firmwares IoT.

La commande suivante a été exécutée sur le fichier dump :

```
unblob ~/firmware_camera.bin
```

L'outil a identifié et extrait automatiquement plusieurs segments ("chunks") distincts, comme le montre le rapport d'exécution ci-dessous :

```
(unblob-env) tauma@ThinkPad ~-> unblob ~/helmob3/iot/Labo2_camera_Tapo_C100/Lecture_flash/firmware_camera.bin
unblob (25.11.25)
Output path: /home/tauma/helmob3/iot/Labo2_camera_Tapo_C100/Lecture_flash/firmware_camera_bin_extract
Extracted files: 71
Extracted directories: 54
Extracted links: 0
Extraction directory size: 18.77 MB
Summary
Chunks distribution


| Chunk type     | Size      | Ratio  |
|----------------|-----------|--------|
| ELF32          | 9.31 MB   | 51.79% |
| SQUASHFS_V4_LE | 3.08 MB   | 17.12% |
| UNKNOWN        | 2.23 MB   | 12.39% |
| XZ             | 1.67 MB   | 9.30%  |
| JFFS2_NEW      | 512.00 KB | 2.78%  |
| PADDING        | 508.49 KB | 2.76%  |
| TAR            | 460.00 KB | 2.50%  |
| LZMA           | 145.37 KB | 0.79%  |
| GZIP           | 104.62 KB | 0.57%  |


Chunk identification ratio: 87.61%
Encountered errors


| Severity       | Name                              |
|----------------|-----------------------------------|
| Severity.ERROR | ExtractorDependencyNotFoundReport |

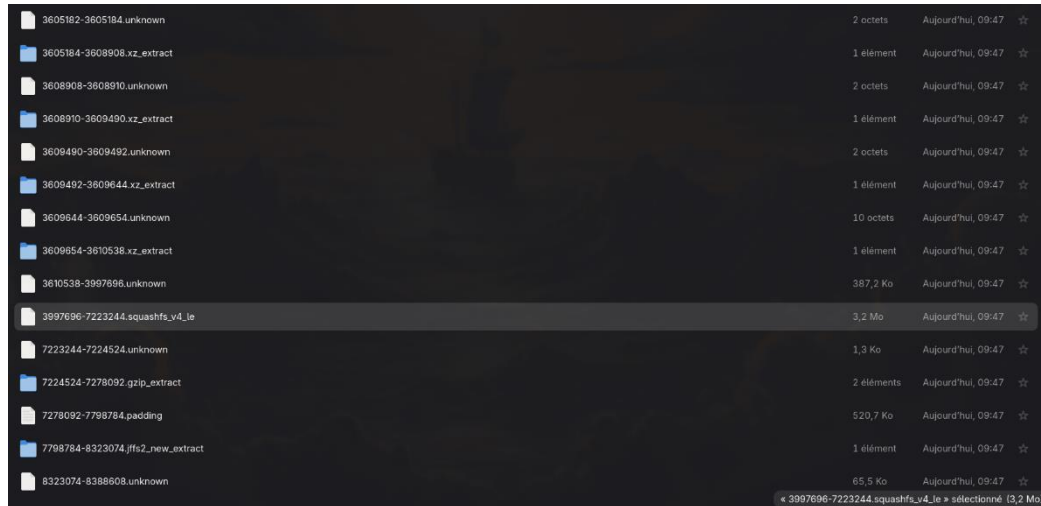

```



### 3.4.3 Identification du système de fichiers (SquashFS)

L'analyse des résultats d'extraction révèle un élément critique pour la suite de nos recherches : la présence d'une partition SquashFS (version 4, Little Endian) d'une taille d'environ 3 Mo.

En explorant le dossier de sortie généré par unblob, nous avons pu isoler ce fichier spécifique :



3605182-3605184.unknown	2 octets	Aujourd'hui, 09:47	☆
3605184-3608908.xz_extract	1 élément	Aujourd'hui, 09:47	☆
3608908-3608910.unknown	2 octets	Aujourd'hui, 09:47	☆
3608910-3609490.xz_extract	1 élément	Aujourd'hui, 09:47	☆
3609490-3609492.unknown	2 octets	Aujourd'hui, 09:47	☆
3609492-3609644.xz_extract	1 élément	Aujourd'hui, 09:47	☆
3609644-3609654.unknown	10 octets	Aujourd'hui, 09:47	☆
3609654-3610538.xz_extract	1 élément	Aujourd'hui, 09:47	☆
3610538-3997696.unknown	387,2 Ko	Aujourd'hui, 09:47	☆
3997696-7223244.squashfs_v4_le	3,2 Mo	Aujourd'hui, 09:47	☆
7223244-7224524.unknown	1,3 Ko	Aujourd'hui, 09:47	☆
7224524-7278092.gz_extract	2 éléments	Aujourd'hui, 09:47	☆
7278092-7798784.padding	520,7 Ko	Aujourd'hui, 09:47	☆
7798784-8323074.jffs2_new_extract	1 élément	Aujourd'hui, 09:47	☆
8323074-8388608.unknown	65,5 Ko	Aujourd'hui, 09:47	☆

Ce fichier ...squashfs\_v4\_le correspond au Root Filesystem (RootFS) de la caméra. C'est dans cette archive que se trouvent l'arborescence des dossiers Linux, les scripts de démarrage et les binaires propriétaires de TP-Link que nous cherchons à analyser.

## 3.5 Décompression et analyse du système de fichiers (RootFS)

### 3.5.1 Extraction du contenu SquashFS

L'analyse précédente avec unblob ayant identifié une partition de type SquashFS v4 (Little Endian), nous avons procédé à sa décompression manuelle pour accéder à l'arborescence des fichiers.

Nous avons utilisé l'outil unsquashfs avec l'option -d pour extraire le contenu dans un répertoire nommé rootfs. La commande exécutée est la suivante :

```
unsquashfs -d rootfs 3997696-7223244.squashfs_v4_le
```

Comme l'illustre la capture ci-dessous, l'outil a réussi à extraire 69 fichiers et 20 répertoires sans erreur.

```
tauma@ThinkPad ~/h/i/L/L/firmware_camera.bin_extract> unsquashfs -d rootfs 3997696-7223244.squashfs_v4_le
Parallel unsquashfs: Using 16 processors
75 inodes (196 blocks) to write

[=====]

created 69 files
created 20 directories
created 6 symlinks
created 0 devices
created 0 fifos
created 0 sockets
created 0 hardlinks
```

### 3.5.2 Exploration de l'arborescence

Une fois l'extraction terminée, nous avons exploré le dossier rootfs. Nous y retrouvons une structure de répertoires standard pour un système Linux embarqué (bin, etc, lib, usr), confirmant que nous avons bien accès à la racine du système d'exploitation de la caméra.

```
tauma@ThinkPad ~/h/i/L/L/firmware_camera.bin_extract> ll rootfs/
total 20K
drwxr-xr-x 2 tauma tauma 4,0K 17 jun 2024 bin/
drwxr-xr-x 3 tauma tauma 4,0K 17 jun 2024 config/
drwxr-xr-x 7 tauma tauma 4,0K 17 jun 2024 etc/
drwxr-xr-x 4 tauma tauma 4,0K 17 jun 2024 lib/
drwxr-xr-x 5 tauma tauma 4,0K 17 jun 2024 usr/
```

### 3.5.3 Recherche de certificats et clés cryptographiques

L'objectif principal de cette extraction était d'identifier la présence de certificats ou de clés privées stockés en clair, ce qui constituerait une vulnérabilité potentielle.

Nous avons concentré notre recherche sur le répertoire /etc, emplacement habituel des fichiers de configuration et des secrets sur ce type d'appareil. L'exploration de ce dossier (cd rootfs/etc) a immédiatement révélé la présence de deux fichiers critiques pour le serveur web embarqué de la caméra (uhttpd) :

- uhttpd.crt : Le certificat public
- uhttpd.key : La clé privée associée

```
tauma@ThinkPad ~/h/i/L/L/firmware_camera.bin_extract> cd rootfs/etc
tauma@ThinkPad ~/h/i/L/L/f/r/etc> ll
total 56K
drwxr-xr-x 3 tauma tauma 4,0K 6 mar 2024 data/
drwxr-xr-x 2 tauma tauma 4,0K 17 jun 2024 default_isp/
-rw-r--r-- 1 tauma tauma 0 6 mar 2024 dnsd.conf
-rwxr-xr-x 1 tauma tauma 19K 10 mai 2024 dsd_convert.json*
-r--r--r-- 1 tauma tauma 32 17 jun 2024 hw_id.bin
drwxr-xr-x 2 tauma tauma 4,0K 24 mai 2024 isp_extra/
drwxr-xr-x 2 tauma tauma 4,0K 17 jun 2024 OSD/
drwxr-xr-x 2 tauma tauma 4,0K 17 jun 2024 plugins/
lrwxrwxrwx 1 tauma tauma 19 17 jun 2024 sensor -> /tmp/base-files/cfg
-rwxr-xr-x 1 tauma tauma 741 6 mar 2024 uhttpd.crt*
-rwxr-xr-x 1 tauma tauma 891 6 mar 2024 uhttpd.key*
-rwxr-xr-x 1 tauma tauma 871 6 mar 2024 webrtc_profile.ini*
```

Afin de réaliser un inventaire exhaustif de tous les certificats présents dans le système de fichiers, nous avons exécuté une commande de recherche automatisée ciblant les extensions courantes (.crt, .pem, .key) :

```
tauma@ThinkPad ~/h/i/L/L/> find . -maxdepth 10 -type f ( -name 'crt' -o -name 'pem' -o -name 'key' ) 2>/dev/null
```

Cette recherche a permis de localiser deux autres fichiers dans le sous-dossier data/dss\_certificates :

1. StarfieldClass2CertificationAuthority.pem (Une autorité de certification racine).
2. dpss\_public\_key.pem (Une clé publique probablement utilisée pour la vérification de signature ou le chiffrement de flux vidéo).

```
tauma@ThinkPad ~/h/i/L/L/f/r/etc> find . -maxdepth 10 -type f \( -name '*crt*' -o -name '*pem*' -o -name '*key*' \) 2>/dev/null
./uhttpd.key
./data/dss_certificates/StarfieldClass2CertificationAuthority.pem
./data/dss_certificates/dpss_public_key.pem
./uhttpd.crt
```

## 4. Conclusion

Ce laboratoire consacré à l'analyse de la caméra TP-Link Tapo C100 nous a permis d'explorer concrètement les surfaces d'attaque matérielles d'un dispositif IoT grand public. L'objectif initial, qui était d'extraire et d'analyser le firmware, a été pleinement atteint grâce à la mise en œuvre complémentaire de deux vecteurs d'attaque : l'exploitation du port série (UART) et l'extraction physique de la mémoire Flash (SPI Dump).

La première phase via l'UART a mis en lumière un choix de conception risqué : le maintien d'une interface de débogage active et verbeuse sur un produit final. Bien qu'un effort d'obfuscation soit notable (le port étant situé au dos de la carte et dépourvu de tout marquage standard) cette "sécurité par l'obscurité" n'a pas suffi à empêcher l'identification des broches. Une fois la connexion physique établie, la compromission du système a été triviale en raison de l'utilisation d'identifiants constructeur par défaut (root / slpingenic) documentés en ligne.

La seconde phase, via l'utilisation du programmeur CH341A et de la pince de test, a démontré qu'il est possible de réaliser une copie intégrale du système sans dessouder de composant sans destruction du matériel et sans posséder le mot de passe. L'absence de protection contre la lecture (Readout Protection) sur la puce mémoire XMC 25QH64 facilite grandement la récupération du code binaire.

Enfin, l'analyse de l'image extraite, réalisée avec les outils Unblob et SquashFS-tools, a révélé que le système de fichiers n'est pas chiffré. Cette transparence a permis d'exposer des données sensibles critiques, notamment les clés privées RSA du serveur web (uhttpd.key) et les certificats de communication, stockés en clair dans le répertoire /etc.

En conclusion, ce travail démontre que la sécurité de ce dispositif repose essentiellement sur l'intégrité physique de son boîtier. Une fois qu'un attaquant dispose d'un accès physique à la caméra, la compromission totale du système (accès root, vol de certificats, modification de firmware) est réalisable avec un matériel peu coûteux et des logiciels open source.

## 5. Bibliographie

Bios-Mods. (2016, 24 décembre). *[GUIDE] Flash BIOS with CH341A programmer*

<https://www.bios-mods.com/forum/Thread-GUIDE-Flash-BIOS-with-CH341A-programmer?page=3>

Electronics&Computers. (2025, 29 juin). *How to Fix a Corrupted BIOS with a \$5 Programmer / CH341A BIOS Flash Tutorial* [Vidéo]. YouTube.

<https://www.youtube.com/watch?v=mSLKILcOAKM>

Landon. (n.d.). *TP-LINK Tapo C100*. Landon's Notes.

<https://notes.landon.pw/notes/embedded/TP-LINK-Tapo-C100>

Matt Brown. (2024, 17 décembre). *In-Circuit Firmware Extraction with the CH341A - The Poor Man's Flash Programmer* [Vidéo]. YouTube.

<https://www.youtube.com/watch?v=6Z5aWu9tqmA>

Matt Brown. (2025, 3 décembre). *Is This Malware? - Rooting the SuperBox S6 Pro* [Vidéo]. YouTube.

<https://www.youtube.com/watch?v=R82pt4rLhBQ>

ONEKEY. (n.d.). *Installation*. Unblob.

<https://unblob.org/installation/>