

Master 2 – Cybersécurité - Année 2024-25

CYBERSÉCURITÉ DES SYSTÈMES EMBARQUÉS

**Étude des composants, protocoles,
outils d'analyse et techniques d'attaque
et de défense en cybersécurité physique
et radio**

Abdel-Malik Fofana

Maître : Luc Bouganim (INRIA)

Tuteur : Patrice Martin (Université Paris Cité)

Table des matières

1 Introduction.....	4
2 Cybersécurité physique.....	4
2.1 Explication des termes et composants.....	4
2.1.1 Microcontrôleurs (CPU, MCU, SoC, STM32, ESP32).....	5
2.1.2 Stockage embarqué (EEPROM, eMMC, nvme , hdd, sd ,).....	5
2.1.3 Modules de sécurité (TPM).....	6
2.2 Attaques physiques connues.....	7
2.2.1 Dump de mémoire via interfaces (I2C, UART, SPI).....	7
2.2.2 Lecture d'EEPROM (ex : ECU moto Red Key, BIOS, consoles de jeux)....	9
2.2.3 Attaques par accès physique (glitch, cold boot, etc.).....	10
Pourquoi est-il utile de parler de ce type d'attaque dans les pc?.....	12
2.2.4 Attaques via périphériques USB.....	12
2.3 Défenses physiques.....	14
2.3.1 Protection des interfaces.....	14
2.3.2 Sécurisation des données (chiffrement, blindage).....	15
2.3.3 Limitation des accès (boîtier, firmware sécurisé).....	16
2.3.4 Défense physique dans les environnements industriels (SCADA, ICS)...	16
3 Cybersécurité radio et sans-fil.....	18
3.1 Protocoles concernés.....	18
3.1.1 Wi-Fi (WEP, WPA2, Radius, WPA3).....	18
3.1.1.1 Présentation protocole WEP.....	18
3.1.1.2 Présentation du protocole WPA2.....	21
3.1.1.3 Présentation protocole WPA3.....	23
3.1.2 Bluetooth classique et BLE.....	25
3.1.2.1 Définition et différences.....	25
3.1.3 NFC et RFID.....	25
RFID.....	25
NFC.....	26
Cartes MIFARE.....	26
3.1.4 Protocoles RF propriétaires : fixed code vs rolling code.....	27
3.2 Attaques radio courantes.....	29
3.2.1 Attack sur Wi-Fi WEP et WPA2 (Aircrack , Deauth Flipper, ESP32).....	29
3.2.2 Bluetooth Attack : attaque DoS BLE sans appairage (iPhone) et mouse jacking.....	31
Exploitation via le Flipper Zero.....	31
Correctif apporté par iOS 17.2.....	32
3.2.3 Clonage de badge NFC et RFID (fixed code).....	34
3.2.4 Copie de télécommande voiture (rolling code)	35

3.2.5 Brouillage radio divers.....	35
3.3 Défenses contre les attaques radio.....	38
3.3.1 Protocoles wifi.....	38
3.3.2 Détection d'anomalies radio.....	39
3.3.3 Blindage physique (portefeuille, couche métallique, etc.).....	40
4 Études de cas.....	42
4.1 BluAttack : analyse de l'attaque et effets sur iPhone.....	42
4.2 Clonage NFC : méthode et validation.....	43
Exemple de procédure :	43
4.3 Rolling Code : vulnérabilité des clés de voiture.....	45
4.4 Deauth Attack sur caméras Wi-Fi.....	47
Description du fonctionnement et de l'attaque.....	47
Exploitation des données capturées.....	49
4.5 Extraction de Red Key moto via EEPROM.....	51
5 Conclusion et remerciement.....	53
5.1 Conclusion.....	53
5.2 Remerciements.....	54
6 Figures.....	55
A Reference / Source.....	56

1 Introduction

Les systèmes embarqués sont devenus omniprésents dans notre quotidien : qu'il s'agisse de voitures modernes, de dispositifs médicaux, de caméras de surveillance, ou encore d'objets connectés, ils assurent des fonctions critiques tout en restant souvent invisibles pour l'utilisateur. La miniaturisation, alliée à leur spécialisation fonctionnelle, a permis une adoption massive, mais elle s'est parfois faite au détriment de la sécurité. En effet, nombre de ces systèmes ont été conçus avec des priorités de coût, de performance ou de simplicité, sans prendre en compte des menaces pourtant bien connues dans d'autres domaines de l'informatique.

Leur vulnérabilité ne relève pas seulement d'un manque de mises à jour ou d'un logiciel mal protégé : elle réside également dans leur exposition physique, dans la facilité avec laquelle certaines interfaces peuvent être détournées, ou dans les faiblesses de protocoles radio peu ou mal sécurisés. Aujourd'hui, un simple appareil à bas coût comme le Flipper Zero (voir 3.2.2) ou un lecteur d'eeprom permet d'illustrer des attaques autrefois réservées à des laboratoires spécialisés.

Ce mémoire propose donc une exploration technique et méthodologique de la cybersécurité appliquée aux systèmes embarqués, en abordant plusieurs vecteurs d'attaque réels, reproductibles, mais aussi les dispositifs de défense que l'on peut mettre en place. Des tests concrets ont été réalisés autour de composants courants comme les EEPROM, les microcontrôleurs ESP32, les protocoles sans fil (Wi-Fi, Bluetooth, NFC), mais aussi des dispositifs de sécurité plus complexes comme les TPM.

Au-delà du simple inventaire d'outils ou de techniques, ce travail vise à démontrer que la sécurité embarquée est un **équilibre subtil entre accès physique, couches logicielles, environnement réseau et comportement utilisateur**. L'objectif est donc d'offrir un cadre clair de compréhension, accompagné d'expérimentations concrètes, pour mieux appréhender les défis liés à la protection de ces systèmes.

2 Cybersécurité physique

2.1 Explication des termes et composants

Dans cette section, nous présentons les principaux composants matériels qui interviennent dans les systèmes embarqués, et qui seront abordés dans ce mémoire. Il s'agit de poser les bases techniques nécessaires à la compréhension des attaques et expérimentations détaillées dans les chapitres suivants.

2.1.1 Microcontrôleurs (CPU, MCU, SoC, STM32, ESP32)

Les microcontrôleurs, ou **MCU (Microcontroller Units)**, sont des composants essentiels des systèmes embarqués. Ils regroupent généralement un **processeur**, de la mémoire, et des interfaces d'entrée/sortie, le tout sur une seule puce. Leur rôle est de piloter un dispositif autonome (capteur, actionneur, appareil connecté, etc.) de manière rapide, efficace, et souvent avec des contraintes de consommation énergétique réduites.

On distingue plusieurs sous-catégories ou composants associés :

- **CPU (Central Processing Unit)** : cœur de traitement chargé d'exécuter les instructions.
- **MCU (Microcontroller Unit)** : Système embarqué compact intégrant un CPU, de la mémoire et des périphériques d'entrée/sortie sur une seule puce.
- **SoC (System on Chip)** : intègre, en plus du CPU, d'autres blocs fonctionnels comme le Wi-Fi, Bluetooth, GPU, etc., sur une seule puce. C'est le cas par exemple du **ESP32**, très utilisé dans les projets embarqués pour ses capacités réseau intégrées.
- **STM32** : une famille populaire de microcontrôleurs 32 bits développés par STMicroelectronics. Très utilisés dans l'industrie, ils sont réputés pour leur modularité et leur faible consommation.
- **ESP32** : développé par Espressif, ce SoC est très présent dans les environnements d'expérimentation pour sa compatibilité avec Wi-Fi, Bluetooth, et son support dans les environnements de développement comme Arduino ou PlatformIO.

Ces composants sont la base matérielle sur laquelle s'appuient les systèmes embarqués étudiés dans ce mémoire, notamment dans les attaques radio, la lecture d'EEPROM, ou les tests de sécurité physique.

2.1.2 Stockage embarqué (EEPROM, eMMC, nvme , hdd, sd ,)

Le **stockage embarqué** désigne les technologies de mémoire intégrées dans un système embarqué, destinées à conserver des données de façon temporaire ou permanente, même en l'absence d'alimentation.

EEPROM (Electrically Erasable Programmable Read-Only Memory) : mémoire non volatile permettant de stocker des données critiques, comme des identifiants, des configurations ou des clés de sécurité. Elle peut être lue ou modifiée à l'aide de

protocoles comme I2C. Des attaques peuvent viser directement ces puces pour extraire des informations sensibles.

SD / microSD : très utilisées dans les systèmes embarqués pour le stockage de fichiers, logs ou médias. Leur accessibilité physique les rend vulnérables à des extractions de données ou à des manipulations, notamment si les données ne sont pas chiffrées. Les cartes SD peuvent également être clonées ou modifiées pour injecter du code malveillant.

eMMC (embedded MultiMediaCard) : mémoire Flash plus rapide, souvent utilisée comme espace de stockage principal dans des systèmes embarqués plus complexes (ex. : smartphones, modules Android).

HDD (Hard Disk Drive) : bien qu'ils soient rares dans les systèmes embarqués modernes en raison de leur taille, consommation et fragilité, certains systèmes industriels ou équipements spécifiques peuvent encore embarquer des disques durs classiques. Ces derniers restent vulnérables à des attaques classiques d'extraction ou d'analyse forensique.

Ces mémoires peuvent être ciblées lors d'attaques physiques (dump, extraction, remplacement) afin d'obtenir des secrets ou d'altérer le comportement d'un système.

2.1.3 Modules de sécurité (TPM)

Le **TPM** (*Trusted Platform Module*) est un coprocesseur cryptographique dédié à la sécurité, conçu pour fournir des fonctions essentielles telles que le **stockage sécurisé des clés**, la **génération aléatoire**, le **chiffrement**, la **signature**, et l'**attestation de l'intégrité du système**.

Présent dans la plupart des ordinateurs modernes, le TPM constitue une **racine matérielle de confiance**. Son intégration dans les systèmes embarqués vise à renforcer la sécurité contre des attaques logicielles et physiques, en assurant que certaines opérations sensibles (comme le démarrage sécurisé ou l'identification de l'appareil) soient réalisées dans un environnement matériellement isolé.

Selon Arthur & Challener dans le livre *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*, le TPM a été conçu comme une brique fondamentale de la cybersécurité moderne, avec pour objectif de permettre des fonctions avancées de protection comme l'**authentification forte**, la **protection contre les attaques par dictionnaire**, ou encore la **création de clés non exportables**. Ils expliquent que le TPM « permet à un appareil de créer des clés cryptographiques et de les chiffrer de façon à ce qu'elles ne puissent être déchiffrées que par ce même TPM » (Arthur & Challener, 2015, Introduction, p. xxviii).

Avec la version 2.0, le TPM est devenu plus flexible et puissant. Il prend désormais en charge plusieurs algorithmes (SHA-1, SHA-256, RSA, ECC...), offre une **gestion hiérarchique des clés**, et autorise des **politiques d'autorisation complexes** pour un contrôle d'accès fin.

Le TPM peut être utilisé dans un contexte embarqué :

- **en version matérielle**, directement soudée sur la carte mère d'un appareil ;
- ou **via un simulateur logiciel**, souvent utilisé pour le développement, le test ou l'enseignement.

Exemples d'utilisation dans le monde PC / Windows :

- **BitLocker** (Windows) utilise le TPM pour stocker de manière sécurisée les clés de chiffrement du disque. Cela permet de protéger les données même si le disque est volé ou déplacé vers un autre appareil.
- **Windows Hello** exploite le TPM pour sécuriser l'authentification biométrique (empreinte digitale, reconnaissance faciale).
- **Secure Boot** s'appuie sur le TPM pour garantir que seul un système d'exploitation signé et autorisé est chargé au démarrage.
- Dans le monde **Linux**, des outils comme *tpm2-tools* permettent de gérer manuellement les clés stockées dans le TPM, ou de signer du code en ligne de commande.

2.2 Attaques physiques connues

2.2.1 Dump de mémoire via interfaces (I2C, UART, SPI)

De nombreux composants embarqués communiquent entre eux via des **bus série standards** comme **I2C**, **UART**, ou **SPI**. Ces interfaces ne sont généralement pas conçues pour être sécurisées, car elles sont supposées inaccessibles physiquement et logiquement via les priviléges requis mis en place par les OS par exemple. Dans un produit final ou réservé pour le debugging

En accédant à ces **bus** via des sondes ou adaptateurs (comme des **logic analyzers**, des convertisseurs **USB vers UART**, ou des **EEPROM readers**), il est parfois possible de :

- **écouter les échanges** entre microcontrôleur et mémoire ;

- extraire le contenu d'une mémoire Flash ou EEPROM ;
- voire injecter des commandes malveillantes dans certains cas et avoir un accès root (si le firmware n'est pas protégé).

Ces méthodes de dump sont couramment utilisées dans les tests de sécurité des appareils IoT ou des systèmes industriels embarqués, notamment lorsqu'il n'existe pas d'autre vecteur d'accès au contenu mémoire.

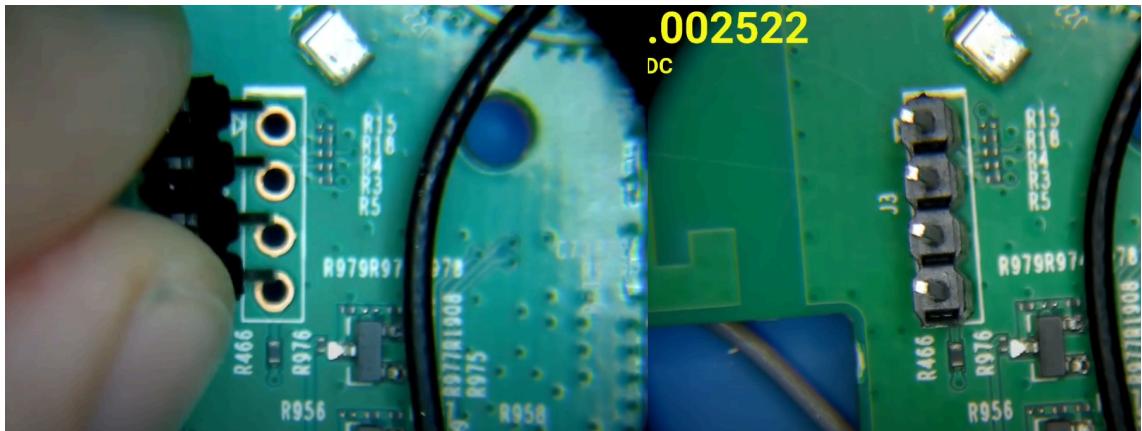


Figure 1 : Accès aux broches UART sur un routeur 4G LTE

Sur cette image, on peut voir le circuit d'un **routeur 4G Mercusys MB110**. Un port **UART** est accessible en y soudant des broches, ce qui permet d'obtenir un shell de debug. les **UART** étaient destinée au debug et ont été supprimé de la carte , sauf que l'on peut détourner l'usage pour faire de l'escalade de privilège

```
[nmat@nripper ~]$ picocom -b 115200 /dev/ttyUSB0
picocom v3.1

port is      : /dev/ttyUSB0
flowcontrol : none
baudrate is  : 115200
parity is    : none
databits are: 8
stopbits are: 1
escape is   : C-a
local echo is: no
noinit is   : no
noreset is  : no
hangup is   : no
nolock is   : no
send_cmd is : sz -vv
receive_cmd is: rz -vv -E
imap is     :
omap is     :
emap is     : crcrlf,delbs,
logfile is  : none
initstring  : none
exit_after is: not set
exit is     : no
```

Figure 2 : Connexion UART à l'aide de `picocom` sur interface `/dev/ttyUSB0`

2.2.2 Lecture d'EEPROM (ex : ECU moto Red Key, BIOS, consoles de jeux)

Dans certains systèmes embarqués critiques, comme les **ECU (Electronic Control Unit)** de véhicules, des données sensibles peuvent être stockées dans des puces **EEPROM** de type **24C02, 24C04**, etc.

Ces mémoires non volatiles contiennent parfois :

- des **codes de démarrage** (comme la **Red Key** sur certaines motos),
- des **données d'appairage de clé**,
- ou encore des **informations d'identification système**.

L'attaque consiste à **accéder physiquement à la puce EEPROM**, souvent soudée directement sur la carte de l'ECU, afin de **lire son contenu** avec un lecteur externe, comme un programmeur **I2C/SPI (CH341A, MiniPro TL866, etc.)**.



Figure 3 : Lecteur d'eprom sur un site marchant connu

Après extraction, les données peuvent être :

- **analysées avec un éditeur hexadécimal** pour rechercher des chaînes spécifiques (codes d'accès, identifiants, etc.),
- **utilisées pour cloner des clés, contourner un anti-démarrage**, ou même démarrer un véhicule sans clé officielle.

Cette technique ne se limite pas au domaine automobile.

Elle est également utilisée dans d'autres contextes, par exemple :

- **Lecture de mot de passe BIOS** : sur des ordinateurs portables anciens, le mot de passe d'accès au BIOS est parfois stocké en clair dans une EEPROM, rendant possible son extraction et son contournement sans intervention logicielle.
- **Modification (modding) de consoles de jeux** : certaines consoles de jeux vidéo (anciennes générations comme la PlayStation 1, PlayStation 2, Xbox originale) stockent des informations sensibles (firmware, paramètres régionaux, codes de verrouillage) dans des EEPROM. En lisant ou en modifiant le contenu, il est possible de :
 - **contourner des protections anti-copie**,
 - ou **installer des firmwares personnalisés**.

Dans tous ces cas, **la maîtrise de l'accès physique à l'EEPROM ouvre la porte à de nombreuses attaques**, démontrant l'importance de protéger physiquement et logiquement ces composants dans un système embarqué.

2.2.3 Attaques par accès physique (glitch, cold boot, etc.)

Les attaques par accès physique exploitent directement l'environnement **matériel ou électrique** du système embarqué. Elles ne ciblent pas forcément un protocole ou une mémoire spécifique, mais plutôt les **comportements anormaux** d'un système lorsqu'il est perturbé, souvent à un niveau très bas. Voici deux types d'attaques particulièrement notoires :

- **Glitch attacks (ou fault injection)**

Les attaques par glitch consistent à **injecter volontairement des erreurs** (souvent électriques ou temporelles) dans un système pour le forcer à adopter un comportement non prévu par son concepteur. Par exemple :

- provoquer une **erreur de vérification d'authenticité**,
- contourner un **mot de passe firmware**,
- ou encore **outrepasser un contrôle d'accès sécurisé**.

Ces attaques peuvent être menées en modifiant brièvement la tension d'alimentation (voltage glitching), en perturbant l'horloge (clock glitching), ou même par injection de **laser ou d'électromagnétisme** dans des environnements plus avancés. Bien que complexes, ces techniques ont démontré leur efficacité contre de nombreux dispositifs embarqués non protégés.

- **Cold boot attacks**

Cette attaque vise les composants **volatile** comme la RAM, dans laquelle des données sensibles peuvent persister quelques secondes après extinction du système. En refroidissant la mémoire et en la transférant rapidement vers un autre système, il est parfois possible de **récupérer des clés de chiffrement ou mots de passe**.

Bien que moins courante dans les systèmes embarqués très contraints et un peu vieilles, cette attaque reste un exemple emblématique des failles possibles liées à **l'environnement physique d'un appareil**.

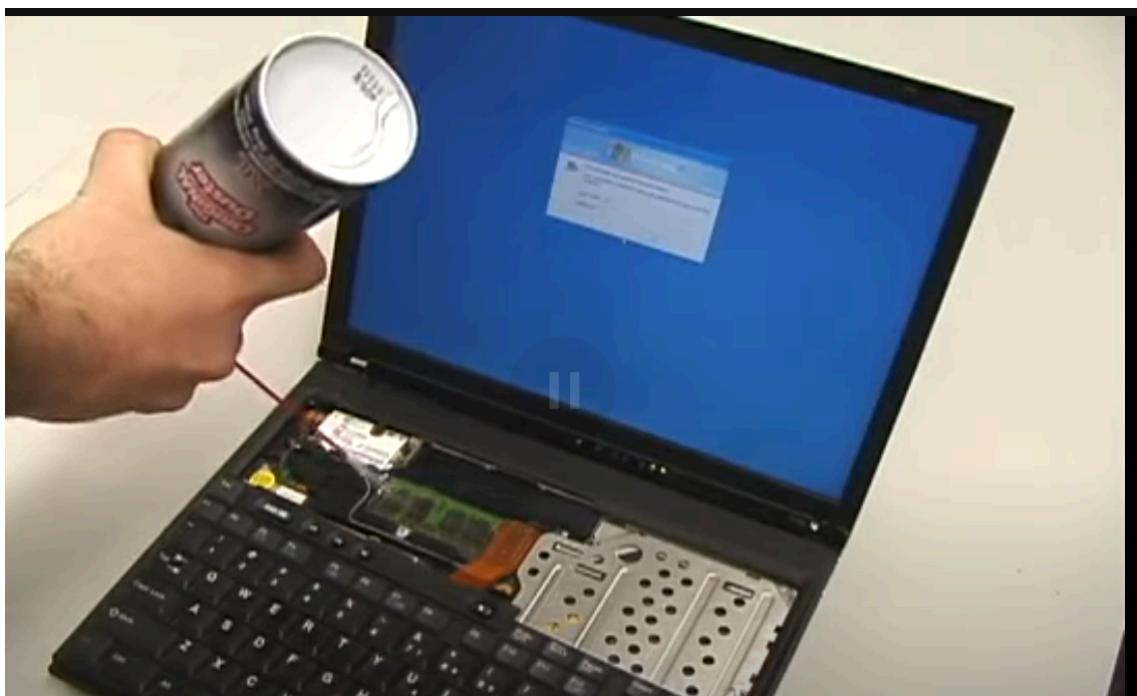


Figure 4 : Gèle de la ram avec spray pour conserver ses données

À l'époque, on pouvait geler la RAM avec un spray pour ralentir l'évaporation des données, rebrancher la RAM sur un autre ordinateur et trouver les clés de chiffrement BitLocker.

Aujourd'hui, plus besoin de refroidir la RAM : il suffit de brancher une clé USB avec les bons outils en précisant qu'il faut quand même être rapide . On peut retrouver la clé et déchiffrer le disque dur.

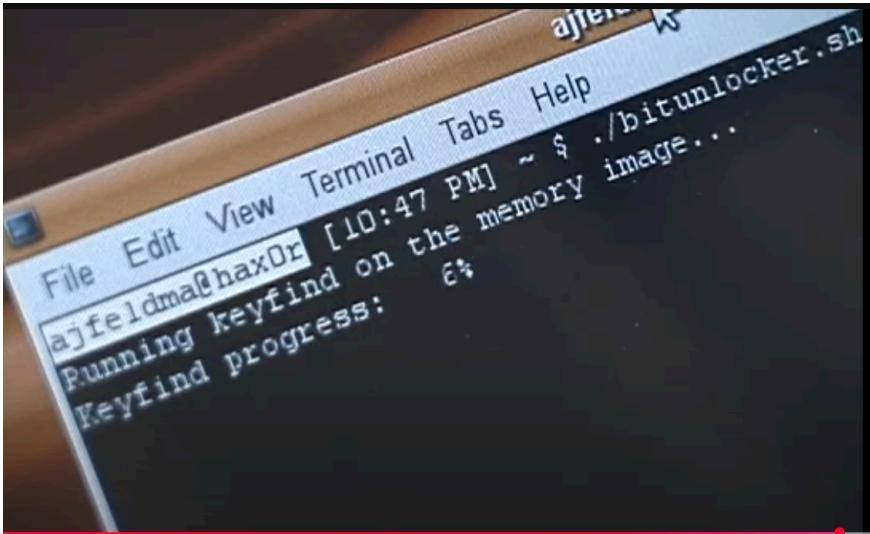


Figure 5 : Exécution de l'outil `bitunlocker.sh` sur une image mémoire

Cette capture d'écran montre le lancement d'un script nommé `bitunlocker.sh` conçu pour retrouver les clés de chiffrement de BitLocker en scannant une image mémoire obtenue lors d'une attaque de type *Cold Boot*. L'outil utilise un algorithme de recherche appelé `keyfind` pour extraire les clés présentes en RAM, en contournant ainsi la protection de chiffrement du disque.

Pourquoi est-il utile de parler de ce type d'attaque dans les pc?

Parce que certains systèmes embarqués, comme des tablettes industrielles ou des PC embarqués, peuvent eux aussi utiliser BitLocker.

Et que ces attaques physiques sont transposables à partir du moment où l'on possède de la ram.

2.2.4 Attaques via périphériques USB

Les attaques via périphériques USB exploitent la confiance implicite qu'accorde un système à tout dispositif branché physiquement sur ses ports. Plusieurs techniques permettent d'exécuter du code malveillant ou de causer des dégâts matériels par simple connexion d'un câble ou d'une clé



Figure 6 : USB rubber Ducky

- **Injection de charges utiles (USB Rubber Ducky, reverse shell, virus personnalisés) :**

Un périphérique USB, comme le **Rubber Ducky**, se fait passer pour un **clavier HID** et injecte automatiquement des frappes clavier programmées. Cela permet, par exemple, d'ouvrir un terminal, d'exécuter un script, ou d'établir un **reverse shell** pour prendre le contrôle du système à distance, sans alerter l'utilisateur.

Au-delà de simples commandes, une clé USB peut également être utilisée pour **injecter des charges utiles complexes**, comme des **malwares personnalisés**.

- **Attaque furtive via câbles modifiés (OMG Cable) :**

Les **câbles OMG** ressemblent extérieurement à de simples câbles de recharge ou de synchronisation USB classiques. En réalité, ils embarquent un microcontrôleur capable d'injecter des commandes HID, de capturer des frappes clavier, ou d'ouvrir un accès réseau distant une fois connectés.



- **Destruction matérielle (USB Killer) :**

Le **USB Killer** est un dispositif capable d'envoyer des décharges électriques rapides et puissantes sur les lignes de données USB, provoquant des

dommages physiques irréversibles sur la carte mère, rendant l'appareil inutilisable.



Figure 8 : USB KILLER

Exemple connu d'attaque par usb via malware : Stuxnet a infecté des systèmes industriels en Iran en utilisant des **clés USB malveillantes** pour pénétrer des réseaux isolés (air-gapped) et compromettre directement des **automates industriels (PLC Siemens)**, démontrant ainsi que des attaques USB peuvent viser et contrôler des **systèmes embarqués critiques**.

2.3 Défenses physiques

Pour répondre aux attaques décrites précédemment, de nombreuses **contre-mesures matérielles** et logicielles ont été développées. Ces défenses visent à rendre l'accès physique plus difficile, à **protéger les interfaces sensibles**, ou à **déetecter les manipulations anormales**.

2.3.1 Protection des interfaces

Les interfaces comme **UART**, **SPI**, **JTAG** ou **I²C** sont essentielles durant la phase de développement, mais deviennent des vecteurs d'attaque s'il reste possible d'y accéder une fois l'appareil déployé. En effet, un attaquant peut s'en servir pour extraire le firmware, injecter du code ou manipuler la configuration système.

Parmi les bonnes pratiques à mettre en place :

- **Désactivation des interfaces non utilisées** : tout port inutile devrait être désactivé via les paramètres du microcontrôleur ou par fusibles matériels. Chaque interface active constitue une surface d'attaque supplémentaire.

- **Authentification d'accès** : si une interface doit rester active, elle doit exiger une authentification robuste avant tout échange (clé, mot de passe ou signature).
- **Protection physique** : l'accès aux broches peut être rendu difficile grâce à l'utilisation de **résine epoxy**, ou en plaçant les pistes critiques en couches internes de la carte PCB.
- **Détection d'intrusion** : certains systèmes embarqués intègrent des capteurs capables de détecter l'ouverture d'un boîtier ou une tentative de branchement sur une interface de debug, déclenchant des actions comme l'effacement automatique des données sensibles.

Ces mesures doivent être combinées avec des politiques de test rigoureuses, notamment des audits réguliers sur la configuration matérielle effective.

2.3.2 Sécurisation des données (chiffrement, blindage)

Les données critiques (clés, identifiants, firmware) sont souvent stockées en clair dans des mémoires EEPROM, Flash ou RAM. Si ces composants sont accessibles physiquement, l'attaquant peut tenter une lecture directe, voire une attaque plus avancée de type **cold boot**, **side-channel**, ou **glitch**.

Pour contrer ces scénarios :

- **Chiffrement matériel** : les clés et données sensibles doivent être protégées par chiffrement AES, exécuté au sein de **modules matériels dédiés** tels que les **TPM**, **Secure Elements**, ou **HSM** embarqués. Ces composants empêchent l'extraction des clés même en cas d'accès physique au système.
- **Protection mémoire** : certains microcontrôleurs supportent les **MPU (Memory Protection Unit)** ou **ARM TrustZone**, permettant de cloisonner les zones mémoire critiques et de les protéger contre les lectures non autorisées.
- **Contre-mesures physiques** : pour les attaques par **canaux auxiliaires** (analyse de consommation, de température ou d'émissions électromagnétiques), des protections comme **l'ajout de bruit**, le **blindage électromagnétique**, ou **l'égalisation de la consommation** (masking) permettent de réduire fortement le risque d'exfiltration de données cryptographiques.
- **Firmware chiffré** : même si l'accès mémoire est possible, chiffrer le firmware (et l'utiliser avec **Secure Boot**) empêche l'analyse ou la modification du disque.

Ces approches doivent être adaptées à la puissance disponible sur le SoC ou MCU, en tenant compte des ressources limitées des systèmes embarqués.

2.3.3 Limitation des accès (boîtier, firmware sécurisé)

Au-delà des composants électroniques, la **protection physique du dispositif entier** est essentielle. L'attaquant ne peut pas manipuler ce qu'il ne peut pas ouvrir ou atteindre facilement.

- **Boîtiers scellés** : utiliser des coques inviolables, des vis anti-retour, ou des soudures étanches limite les possibilités d'ouverture non autorisées.
- **Capteurs de sabotage** : certains systèmes détectent l'ouverture du boîtier ou un changement d'environnement (température, lumière, champ magnétique), ce qui peut déclencher un effacement d'urgence des données critiques ou alerter un serveur distant.
- **Secure Boot** : le système de démarrage doit vérifier la signature cryptographique du firmware avant son exécution. Cela empêche un attaquant d'injecter un firmware modifié ou malveillant.
- **Mise à jour sécurisée** : toutes les mises à jour doivent être **signées numériquement**, et idéalement **chiffrées**, pour éviter l'installation de logiciels compromis.
- **Réduction de la surface d'attaque** : comme le recommande Cyberphinx (<https://cyberphinx.de/blog/embedded-security-basics/>), il est essentiel de désactiver toutes les fonctions non utilisées, y compris les services de debug ou les interfaces réseau inutiles.

2.3.4 Défense physique dans les environnements industriels (SCADA, ICS)

Définitions

Les systèmes **SCADA** (Supervisory Control And Data Acquisition) et **ICS** (Industrial Control Systems) sont des infrastructures utilisées pour **surveiller, automatiser et piloter** des processus industriels critiques à distance.

Un système SCADA comprend généralement :

- des **capteurs et actionneurs** sur le terrain (niveau physique),
- des **automates programmables** (PLC) ou **unités distantes** (RTU),

- un ou plusieurs **serveurs de supervision** qui collectent, analysent et affichent les données via une interface HMI (Human-Machine Interface),
- des **réseaux industriels** utilisant des protocoles comme Modbus, DNP3, OPC-UA ou Profinet.

Les ICS sont un terme plus général qui englobe SCADA mais aussi les systèmes DCS (Distributed Control Systems) et autres systèmes d'automatisation industrielle.

Pour en savoir plus :
<https://www.fortinet.com/fr/resources/cyberglossary/scada-and-scada-systems>

Exposition physique accrue

Contrairement aux équipements IT traditionnels, les composants SCADA/ICS sont souvent installés :

- en extérieur ou dans des zones industrielles isolées,
- avec peu ou pas de surveillance humaine continue,
- parfois sans réelle séparation entre réseau opérationnel et réseau bureautique.

Cela les rend vulnérables à des **attaques physiques ciblées**, même sans connexion Internet.

Mesures de défense physique spécifiques

1. Coffrets et armoires renforcées

Les équipements critiques (PLC, modules d'I/O, routeurs industriels, serveurs SCADA) doivent être enfermés dans des **coffrets verrouillés**, parfois blindés ou scellés. Ces boîtiers peuvent être équipés de **capteurs d'ouverture** ou de **systèmes d'alarme** déclenchés en cas de tentative d'intrusion.

2. Contrôle des accès physiques

Les installations doivent disposer de **mécanismes d'accès restreints** :

- clés mécaniques ou électroniques,
- **badges RFID nominatifs** pour identifier les interventions,
- et, idéalement, **vidéosurveillance locale** ou télésurveillance à distance.

3. Désactivation des interfaces sensibles

Les ports de programmation (USB, série, Ethernet) doivent être désactivés, masqués ou physiquement bloqués quand ils ne sont pas utilisés. Il est courant

d'utiliser des **cache-ports**, ou de les désactiver via le BIOS ou firmware des équipements industriels.

4. Disques chiffrés et démarrage sécurisé

Lorsque des systèmes SCADA s'exécutent sur des PC industriels (sous Windows 10 IoT ou Linux), il est recommandé d'utiliser **BitLocker** ou **LUKS**, en association avec un **Secure Boot**, pour empêcher l'exécution de firmware modifié ou l'extraction de données sensibles.

5. Surveillance et journalisation des accès

Un système SCADA bien conçu doit **enregistrer tout accès physique ou logique** : branchement d'un périphérique, modification d'un programme automate, mise à jour d'un firmware. Ces journaux doivent être centralisés et protégés contre la falsification.

Exemple concret

En 2000, l'incident de **Maroochy Water Services (Australie)** a révélé la gravité des attaques physiques sur SCADA. Un ancien technicien a utilisé un ordinateur portable et une antenne radio pour se connecter aux automates à distance et provoquer le déversement d'eaux usées dans l'environnement.

L'absence de chiffrement, d'authentification sur le protocole radio, et de contrôle d'accès physique ont permis cette attaque sans jamais pénétrer physiquement le site.

3 Cybersécurité radio et sans-fil

Les systèmes embarqués et objets connectés reposent de plus en plus sur des **communications sans fil** pour échanger des données, interagir avec leur environnement ou être administrés à distance. Cette connectivité introduit de nouveaux risques : les transmissions peuvent être interceptées, falsifiées ou perturbées, souvent **sans nécessiter un accès physique direct** à l'appareil.

Dans cette partie, on s'intéresse aux principaux **protocoles radio utilisés** (Wi-Fi, Bluetooth, NFC, RF propriétaire), à leurs **vulnérabilités connues**, et aux **techniques d'attaque et de défense** spécifiques à ce type de communication.

3.1 Protocoles concernés

3.1.1 Wi-Fi (WEP, WPA2, Radius, WPA3)

3.1.1.1 Présentation protocole WEP

Le protocole **WEP (Wired Equivalent Privacy)** a été introduit avec la norme Wi-Fi initiale **IEEE 802.11** en 1997, afin de garantir la **confidentialité des communications sans-fil**.

Son objectif était de rendre les transmissions radio aussi sûres que celles passant par un câble Ethernet.

Pour sécuriser les communications, WEP utilise :

- **L'algorithme de chiffrement RC4** (un chiffrement par flux rapide),
- combiné à une **clé secrète partagée** entre les stations (ex : routeur Wi-Fi et clients).

Chaque message (trame Wi-Fi) est chiffré en combinant :

- une **clé WEP fixe** (40 ou 104 bits),
- avec un **vecteur d'initialisation (IV)** de 24 bits, afin de générer une **clé de chiffrement temporaire** unique pour chaque paquet.

Le processus standard est :

1. **Générer une clé temporaire** en concaténant l'IV + la clé WEP partagée.
2. **Utiliser RC4** avec cette clé temporaire pour chiffrer le paquet.
3. **Envoyer le paquet** en incluant l'IV en clair pour que le destinataire puisse refaire la clé temporaire et déchiffrer.

Faiblesses structurelles du WEP

Malgré ses ambitions initiales, WEP souffre de **plusieurs failles de conception majeures**, rendant le protocole **facilement cassable** aujourd'hui.

a) IV trop court (24 bits)

L'**IV (Initialization Vector)** est une valeur aléatoire censée assurer que deux paquets ne soient jamais chiffrés de manière identique.

Cependant, avec seulement **24 bits**, il existe au maximum **16 millions de combinaisons possibles** — un nombre faible dans un réseau actif.

Ainsi :

- Lorsqu'un réseau WEP fonctionne longtemps, **des collisions d'IV** se produisent (deux paquets utilisent le même IV).

- Cela génère **des répétitions dans le chiffrement**, exploitées par les attaquants.

Problème :

L'IV est **envoyé en clair** dans chaque trame Wi-Fi pour permettre au récepteur de recalculer la clé temporaire.

Cela offre **directement aux attaquants** les informations nécessaires pour analyser et attaquer.

b) Répétition des IV et attaques statistiques

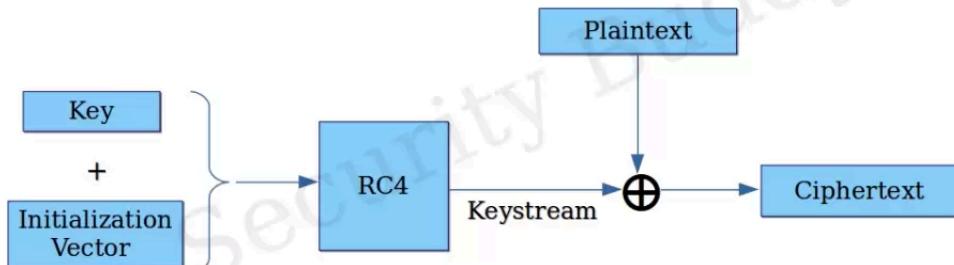
Les outils de craquage Wi-Fi (ex : **aircrack-ng**, **aireplay-ng**) exploitent les répétitions d'IV pour :

- Capturer **des centaines de milliers** voire **des millions** de paquets chiffrés,
- Observer les répétitions et patterns,
- **Reconstituer la clé secrète WEP.**

Plus le trafic est élevé, plus l'attaque est rapide :

- Avec de **l'injection de trafic** (ex : aireplay-ng), on peut forcer la génération de paquets,
- Ce qui accélère fortement la récupération de la clé.

WEP Encryption



The Security Buddy
<https://www.thesecuritybuddy.com/>

Figure 9 : Schema du chiffrement d'un message avec WEP

Résultat pratique

Une fois la clé WEP extraite :

- L'attaquant peut **se connecter librement** au réseau Wi-Fi,
- Capturer, modifier ou injecter du trafic,
- Accéder à des ressources internes (imprimantes, NAS, ordinateurs, etc.).

Le temps pour casser une clé WEP dépend du volume de paquets capturés **Quelques minutes** suffisent en pratique, même avec du matériel grand public.

3.1.1.2 Présentation du protocole WPA2

Présentation du protocole WPA2

WPA2 (Wi-Fi Protected Access II) est un protocole de sécurité standardisé en 2004 par la Wi-Fi Alliance pour remplacer WPA et renforcer la protection des communications sans fil.

Il vise à assurer la confidentialité, l'intégrité et l'authenticité des échanges sur les réseaux Wi-Fi. WPA2 repose sur l'utilisation du chiffrement **AES (Advanced Encryption Standard)** pour protéger les données, associé à un mécanisme d'authentification par **4-Way Handshake** utilisant le protocole **EAPOL** (Extensible Authentication Protocol over LAN).

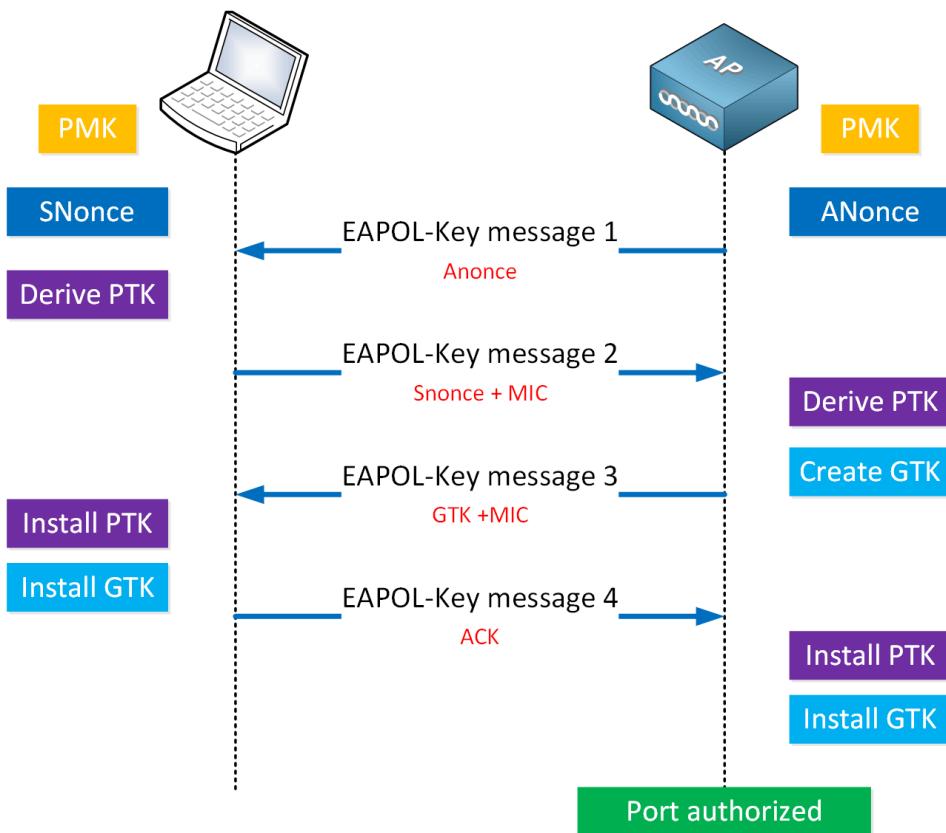


Figure 10 : Schéma du 4-Way Handshake WPA2 (protocole EAPOL)

Lorsqu'un appareil souhaite se connecter à un point d'accès WPA2, un échange en quatre étapes a lieu pour vérifier l'authenticité et établir une clé de session temporaire. Cet échange, appelé **4-Way Handshake**, peut être observé dans un analyseur de paquets comme **Wireshark**, où il est possible de visualiser les **quatre paquets EAPOL** correspondant à chaque étape de l'authentification (voir capture Wireshark)

sniffpmkid_2.pcap						
No.	Time	Source	Destination	Protocol	Length	Info
629	15.210081	FreeboxS_32:78:1e	a8:31:62:2d:48:22	EAPOL	137	Key (Message 1 of 4)
630	15.210995	FreeboxS_32:78:1e	a8:31:62:2d:48:22	EAPOL	137	Key (Message 1 of 4)
632	15.213035	a8:31:62:2d:48:22	FreeboxS_32:78:1e	EAPOL	157	Key (Message 2 of 4)
633	15.214678	FreeboxS_32:78:1e	a8:31:62:2d:48:22	EAPOL	193	Key (Message 3 of 4)
634	15.219370	a8:31:62:2d:48:22	FreeboxS_32:78:1e	EAPOL	135	Key (Message 4 of 4)
638	15.582494	FreeboxS_32:78:1e	SeikoEps_56:9e:cc	EAPOL	137	Key (Message 1 of 4)
643	15.596922	SeikoEps_56:9e:cc	FreeboxS_32:78:1e	EAPOL	159	Key (Message 2 of 4)
673	16.351544	FreeboxS_32:78:1e	SeikoEps_56:9e:cc	EAPOL	137	Key (Message 1 of 4)

Figure 11 : Exemple de lecture de paquet EAPOL sur wireshark

Le premier message est envoyé par le point d'accès pour fournir un nonce (nombre aléatoire) au client. Le client utilise ce nonce, associé à sa propre clé pré-partagée

(PSK), pour générer une clé temporaire appelée PTK (Pairwise Transient Key) et renvoie son propre nonce. Le point d'accès confirme ensuite la validité de la clé, et enfin le client valide à son tour la connexion. À l'issue de ce processus, une session sécurisée est établie.

Bien que robuste, ce mécanisme présente une faiblesse liée aux attaques de désauthentification. En exploitant le fait que les trames de gestion ne sont pas protégées dans WPA2 classique, un attaquant peut envoyer des fausses trames de désauthentification (Deauth Attack) pour forcer une victime à se reconnecter. Cette reconnexion permet alors de capturer le 4-Way Handshake complet et de lancer une attaque hors-ligne pour tenter de casser la clé PSK à l'aide d'outils comme aircrack-ng ou hashcat.

Dans les réseaux d'entreprise, une variante plus sécurisée du WPA2 est utilisée : le **WPA2-Enterprise**. Contrairement au mode personnel, qui repose sur une clé pré-partagée unique pour tous les utilisateurs, WPA2-Enterprise s'appuie sur un serveur **Radius** pour authentifier chaque client individuellement. Le point d'accès agit comme un relais entre l'utilisateur et le serveur Radius, qui vérifie les identifiants et attribue des clés de session spécifiques à chaque appareil connecté.

L'usage d'un serveur Radius renforce considérablement la sécurité par rapport à WPA2-Personal. Chaque utilisateur dispose de son propre canal sécurisé, et l'attaque par capture du handshake devient nettement plus complexe à réaliser.

Voici des rfc pertinentes

RFC 3748 – Extensible Authentication Protocol (EAP)

Voici une RFC intéressante pour comprendre les mécanismes d'authentification utilisés dans les réseaux sans fil sécurisés. Elle décrit le protocole EAP, qui est à la base du 4-Way Handshake dans WPA/WPA2 Enterprise. Ce protocole extensible permet une grande variété de méthodes d'authentification, notamment lorsqu'un serveur RADIUS est impliqué.

et **RFC 2865 – RADIUS (Remote Authentication Dial-In User Service)**

RFC intéressante qui définit le protocole RADIUS, utilisé pour gérer l'authentification centralisée des utilisateurs dans les réseaux. Elle est notamment utilisée dans les déploiements professionnels de WPA2-Enterprise, en remplacement de la simple clé pré-partagée.

3.1.1.3 Présentation protocole WPA3

Présentation du protocole WPA3

WPA3 est la dernière évolution du standard de sécurisation des réseaux Wi-Fi, officialisée en 2018 par la Wi-Fi Alliance.

Son objectif est de corriger les vulnérabilités identifiées dans WPA2, notamment celles concernant l'attaque KRACK (Key Reinstallation Attack), et de renforcer la confidentialité et l'authentification des communications sans fil.

WPA3 repose sur le chiffrement AES, tout comme WPA2, mais introduit également un nouvel algorithme, **AES-GCMP**, qui améliore la protection des données. La taille des clés utilisées peut atteindre **256 bits** pour les environnements nécessitant un niveau de sécurité très élevé.

Le mécanisme d'authentification est également profondément revu avec l'introduction de **SAE (Simultaneous Authentication of Equals)**. Ce protocole remplace la simple clé pré-partagée par un processus d'échange sécurisé basé sur des preuves cryptographiques, rendant les attaques par dictionnaire bien plus difficiles, même lorsque les mots de passe choisis sont faibles.

WPA3 améliore également la protection des données sur les réseaux publics ouverts avec la fonctionnalité **Opportunistic Wireless Encryption (OWE)**, qui chiffre les communications sans nécessiter d'authentification préalable. Enfin, le standard impose le chiffrement des trames de gestion grâce aux **Protected Management Frames (PMF)**, ce qui rend les attaques de désauthentification et de déconnexion beaucoup plus compliquées qu'avec WPA2.

Malgré ses avancées, WPA3 reste encore en cours de déploiement dans de nombreux environnements. Sa mise en œuvre est plus complexe, nécessitant des équipements compatibles et parfois des ajustements importants sur les infrastructures existantes.

On peut d'ailleurs voir sur le site <https://wigle.net/stats> l'utilisation des différents protocoles dans le monde avec une dominance pour wpa2 et wpa3 qui reste minoritaire même devant wep qui est beaucoup moins sécurisé

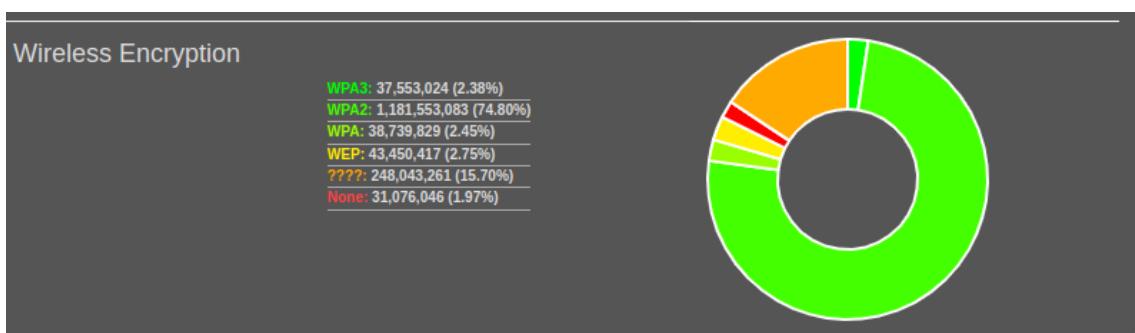


Figure 12 : Diagramme utilisation protocole wifi (wpa , 2 , 3 et wep)

3.1.2 Bluetooth classique et BLE

3.1.2.1 Définition et différences

Le Bluetooth est une technologie de communication sans fil à courte portée utilisée pour échanger des données entre appareils électroniques. Il est présent dans une large gamme de dispositifs, allant des casques audio aux objets connectés industriels. On distingue principalement deux versions : le **Bluetooth classique** et le **Bluetooth Low Energy (BLE)**.

Le **Bluetooth classique**, aussi appelé BR/EDR (Basic Rate / Enhanced Data Rate), est conçu pour les communications continues et stables entre deux appareils. Il est utilisé dans des contextes où un débit relativement élevé est nécessaire, comme la transmission audio, les transferts de fichiers ou la connexion de périphériques (souris, claviers, haut-parleurs). Ce type de Bluetooth consomme davantage d'énergie mais permet des connexions soutenues et fiables.

Le **Bluetooth Low Energy (BLE)**, introduit avec la norme Bluetooth 4.0, a été développé pour répondre aux besoins des objets connectés à faible consommation. Il permet des échanges brefs et intermittents, adaptés à des dispositifs tels que les capteurs, les bracelets de suivi d'activité, les balises de localisation ou les systèmes de domotique. BLE utilise un mécanisme de diffusion appelé advertising, qui permet à un périphérique d'envoyer périodiquement des informations sans avoir besoin d'établir une connexion formelle.

Alors que le Bluetooth classique privilégie la stabilité et la continuité de la liaison, BLE mise sur la légèreté, l'autonomie et la rapidité de communication pour des interactions courtes et peu gourmandes en énergie.

Les deux protocoles peuvent coexister dans un même appareil, mais ils reposent sur des architectures différentes et ne sont pas compatibles entre eux au niveau des communications. Chacun présente des avantages selon les usages visés : Bluetooth classique pour les applications nécessitant de la bande passante, BLE pour les échanges ponctuels et économies en énergie.

3.1.3 NFC et RFID

Les technologies **RFID** (Radio Frequency Identification) et **NFC** (Near Field Communication) permettent des communications sans fil à courte portée, utilisées principalement pour l'identification, l'accès, le paiement ou la transmission rapide d'informations.

RFID

La RFID est une technologie d'identification par radiofréquence qui permet à un lecteur de détecter et lire une puce sans contact direct ni ligne de vue. Elle se divise en plusieurs catégories selon la fréquence :

- **LF (Low Frequency, ~125 kHz)** : utilisée pour les badges d'accès simples ou les puces d'identification animale.
- **HF (High Frequency, 13,56 MHz)** : la plus utilisée pour les cartes sans contact et compatible avec la norme ISO/IEC 14443.
- **UHF (Ultra High Frequency)** : employée pour la logistique et le suivi d'inventaire.

Les tags RFID peuvent être passifs (sans batterie) ou actifs (avec alimentation intégrée).

NFC

Le **NFC** est une forme spécialisée de RFID HF, conçue pour des échanges bidirectionnels entre deux appareils compatibles. Contrairement à la RFID, qui est souvent unidirectionnelle (lecteur vers badge), le NFC permet aux deux parties d'échanger des données, ce qui le rend adapté aux paiements sans contact, à l'appairage rapide (par exemple pour le Bluetooth), ou à l'échange de données entre smartphones.

Le NFC fonctionne à très courte portée (quelques centimètres) et repose sur les mêmes standards que la RFID HF, ce qui les rend compatibles à certains niveaux.

Cartes MIFARE

Les cartes **MIFARE** sont des cartes à puce sans contact très répandues dans les systèmes de transport, de contrôle d'accès, de billetterie ou d'identification. Développées par **NXP Semiconductors**, elles utilisent la norme ISO 14443-A et existent en plusieurs versions, offrant des niveaux de sécurité variés :

- **MIFARE Classic** : ancienne génération, facilement clonable avec des outils comme le Proxmark3 ou le Flipper Zero. Utilise un chiffrement propriétaire (Crypto-1) aujourd'hui considéré comme obsolète. Certain badge d'immeuble sont en mifare classic et sont facilement reproduisibles
- **MIFARE Plus** : amélioration progressive, offrant un mode de transition vers AES, mais reste vulnérable si utilisé en mode rétrocompatible.
- **MIFARE DESFire (EV1, EV2, EV3)** : famille moderne conçue pour les applications critiques. Elle utilise des mécanismes de sécurité avancés, comme

le **chiffrement AES-128**, la **mutual authentication**, et des **contrôles d'accès fins**. En particulier, **MIFARE DESFire EV3** propose des fonctions renforcées de détection d'attaques par canaux auxiliaires et une résistance élevée aux tentatives de clonage.

Ces cartes sont nettement plus résistantes aux attaques que les anciennes générations et restent à ce jour parmi les plus sûres du marché, si elles sont correctement configurées.

En dehors de la famille MIFARE, il existe de nombreuses autres cartes sans contact utilisées dans les environnements professionnels ou grand public. Les **cartes HID Prox** (125 kHz, non cryptées) sont très répandues dans les systèmes d'accès anciens, mais peuvent être clonées très facilement avec des outils comme le Proxmark3 ou le Flipper Zero. À l'inverse, les cartes **iCLASS** (HID, 13,56 MHz) offrent un meilleur niveau de sécurité, bien que certaines versions aient également été compromises par des attaques de rétro-ingénierie. On retrouve également des cartes **Calypso** dans les transports publics (utilisées par exemple en Île-de-France), qui reposent sur des standards ouverts avec des algorithmes de chiffrement robustes. D'autres technologies comme **FeliCa** (utilisée au Japon) ou les **cartes EM** (basiques, 125 kHz) complètent ce paysage hétérogène, avec des niveaux de sécurité très variables selon la génération et le protocole utilisé.



Figure 13 : Badge vigik , carte navigo et carte mifare

3.1.4 Protocoles RF propriétaires : fixed code vs rolling code

Les télécommandes sans fil utilisées pour contrôler des dispositifs comme des portails, des garages ou des lampes fonctionnent souvent sur des fréquences Sub-GHz (ex. 433,92 MHz) et utilisent des protocoles simples appelés **fixed code** (code fixe) ou plus sécurisés appelés **rolling code** (code tournant). La différence entre ces deux mécanismes détermine en grande partie leur résistance aux attaques.

Dans un système à **code fixe**, la télécommande envoie exactement le même signal à chaque pression sur le bouton. Il est donc trivial d'effectuer une **replay attack**, c'est-à-dire d'enregistrer le signal à l'aide d'un appareil comme le **Flipper Zero**, puis de le rejouer ultérieurement pour déclencher la même action (par exemple, allumer une lumière ou ouvrir une porte). Ce type d'attaque est réalisable avec un matériel peu coûteux et sans compétence avancée, car aucun chiffrement ni variation du signal n'est appliqué.

En revanche, les systèmes plus avancés utilisent un **rolling code**, dans lequel chaque appui sur la télécommande génère un code unique, calculé à partir d'un algorithme partagé entre l'émetteur et le récepteur. À chaque utilisation, un nouveau code est attendu, rendant inefficace la simple relecture d'un ancien signal. Par exemple, si l'appareil enregistre le code $n = 314$ et tente de le rejouer plus tard, le récepteur ne l'acceptera plus, car il attend déjà le code suivant (ex. $n = 628$). Ce mécanisme empêche donc la réutilisation d'un code déjà transmis.

Cependant, même les systèmes à rolling code peuvent être vulnérables à une technique appelée **rolljam**. Cette attaque consiste à utiliser un brouilleur radio (signal jammer) pour empêcher la réception du code par la cible tout en l'enregistrant. Lorsque l'utilisateur appuie sur sa télécommande, le code est bloqué mais intercepté. Il appuie donc une deuxième fois, générant un nouveau code également intercepté. Ensuite, l'attaquant désactive le brouilleur et envoie le premier code intercepté, qui est toujours valide car le récepteur ne l'a jamais reçu initialement. Il dispose alors d'un **code valide en réserve**, qu'il pourra utiliser plus tard pour ouvrir le dispositif à distance.

Ainsi, bien que les rolling codes offrent un **niveau de sécurité nettement supérieur**, ils ne sont pas totalement invulnérables. La sécurité dépend autant du protocole que de l'implémentation, et certains systèmes peuvent être mal configurés ou trop tolérants aux désynchronisations, ce qui permet parfois des attaques avec un **Flipper Zero ou un enregistreur radio spécialisé**. Les fabricants doivent donc non seulement utiliser des algorithmes robustes, mais aussi mettre en place des garde-fous tels que des fenêtres de synchronisation limitées, un compteur de tentatives, et une authentification mutuelle renforcée.

Voici un schéma provenant de la vidéo [Rolling codes explained #flipperzero](#) qui explique le fonctionnement d'un **système à rolling code**, utilisé dans des télécommandes sécurisées (par exemple pour ouvrir un garage). Chaque pression sur la télécommande génère un nouveau code basé sur une formule commune avec le récepteur (ici : $n \times \pi \times 100$). Le récepteur n'accepte que le **code attendu à ce moment-là**. Une fois un code utilisé (comme **#628**), il devient invalide. Si un attaquant (comme avec un Flipper Zero) enregistre ce code pour le rejouer plus tard, le système le rejettéra car il a déjà été consommé. Ce mécanisme empêche les **replay attacks classiques**.

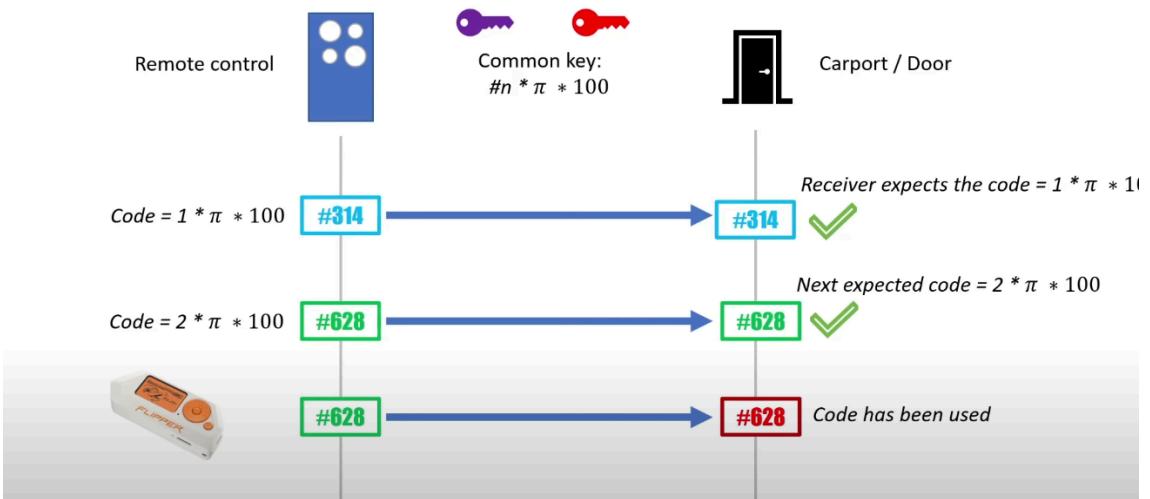


Figure 14: Fonctionnement du Rolling Code pour la sécurisation des télécommandes sans fil

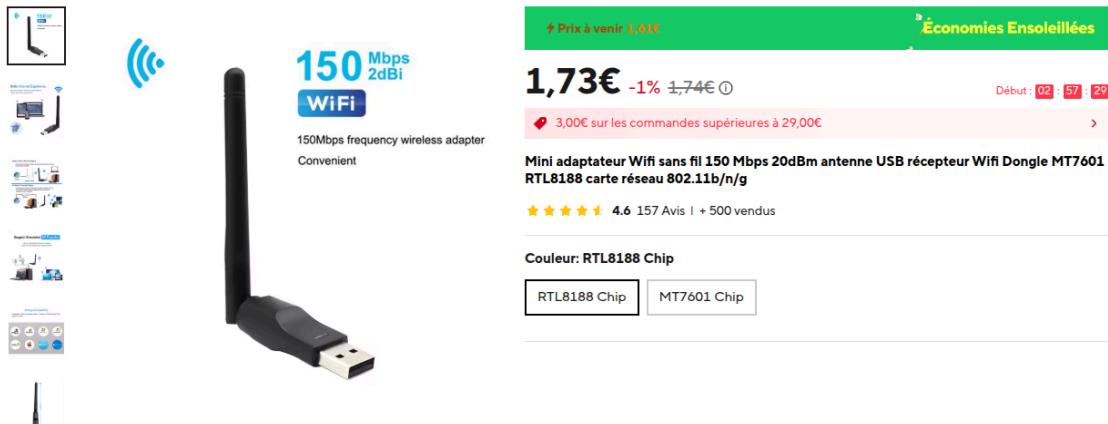
3.2 Attaques radio courantes

3.2.1 Attack sur Wi-Fi WEP et WPA2 (Aircrack , Deauth Flipper, ESP32)

Une fois les failles des protocoles WEP et WPA2 comprises, il est important de montrer comment elles sont exploitées dans la pratique, notamment à l'aide d'outils accessibles tels que **Aircrack-ng**, ou des dispositifs compacts comme le **Flipper Zero** et l'**ESP32 Deauther**.

Pour **WEP**, l'exploitation repose sur la **capture de paquets chiffrés**. En utilisant une carte Wi-Fi USB compatible "monitor mode"

C'est une attaque extrêmement simple, automatisable et peu chère



source : [Achetable ici pour 1€73](#)

il est possible, via la suite Aircrack-ng, de capturer un grand nombre de trames contenant les IV (vecteurs d'initialisation). Si le trafic est insuffisant, des outils comme `aireplay-ng` permettent de générer artificiellement des paquets (injection ARP). Une fois plusieurs milliers de paquets collectés, `aircrack-ng` est capable de reconstituer la clé WEP par une attaque statistique.

Pour **WPA2**, la méthode consiste à **forcer un client à se reconnecter au réseau** pour intercepter le fameux **4-Way Handshake**, puis à le casser en local. Cela se fait en 3 étapes :

1. Capture du trafic avec `airodump-ng` (interface en mode monitor).
2. Envoi de paquets de désauthentification (`aireplay-ng --deauth`) pour déconnecter un client.
3. Interception du handshake et attaque hors-ligne avec `aircrack-ng -w dictionnaire.txt`.

La **qualité du mot de passe** détermine le succès de cette attaque. Un mot de passe trop simple peut être cassé rapidement, tandis qu'une phrase de passe robuste résistera à toutes les tentatives, même longues.

En parallèle, des outils comme **Flipper Zero** ou **ESP32 Deauther** facilitent les attaques de désauthentification. Ils permettent de **forcer la reconnexion d'un client** à un point d'accès et de capturer les paquets importants (4 ways handshakes) — lequel pourra ensuite être utilisé avec des outils plus puissants comme Aircrack ou Hashcat.

Explication en schéma

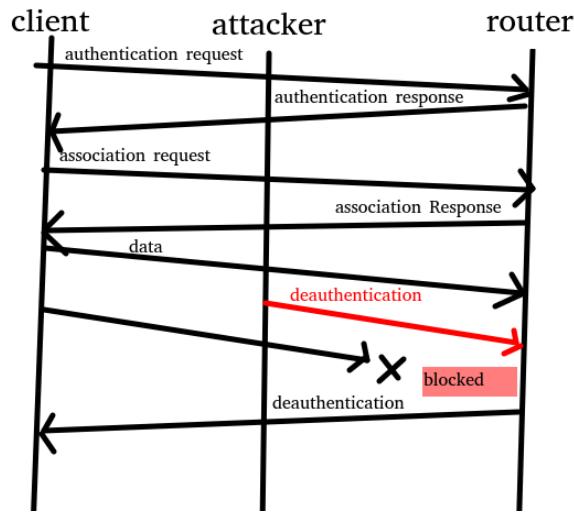


Figure 15: Attaque de désauthentification (Deauth Attack)

Ce mécanisme est particulièrement redoutable sur les réseaux d'objets connectés (caméras, imprimantes Wi-Fi...), qui utilisent souvent des mots de passe faibles et ne mettent pas en œuvre de mécanismes comme **802.11w** (Protected Management Frames). Un exemple concret d'exploitation de cette faille sera présenté dans la section **5.4 Deauth Attack sur caméras Wi-Fi**.

3.2.2 Bluetooth Attack : attaque DoS BLE sans appairage (iPhone) et mouse jacking

Exploitation via le Flipper Zero

Le **Flipper Zero** est un appareil multifonctionnel de test de sécurité, capable d'interagir avec divers protocoles sans fil. En utilisant un firmware tiers, tel que **Xtreme**, (<https://github.com/Flipper-XFW/Xtreme-Firmware>) le Flipper Zero peut émettre des paquets BLE simulant des demandes de connexion provenant d'appareils Apple légitimes, tels que des faux AirTags ou des faux AirPods.

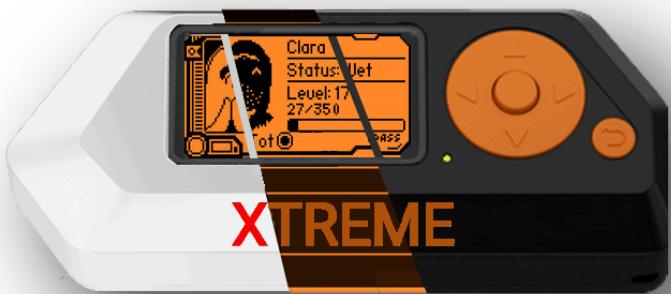


Figure 15 bis : flipper zero avec le firmware “XTREME”

Cette technique, connue sous le nom de "**BLE Spam**", consiste à inonder les iPhones à proximité de notifications de connexion Bluetooth, provoquant une surcharge de l'interface utilisateur. Avant la mise à jour iOS 17.2, cette attaque pouvait entraîner des ralentissements significatifs, des blocages temporaires et des redémarrages forcés des appareils ciblés.

La portée de cette attaque dépend de la puissance de l'émetteur et de l'environnement, mais des tests ont montré qu'elle pouvait affecter des appareils situés à plusieurs dizaines de mètres.

Correctif apporté par iOS 17.2

Apple a adressé cette vulnérabilité dans la mise à jour **iOS 17.2**, publiée en décembre 2023. Cette mise à jour a introduit des mécanismes limitant le nombre de notifications de connexion Bluetooth pouvant être reçues dans un intervalle de temps donné, réduisant ainsi l'efficacité de l'attaque.

Bien que quelques notifications puissent encore apparaître, elles ne suffisent plus à provoquer des dysfonctionnements majeurs des appareils. Une bonne pratique à l'époque était de désactiver le bluetooth quand il n'est pas utilisé.

David Bombal, le célèbre YouTubeur, nous montre dans cette courte vidéo comment l'attaque se déroule et comment on s'en protège.
<https://www.youtube.com/shorts/eoYNZBrlFak>

Le terme **MouseJacking** désigne initialement une vulnérabilité touchant les périphériques **souris et claviers sans fil non-Bluetooth**, utilisant la bande de **2,4 GHz** via un dongle USB. Comme l'a révélé la start-up de cybersécurité Bastille en

2016, cette faille affecte des produits de grandes marques comme Logitech, Dell ou Lenovo. Elle permet à un attaquant à distance (jusqu'à 100 mètres) d'injecter **des frappes clavier malveillantes**, même si le périphérique d'origine n'est qu'une souris. En effet, le dongle peut être trompé pour accepter des paquets se faisant passer pour un clavier HID. Il devient alors possible d'exécuter du code à distance, d'ouvrir un terminal, voire d'installer un malware ou un rootkit sur la machine ciblée. L'attaque s'appuie souvent sur une puce **Nordic Semiconductor nRF24L** mal configurée, dont le chiffrement n'est pas activé par défaut dans certains produits. L'exploitation ne requiert qu'un petit équipement radio peu coûteux et quelques lignes de code Python, comme le décrit [IT-Connect](#).

La vulnérabilité exploitée dans l'attaque dite **MouseJacking** provient principalement des **choix des constructeurs**, qui n'ont pas jugé nécessaire de chiffrer les communications entre le **dongle USB** et la **souris sans fil**. Contrairement aux claviers, pour lesquels le chiffrement est généralement implémenté afin d'éviter que des frappes sensibles ne soient interceptées, les souris ont été considérées comme peu critiques, puisque leurs signaux se limitent à des clics et des mouvements. Pour gagner en performance et réduire les coûts de développement, certains fabricants ont donc laissé ces échanges **en clair**, sans authentification ni chiffrement.

Cette négligence ouvre la porte à des attaques sérieuses sans que la victime ne soit fautive : il est possible d'écouter les échanges radio, de récupérer **l'adresse du dongle**, puis de **leurrer** ce dernier en se faisant passer pour la souris. L'attaquant peut alors injecter des mouvements ou des clics arbitraires, voire aller plus loin en **faisant passer la souris pour un clavier** (périphérique HID), ce qui lui permet d'exécuter du code ou des commandes sur la machine ciblée. L'effet est comparable à celui d'une **clé USB malveillante branchée physiquement**, ce qui démontre à quel point l'absence de chiffrement peut transformer un simple périphérique sans fil en **vecteur d'intrusion critique**.

Il ne faut toutefois **pas confondre cette attaque informatique** avec une autre forme de "**mouse jacking**", très médiatisée dans le domaine automobile. Comme l'explique un reportage de TF1 Info, cette expression est également utilisée pour décrire des techniques de vol de voiture **sans effraction**, qui exploitent les failles des systèmes électroniques embarqués. Dans ce cas, les voleurs interceptent le signal de la clé sans contact ou reprogramment une nouvelle clé à l'aide d'un outil de diagnostic pirate. Ils peuvent aussi capter le signal d'une clé rangée dans une maison à l'aide d'un amplificateur, et l'utiliser pour ouvrir et démarrer le véhicule en quelques secondes. En 2023, plus de 90 % des vols de voitures en France auraient été réalisés par ce type de méthode, selon TF1 Info. Cette forme de piratage requiert peu d'effort physique et repose sur l'exploitation directe du système embarqué du véhicule, comme l'a démontré un cas documenté dans un garage de Meurthe-et-Moselle, où les voleurs ont pu **démarrer une voiture sans jamais accéder à la clé physique** ([TF1 Info, 2024](#)).

Ainsi, si les deux attaques portent le même nom, elles concernent **des contextes techniques totalement différents** : l'une cible les **interfaces HID sans fil** dans le domaine informatique, l'autre concerne **les systèmes électroniques embarqués dans l'automobile**. Toutes deux illustrent néanmoins les risques croissants liés à la généralisation des communications sans fil et des protocoles mal sécurisés.

3.2.3 Clonage de badge NFC et RFID (fixed code)

La copie d'un badge d'accès sans contact, comme ceux utilisés dans les systèmes **VIGIK** (fréquents dans les immeubles pour ouvrir les portes d'entrée), repose souvent sur l'utilisation d'un **code fixe** transmis à chaque lecture. De nombreux badges RFID de type **13,56 MHz** ou **125 kHz** ne possèdent aucun chiffrement ni mécanisme d'authentification évolutif, ce qui les rend vulnérables à une simple **attaque par clonage**. Avec un **Flipper Zero**, il suffit d'approcher le badge original et d'utiliser la fonction « Lire » pour capturer son identifiant unique. Une fois ce code enregistré, deux méthodes sont possibles pour le réutiliser : soit **émuler le badge directement depuis le Flipper**, soit le **copier sur un badge vierge** de type **magic tag** (carte ou porte-clé reprogrammable).

Les magic tags permettent d'écrire l'identifiant capturé sur un support physique, créant ainsi une **copie matérielle indiscernable de l'original**. Cette opération se fait via la fonction « Écrire » du Flipper, après avoir sélectionné un badge vierge compatible (souvent basé sur le chipset T5577 pour le 125 kHz ou des cartes chinoises réinscriptibles pour le 13,56 MHz). Le clonage est alors complet et le badge copié fonctionne exactement comme l'original sur les lecteurs d'accès. Cette technique montre les limites de sécurité des systèmes qui reposent encore sur des identifiants statiques non protégés, sans chiffrement ni rolling code.



Figure 16 : Un flipper zero qui lit le badge vigik, et une magic card pour copier le badge dessus

3.2.4 Copie de télécommande voiture (rolling code) .

Voici une attaque de type **RollJam**, qui permet de contourner la sécurité des systèmes à rolling code. L'attaquant utilise un **jammer** pour bloquer les signaux envoyés par la télécommande, empêchant ainsi leur réception par le récepteur (ex. portail ou voiture). Pendant ce blocage, le dispositif (ex. Flipper Zero) enregistre les deux premiers codes générés par la télécommande : ici #314 et #628.

Lorsque le brouillage s'arrête, l'attaquant peut **rejouer le premier code #314**, qui sera accepté car le récepteur l'attend encore (il ne l'a jamais reçu). Le code #628, enregistré ensuite, devient alors un **code valide non encore utilisé**, que l'attaquant pourra exploiter plus tard pour ouvrir le système une seconde fois, **sans déclencher d'alerte**. Cela contourne le principe de sécurité du rolling code en désynchronisant volontairement l'émetteur et le récepteur.

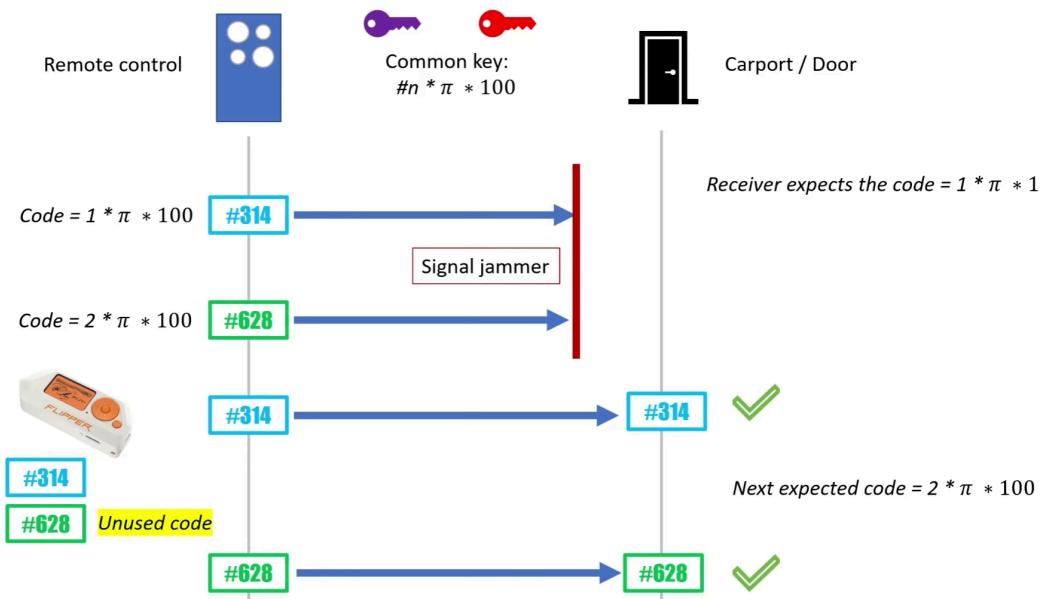


Figure 17 : Copie d'une clé protégé par le rolling code

3.2.5 Brouillage radio divers

Le **brouillage radio** (ou jamming) désigne toute action — intentionnelle ou accidentelle — visant à perturber ou bloquer les communications sans fil, en saturant une bande de fréquence avec des signaux parasites. Il peut s'agir d'une attaque directe contre des

dispositifs comme les télécommandes RF, les connexions Wi-Fi, ou les transmissions Bluetooth, mais il peut aussi être utilisé à des fins de protection.

Dans le domaine de la cybersécurité offensive, un brouilleur est souvent utilisé pour **interrompre un échange** entre un émetteur et un récepteur. Par exemple, lors d'une **attaque RollJam**, l'attaquant brouille les transmissions d'un rolling code pour empêcher le récepteur de les recevoir, tout en les enregistrant. Il peut ensuite les rejouer plus tard pour obtenir un accès non autorisé.

Cependant, le brouillage n'est pas toujours malveillant. Dans **le domaine militaire**, il joue même un **rôle de protection critique**. Les véhicules blindés de la gamme **Scorpion**, développés pour l'armée française (programme SCORPION), intègrent des systèmes de brouillage sophistiqués pour protéger les convois contre les menaces **IED (engins explosifs improvisés)**. Ces bombes artisanales peuvent être déclenchées à distance par des signaux radio — comme ceux d'un téléphone mobile. Pour contrer cela, certains brouilleurs embarqués, conçus par des industriels comme **Eviden (filiale d'Atos)**, utilisent un mécanisme **de détection d'ondes suspectes**, suivi d'une **contre-émission immédiate** pour neutraliser le signal avant qu'il ne puisse atteindre son objectif. Cette technologie protège efficacement les troupes contre les attaques à déclenchement sans fil et contribue à la **cybersécurité physique des systèmes embarqués militaires**.



Figure 18 : Véhicule du programme scorpion protégé des attaques par ondes

Enfin, il existe aussi des cas de **brouillage involontaire**, notamment causés par des **appareils électroménagers** ou professionnels. Des **fours industriels à micro-ondes**, souvent présents dans les hôtels ou les hôpitaux, peuvent émettre des interférences

sur la bande des 2,4 GHz, la même que celle utilisée par le Wi-Fi et le Bluetooth. Ces émissions non intentionnelles peuvent désactiver temporairement des dispositifs connectés ou dégrader leur performance, posant des problèmes dans des environnements sensibles.

Il est **légal d'acheter ou de posséder un brouilleur radio** en France, notamment à des fins pédagogiques, de collection ou de recherche. En revanche, **son utilisation est strictement interdite** par la loi, car elle peut perturber des communications essentielles (Wi-Fi, GPS, téléphonie, médical etc.). L'activation d'un tel dispositif, même à titre privé, constitue une infraction passible de **sanctions pénales**, selon l'article L.39-1 du Code des postes et des communications électroniques.

En somme, le brouillage radio, qu'il soit **défensif, offensif ou accidentel**, reste un enjeu central de la cybersécurité des systèmes sans fil. Une bonne conception doit tenir compte de ces interférences, en ajoutant des protections physiques (blindage), logicielles (authentification, redondance), ou même tactiques (changements de fréquence, audits réguliers du spectre).



figure 19 : brouilleur de drone à 1578.99€ exemple de brouilleur d'onde militaire disponible à l'achat

3.3 Défenses contre les attaques radio

3.3.1 Protocoles wifi

La protection d'un réseau Wi-Fi nécessite plusieurs mesures combinées pour assurer à la fois la confidentialité, l'intégrité et la résilience face aux attaques physiques et logiques. Voici les principales méthodes recommandées :

Activer 802.11w (Protected Management Frames)

La norme 802.11w protège les trames de gestion en les authentifiant et en les chiffrant. Cela rend les attaques de désauthentification (Deauth Attack) beaucoup plus difficiles. Il est fortement conseillé d'activer cette protection sur les routeurs ou points d'accès compatibles.

Utiliser des mots de passe longs et complexes

Un mot de passe Wi-Fi solide doit comporter au moins 16 caractères, mélangeant lettres, chiffres et symboles. Cela réduit considérablement l'efficacité des attaques par dictionnaire sur les captures de handshake.

Mettre en place un filtrage d'adresse MAC

Limiter l'accès au réseau aux seuls appareils autorisés via leur adresse MAC ajoute une protection supplémentaire, bien que ce filtrage puisse être contourné par usurpation.

Utiliser WPA3 dès que possible

WPA3 renforce la sécurité par l'authentification simultanée des égaux (SAE) et la protection obligatoire des trames de gestion. Il rend les attaques sur handshake beaucoup plus difficiles.

Recourir à un VPN sur les réseaux Wi-Fi

Un VPN chiffre toutes les communications réseau, même sur un Wi-Fi sécurisé, et protège contre l'interception de trafic ou les attaques de type Evil Twin.

Utiliser un serveur Radius pour l'authentification

En entreprise, Radius permet d'authentifier chaque utilisateur individuellement, supprimant l'usage d'une clé partagée. Cela limite les risques en cas de compromission d'un compte et renforce globalement la sécurité du réseau.

En appliquant conjointement ces mesures, il est possible de renforcer efficacement la sécurité des communications Wi-Fi contre la majorité des attaques connues.

3.3.2 Détection d'anomalies radio

La détection d'anomalies radio consiste à surveiller l'environnement électromagnétique afin d'identifier des perturbations inhabituelles, des tentatives de brouillage, ou des signaux suspects pouvant indiquer une attaque ou un dysfonctionnement. Dans les systèmes embarqués et les réseaux sans fil (Wi-Fi, Bluetooth, RF propriétaire), cette surveillance est un moyen de défense crucial contre les attaques par **jamming, replay, ou usurpation de signal**.

Des outils spécialisés appelés **analyseurs de spectre** ou **sniffers radio** permettent d'observer l'activité sur une ou plusieurs bandes de fréquence, et de repérer des pics anormaux, des signaux continus non identifiés, ou des collisions suspectes. Ces anomalies peuvent indiquer la présence d'un brouilleur ou d'un dispositif de surveillance passif (comme un enregistreur de trames RF).

Dans les contextes critiques (militaire, industriel, IoT sécurisé), ces systèmes sont parfois intégrés à une **solution de monitoring automatisée**, capable de déclencher une alerte dès qu'une activité radio anormale est détectée dans une zone protégée. Par exemple, un excès de trames Wi-Fi deauth ou de requêtes Bluetooth peut être un indicateur d'attaque active.

Par ailleurs, dans le secteur civil, **les opérateurs télécoms** comme **Bouygues, Orange ou Free** disposent de systèmes capables d'**analyser la qualité du signal sur leur réseau mobile**. Lorsqu'un utilisateur signale une mauvaise connexion persistante, ces opérateurs peuvent **croiser les données techniques du réseau avec des cartes de couverture** pour détecter une éventuelle **zone affectée par un brouillage**. Ces informations peuvent ensuite être transmises à l'**ANFR** (Agence Nationale des Fréquences), qui est compétente pour enquêter et intervenir.

Ainsi, la détection proactive des anomalies radio est une composante essentielle de la sécurité sans fil, permettant non seulement d'identifier les attaques, mais aussi d'optimiser la résilience des systèmes face aux interférences, qu'elles soient malveillantes ou accidentnelles.

Voici un exemple de cartoradio de l'anfr qui en autre permet de reperer les zones de brouillages

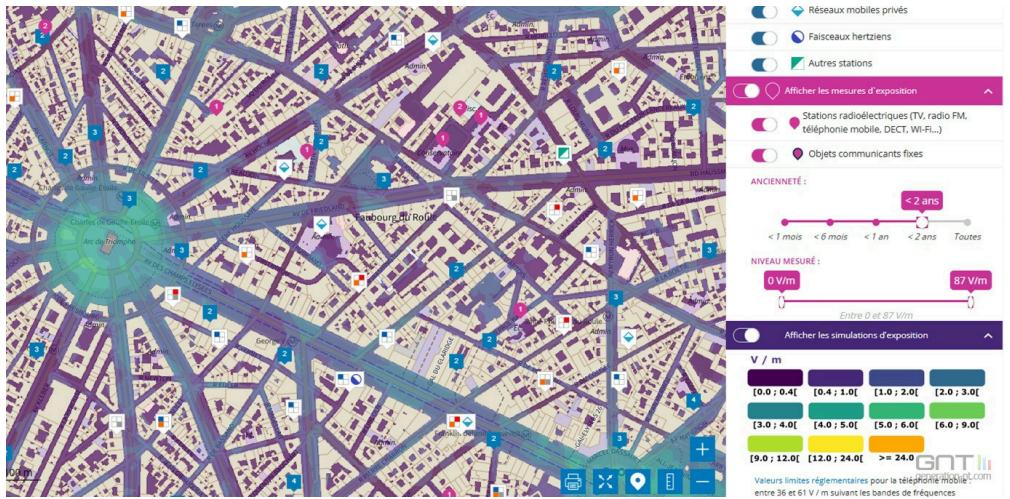


Figure 20 : ANFR Cartoradio exposition radiofréquences mobiles

3.3.3 Blindage physique (portefeuille, couche métallique, etc.)

Le **blindage physique** consiste à limiter ou bloquer les émissions et réceptions d'ondes électromagnétiques dans un périmètre donné, afin de prévenir l'interception, le brouillage ou l'exfiltration d'informations sensibles. Cette approche est essentielle dans les systèmes sans fil (RF, NFC, Wi-Fi, etc.), notamment dans les environnements embarqués ou critiques.

Dans le quotidien, on retrouve ce principe dans les **portefeuilles anti-RFID**. Ils intègrent une fine couche métallique (souvent en aluminium ou maille de cuivre) qui agit comme une **mini cage de Faraday**, empêchant la lecture involontaire ou malveillante de badges NFC, cartes bancaires ou titres de transport. Ce type de protection est devenu courant face à la banalisation des lecteurs RFID portables.



Figure 21 : portefeuilles anti-RFID

Au-delà de ces usages personnels, le blindage est également appliqué dans des environnements sensibles via l'utilisation de **couches métalliques**, de peintures

conductrices, ou encore de **structures en béton armé**. Ces matériaux peuvent bloquer tout ou partie du spectre radiofréquence, et sont parfois utilisés dans la conception de salles sécurisées, de centres de données, les centrales nucléaires ou des laboratoires de test RF.

Un exemple concret de cette logique à grande échelle concerne **les centres de données d'opérateurs télécoms ou de fournisseurs d'hébergement**, comme Free en France. Certains de leurs **serveurs stratégiques sont enterrés profondément dans des bunkers**, d'anciens forts militaires ou dans des **galeries souterraines protégées**, afin d'offrir :

- Une **protection physique** contre les intrusions et les catastrophes naturelles ;
- Une isolation thermique et électromagnétique naturelle (par la roche ou le béton) ;
- Une **résilience accrue face aux attaques physiques ou RF**.

Ces installations, parfois situées dans d'anciennes carrières ou sites désaffectés militaires, bénéficient en plus de **l'absence d'exposition aux ondes extérieures**, rendant les communications ou interférences non sollicitées pratiquement impossibles.

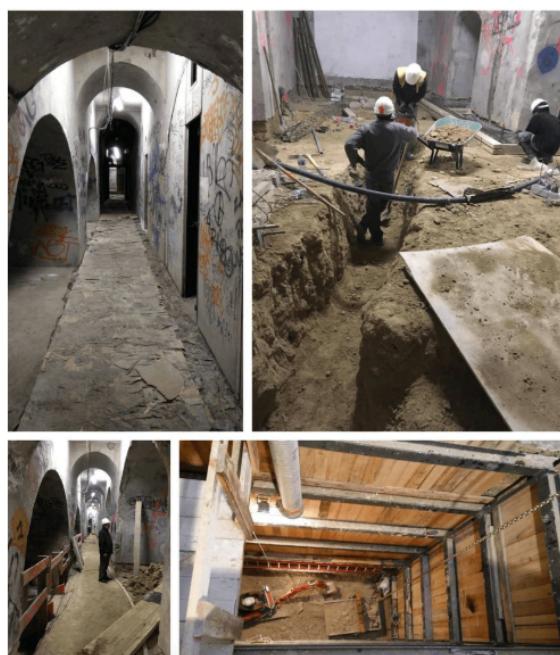


Figure 22 : bunker transformer en data center

4 Études de cas

4.1 BluAttack : analyse de l'attaque et effets sur iPhone

L'attaque surnommée **BluAttack** désigne un **dénie de service (DoS)** ciblant les iPhone via leur interface **Bluetooth Low Energy (BLE)**. Elle est rendue possible grâce à l'utilisation du **Flipper Zero** équipé d'un firmware modifié (comme le firmware "[Xtreme](#)"), qui permet d'exploiter un comportement spécifique de certains appareils Apple tournant sous **iOS 17.0.3** ou des versions antérieures.

Le principe de l'attaque repose sur une **inondation de requêtes Bluetooth**, envoyées en boucle depuis le Flipper vers l'iPhone cible. Le Flipper utilise l'application "Apple BLE Spam", qui envoie un grand nombre de paquets BLE (notamment des demandes de couplage, ou de service advertisement) en très peu de temps. Ces requêtes sont diffusées dans l'environnement à l'aide de la radio 2,4 GHz du Flipper, accessible depuis le menu Sub-GHz ou Bluetooth.

Lorsque l'iPhone est à proximité et que son Bluetooth est activé, il tente de **gérer et interpréter chaque requête BLE entrante**. Toutefois, l'attaque consiste à **saturer cette pile logicielle BLE**. Après quelques dizaines de secondes ou quelques minutes de réception continue, le système **devient instable**, puis complètement **verrouillé** : l'écran fige, les touches ne répondent plus, et l'appareil doit être redémarré de force (**hard reset**).

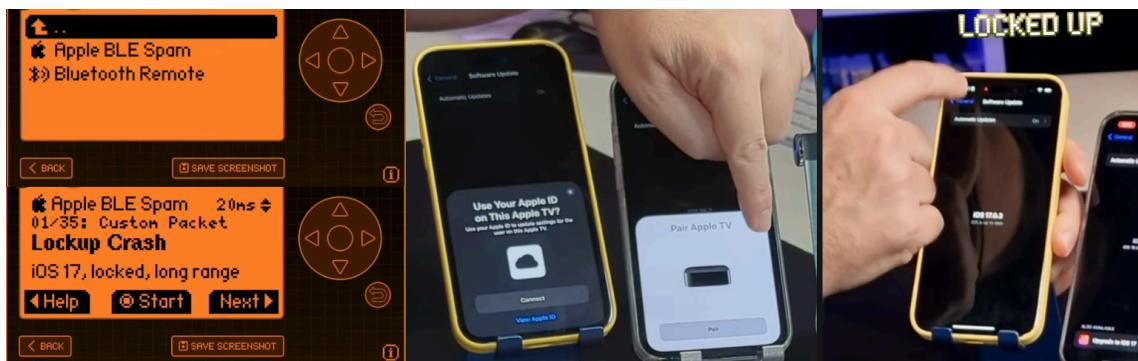


Figure 23 : Ble Spam Iphone crash Attack sur iOS 17.0.2

Ce comportement n'apparaît que si le Bluetooth de l'iPhone est activé. Dès que celui-ci est désactivé (de manière complète), l'appareil devient insensible à l'attaque. Cela prouve que la vulnérabilité provient directement du **traitement logiciel des paquets BLE** et non d'un exploit plus bas niveau.

L'attaque BluAttack ne permet pas une prise de contrôle ou un accès aux données, mais démontre l'impact qu'une **surcharge radio intentionnelle** peut avoir sur la

disponibilité d'un système mobile. Elle illustre également la nécessité, pour les constructeurs, de **tester leurs piles de protocoles contre des cas d'usage extrêmes ou malveillants**.

À noter que cette vulnérabilité a été remontée à Apple, et semble **corrigée à partir d'iOS 17.2**, ce qui renforce l'idée que l'écosystème Apple, bien que fermé, n'est pas exempt de failles côté communication sans fil.

4.2 Clonage NFC : méthode et validation

Le clonage NFC est une attaque physique simple à mettre en œuvre lorsqu'un **badge**, **une carte ou un dispositif sans contact** repose sur un protocole faible ou sans chiffrement. Grâce à des outils accessibles comme le **Flipper Zero**, il est aujourd'hui possible de copier certains badges NFC ou RFID en quelques secondes.

Le **Flipper Zero** intègre un lecteur/émetteur NFC capable de lire, stocker et réémettre des identifiants de carte ou de badge. Il peut également écrire sur certains supports vierges compatibles (comme les **Magic Tags**). Ce type d'attaque est très répandu sur les systèmes **d'accès sans contact peu sécurisés**, comme certains **badges Vigik**, des cartes d'entrée de résidences ou de parkings, utilisant des protocoles anciens (ISO 14443-A, MIFARE Classic, etc.).

Exemple de procédure :

Dans l'expérimentation réalisée :

1. **Lecture du badge original** : en approchant un badge NFC contre le dos du Flipper Zero, celui-ci lit son identifiant (UID) ainsi que, si possible, d'autres blocs de mémoire. Cette opération ne nécessite aucun accès au système d'origine — seule la proximité physique du badge est requise.



Figure 24 : module de lecture de badge nfc du flipper zero



Figure 25 : Copie d'un badge nfc

- Clonage sur un tag NFC vierge : en utilisant un **Magic NFC Tag** (carte réinscriptible), le Flipper écrit les mêmes informations récupérées sur le tag vierge, créant ainsi une **copie fonctionnelle** du badge original. Le tag cloné peut alors être utilisé à la place du vrai, par exemple pour ouvrir une porte sécurisée ou activer un lecteur de contrôle d'accès.



Figure 26 : nfc magic permettant de faire des copies de badge nfc

Outre le Flipper Zero, **d'autres outils existent**, notamment des applications mobiles (sous Android avec NFC), ou des lecteurs spécialisés comme le Proxmark3, qui permettent de cloner des cartes plus avancées avec un accès bas niveau.

Ce type de vulnérabilité soulève aussi des inquiétudes autour de la **lecture sans contact de cartes bancaires**. Bien que les cartes modernes soient protégées et la double authentification soit désormais systématique pour les achats en ligne, il reste possible d'**intercepter un numéro de carte** à courte distance (souvent limité à quelques centimètres) dans des cas non protégés. Pour les **petits achats sans code PIN**, une fraude est encore envisageable, surtout si les protections côté banque ne sont pas strictement configurées.

Le clonage NFC, bien qu'en apparence anodin, révèle l'importance de **sécuriser les badges d'accès avec des protocoles modernes et chiffrés**, et de **restreindre physiquement leur exposition** (porte-badge blindé, désactivation hors usage, etc.).

4.3 Rolling Code : vulnérabilité des clés de voiture

source : [Rolling codes explained #flipperzero](#) de la chaîne tech and fun

Les systèmes d'ouverture sans contact utilisés dans les voitures modernes ou portails motorisés reposent souvent sur des **codes tournants**, ou **rolling codes**, conçus pour empêcher les attaques par répétition de signal (replay). Contrairement à un système à **code fixe** (qui envoie toujours le même identifiant radio), les systèmes à rolling code génèrent un **nouveau code unique à chaque appui**.

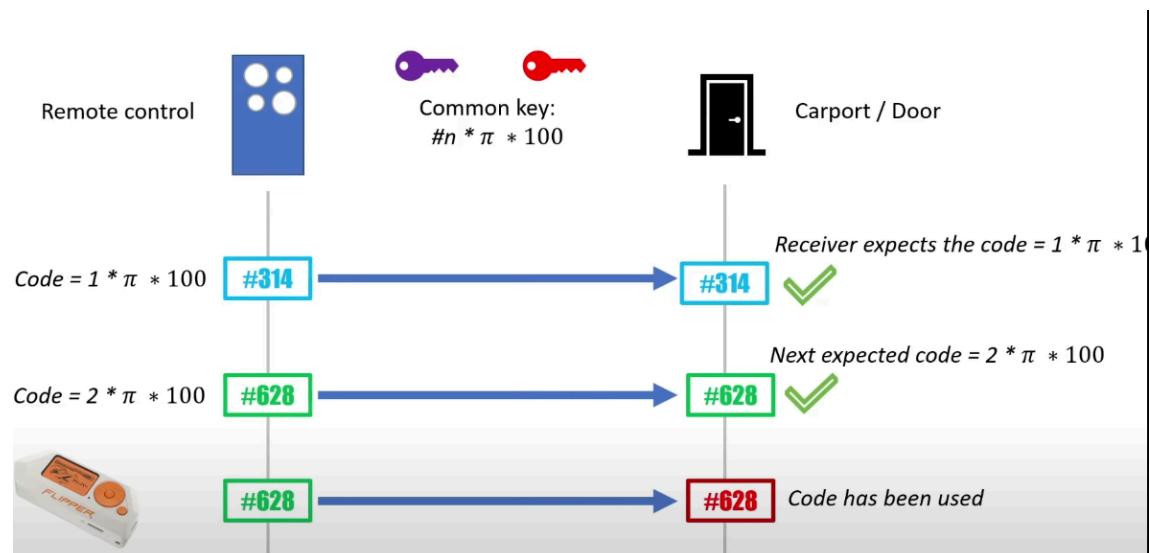


Figure 27 : Fonctionnement normal du rolling code après hacking raté

Dans le premier schéma :

- Lors de la première pression sur la télécommande, le code envoyé est calculé par une formule commune au récepteur et à l'émetteur (par exemple $n \times \pi \times 100$). Ici, $n=1$, ce qui donne 314 . Le récepteur, s'il attend ce code, déverrouille

la porte.

- Lors de la seconde pression, le code suivant ($n=2$, donc **628**) est attendu. Il est également accepté.
- Si un attaquant a enregistré ce code avec un appareil comme le **Flipper Zero**, et tente de le rejouer plus tard, **le récepteur refuse la commande**, car ce code a déjà été utilisé.

Ce système empêche donc qu'un code intercepté soit réutilisé plus tard. C'est une protection simple mais efficace contre les attaques de type replay.

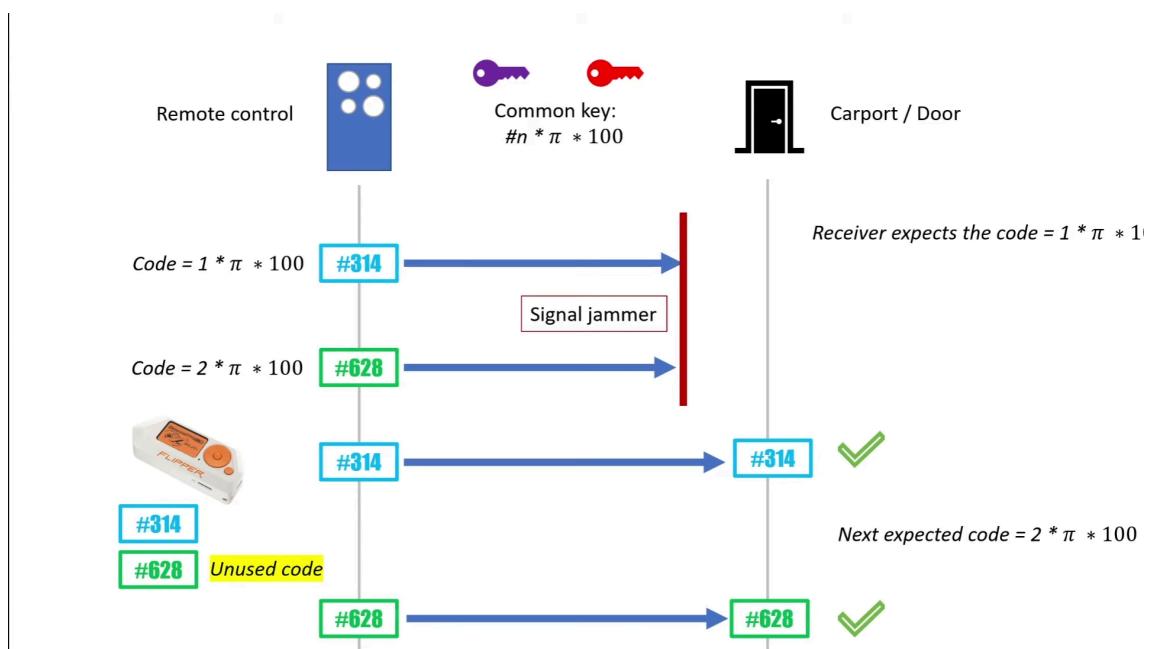


Figure 28 : Exploitation de la faille avec brouillage (hacking réussi)

La seconde image illustre une attaque plus sophistiquée exploitant la faiblesse d'un rolling code : **l'attaque par jamming + enregistrement**.

- L'attaquant brouille le signal radio (avec un **signal jammer**) entre la télécommande et le récepteur.
- Lorsque l'utilisateur légitime appuie une première fois, le code **314** est émis mais **n'est pas reçu** par la voiture. L'attaquant, lui, l'enregistre.
- L'utilisateur réessaie (deuxième appui), un **nouveau code 628** est envoyé, aussi bloqué par le brouillage, mais également enregistré.
- Ensuite, l'attaquant arrête le brouillage et **rejoue le premier code 314**. Le récepteur, ne l'ayant jamais reçu, **l'accepte comme nouveau**.

- Plus tard, l'attaquant possède encore **le code 628 en réserve**, qu'il pourra utiliser à son tour : la sécurité est donc contournée.

Cette méthode montre que même un système basé sur des rolling codes peut être vulnérable à une **attaque de type jamming + capture**, surtout si l'utilisateur ne se rend pas compte que la première commande n'a pas été reçue.

En conclusion, bien que les **rolling codes** soient plus sûrs que les codes fixes, **ils ne sont pas invulnérables**. Le **Flipper Zero**, bien qu'il ne puisse généralement pas casser les systèmes bien conçus, peut être utilisé dans des contextes où le protocole est mal implémenté, ou combiné avec des attaques plus avancées comme le brouillage. Il est donc crucial de vérifier que **le système d'ouverture confirme l'exécution de la commande (visuelle ou sonore)** pour ne pas se faire piéger.

4.4 Deauth Attack sur caméras Wi-Fi

De nombreuses caméras IP utilisées en domotique ou dans des installations de vidéosurveillance professionnelles se connectent au réseau local via une liaison Wi-Fi. Typiquement, la caméra se connecte au routeur domestique ou professionnel, lequel est relié à Internet. Grâce à une application mobile, l'utilisateur peut accéder à la caméra à distance, via la 4G ou tout autre réseau externe.

Description du fonctionnement et de l'attaque

Dans un fonctionnement normal, l'utilisateur peut consulter en temps réel le flux de la caméra depuis son smartphone. Cependant, une vulnérabilité peut être exploitée via une attaque dite de désauthentification (Deauth Attack). Cette attaque perturbe volontairement la connexion Wi-Fi entre la caméra et le routeur, dans le but de provoquer une reconnexion de la caméra et d'en capturer les échanges cryptographiques.

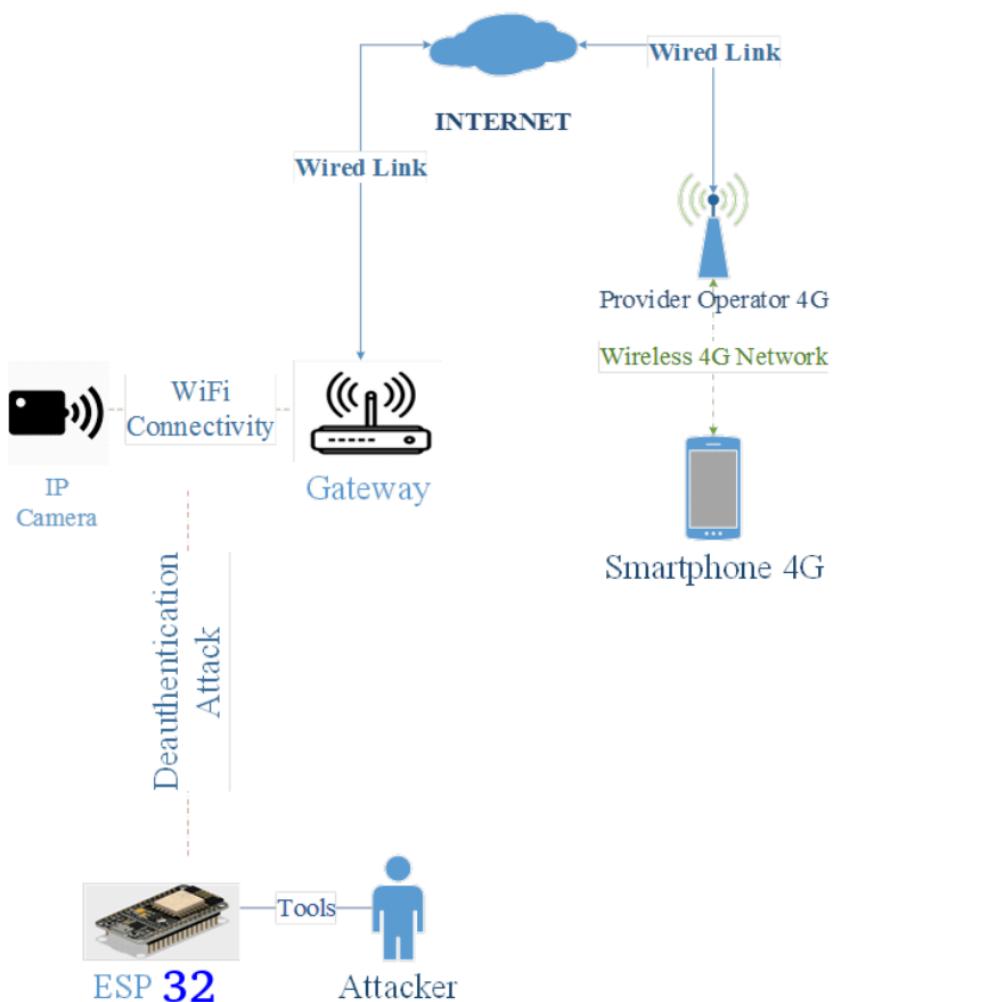


Figure 29 : Schema du deauth attack sur la camera wifi avec le flipper zero

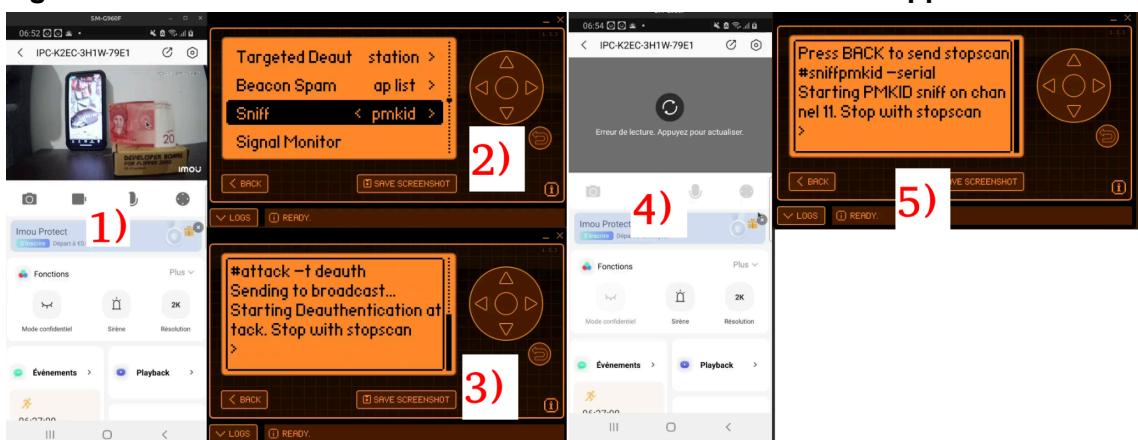


Figure 30 : Deauth attack sur la camera wifi avec le flipper zero

1. Connexion normale

Au départ, la caméra est connectée au Wi-Fi et accessible depuis l'extérieur. Elle échange des données avec le routeur, ce qui permet à l'utilisateur de la

voir en direct.

2. Préparation de l'attaque avec sniffer

L'attaquant met en place un dispositif (comme une antenne en mode monitor ou un Flipper Zero) pour sniffer les paquets. Il attend la diffusion de paquets spécifiques appelés PMKID (Pairwise Master Key Identifier). Ces paquets, transmis au début d'une session WPA/WPA2, peuvent parfois être interceptés sans nécessiter la capture complète du 4-Way Handshake.

3. Lancement de la désauthentification

L'attaquant émet des paquets de type deauth, qui simulent un ordre légitime du routeur. Cela force la caméra à se déconnecter du réseau Wi-Fi. Cette étape est critique car elle déclenche une nouvelle authentification.

4. Déconnexion visible de la caméra

Du point de vue de l'utilisateur, la caméra apparaît soudainement hors ligne. Elle n'est plus accessible via l'application, ce qui indique que la connexion a été interrompue.

5. Capture du 4-Way Handshake

Lorsque la caméra tente de se reconnecter, elle initie un nouveau processus d'authentification WPA/WPA2. Ce processus, appelé 4-Way Handshake, permet de vérifier mutuellement les identifiants entre la caméra et le routeur. Si ce moment est capturé, les paquets EAPOL obtenus peuvent ensuite être utilisés pour effectuer un brute-force du mot de passe Wi-Fi.

Exploitation des données capturées

Une fois les paquets EAPOL extraits sous forme de fichier .pcap l'attaquant peut les convertir au format compatible avec des outils comme Hashcat (lien de l'outil de conversion ici <https://hashcat.net/cap2hashcat/index.pl>). Il devient alors possible de tester une large liste de mots de passe (dictionnaire) afin d'identifier celui utilisé par le réseau. Si le mot de passe est faible ou courant, la probabilité de succès est élevée.

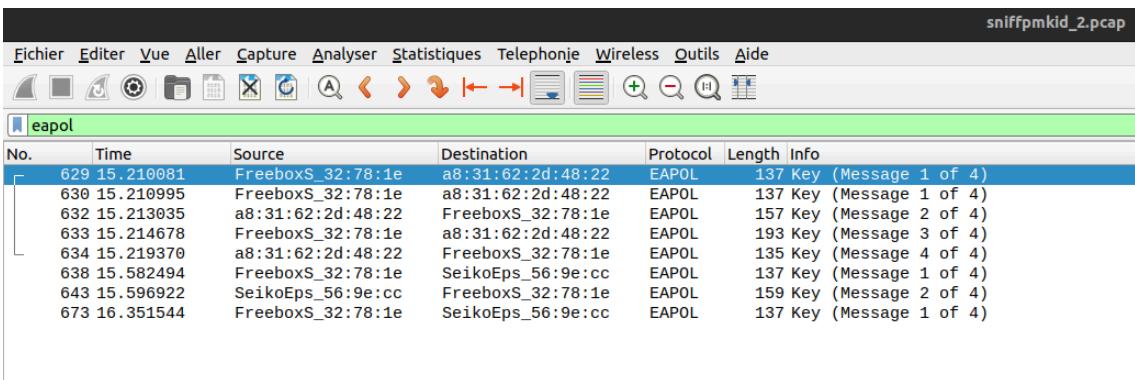


Figure 31 : Exemple de lecture de paquet EAPOL sur wireshark

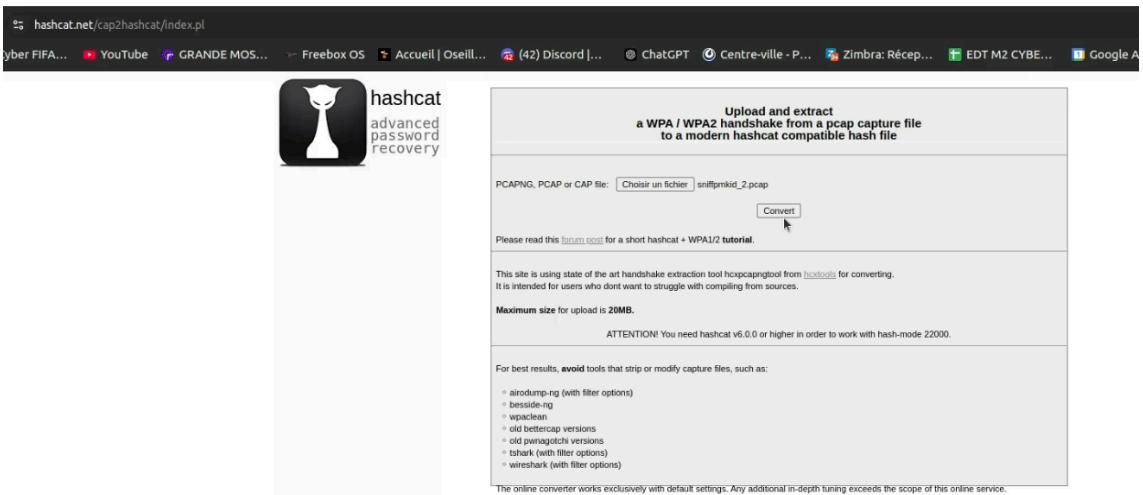
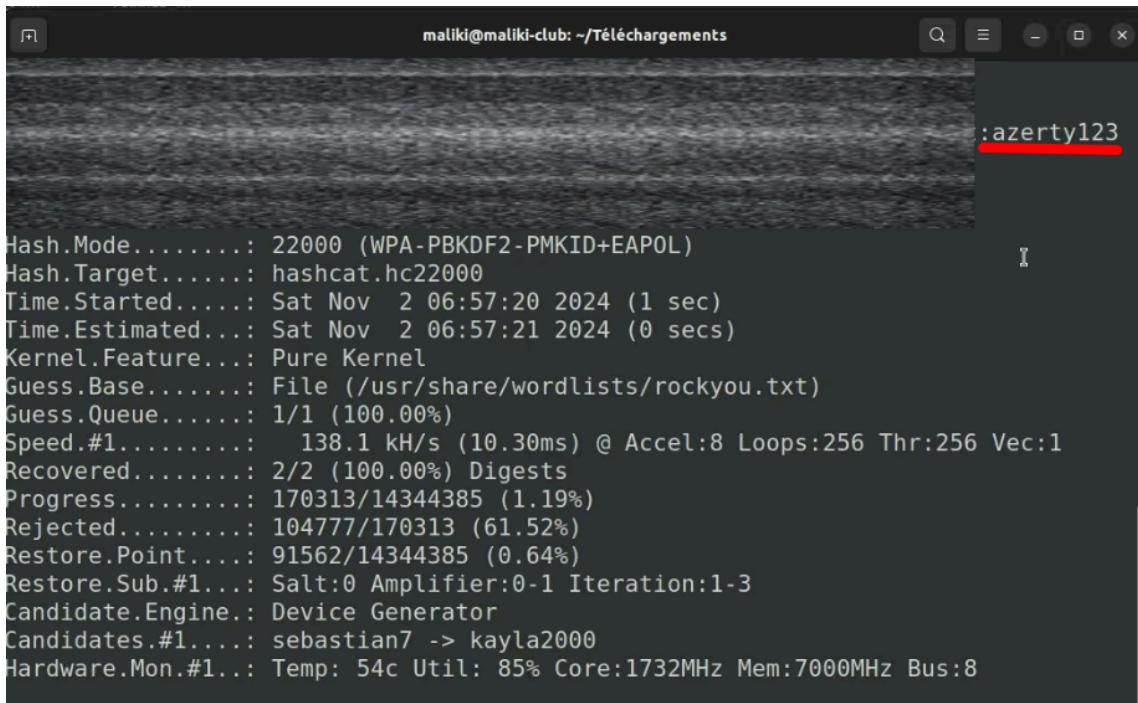


Figure 31 bis : Site de conversion de paquet EAPOL pour le brut force avec hashcat

Dans l'exemple observé, un mot de passe simple comme "azerty123" a pu être retrouvé très rapidement à l'aide d'un dictionnaire de mots courants. Cela démontre que, même si les protocoles comme WPA2 sont solides, leur sécurité dépend fortement du choix du mot de passe.



A screenshot of a terminal window titled "maliki@maliki-club: ~/Téléchargements". The window displays the output of a hashcat command. The password being cracked is "azerty123", which is highlighted with a red box. The terminal shows various parameters and progress of the attack, including hash mode (WPA-PBKDF2-PMKID+EAPOL), target hash (hashcat.hc22000), start time (Sat Nov 2 06:57:20 2024), estimated end time (0 secs), kernel feature (Pure Kernel), guess base (rockyou.txt), queue length (1/1), speed (138.1 kH/s), recovered digest count (2/2), progress (170313/14344385, 1.19%), rejected count (104777/170313, 61.52%), restore point (91562/14344385, 0.64%), restore sub-point (Salt:0 Amplifier:0-1 Iteration:1-3), candidate engine (Device Generator), and hardware monitoring (Temp: 54c Util: 85% Core:1732MHz Mem:7000MHz Bus:8).

```
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target....: hashcat.hc22000
Time.Started...: Sat Nov 2 06:57:20 2024 (1 sec)
Time.Estimated.: Sat Nov 2 06:57:21 2024 (0 secs)
Kernel.Feature.: Pure Kernel
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue....: 1/1 (100.00%)
Speed.#1.....: 138.1 kH/s (10.30ms) @ Accel:8 Loops:256 Thr:256 Vec:1
Recovered.....: 2/2 (100.00%) Digests
Progress.....: 170313/14344385 (1.19%)
Rejected.....: 104777/170313 (61.52%)
Restore.Point.: 91562/14344385 (0.64%)
Restore.Sub.#1.: Salt:0 Amplifier:0-1 Iteration:1-3
Candidate.Engine.: Device Generator
Candidates.#1...: sebastian7 -> kayla2000
Hardware.Mon.#1.: Temp: 54c Util: 85% Core:1732MHz Mem:7000MHz Bus:8
```

Figure 32 : Brute force mot de passe wifi avec hashcat

4.5 Extraction de Red Key moto via EEPROM

Dans certains modèles de motos, le système d'antidémarrage repose sur une clé maîtresse appelée **Red Key**. Cette clé est essentielle pour programmer de nouvelles clés ou pour réinitialiser l'ECU (Electronic Control Unit). Si cette Red Key est perdue, elle n'est généralement pas rééditée par le constructeur, ce qui peut entraîner des frais élevés, voire le remplacement complet du boîtier électronique.

Une technique permet toutefois de retrouver cette clé via une attaque physique ciblée sur la mémoire de l'ECU. La procédure repose sur la lecture directe d'une **EEPROM** présente dans le circuit électronique de la moto. Pour notre scooter, il s'agit le plus souvent d'une mémoire de type **24C02 ou 24C04**.

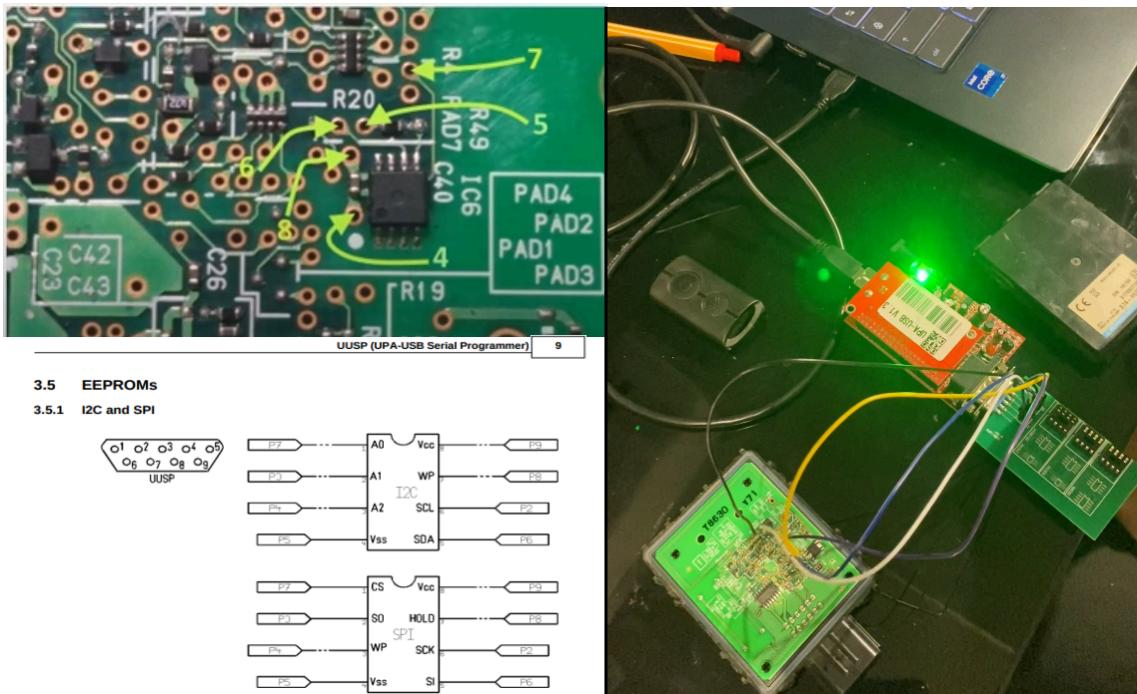


Figure 33 : Lecture d'une EEPROM d'ECU moto via interface I2C à l'aide d'un programmeur USB

L'opération se déroule en plusieurs étapes :

- On identifie sur le circuit imprimé les broches de l'EEPROM, puis on y soude ou connecte avec soin des fils.
- Ces fils sont ensuite reliés à un **lecteur d'EEPROM** compatible (comme un CH341A ou un TL866).
- Le lecteur est branché à un ordinateur, et à l'aide d'un logiciel, on procède à la lecture complète de la mémoire.
- Lors de cette lecture, le contenu de la mémoire s'affiche sous forme hexadécimale. Notre clé est chiffré ici.

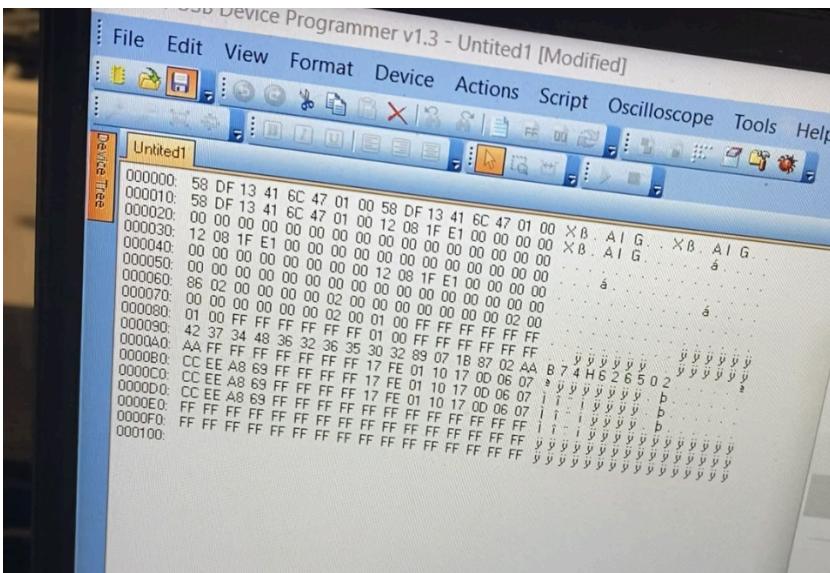


Figure 34 : Hexadecimale de la red key

Le fichier obtenu peut être analysé en utilisant un éditeur hexadécimal. Par un travail de **reverse engineering**, on peut retrouver dans ces lignes de code la **séquence correspondant à la Red Key**. Cette séquence est souvent encodée selon un format propriétaire.

Une fois le code retrouvé :

- Il peut être entré manuellement via une procédure au tableau de bord (si la moto le permet).
- Utilisé pour programmer une **nouvelle clé** compatible, en injectant les bons identifiants dans l'ECU.

Cette attaque démontre que même sans la clé d'origine, il est parfois possible de **reprendre le contrôle d'un véhicule** en accédant physiquement à sa mémoire. Cela souligne l'importance, pour les constructeurs, de chiffrer les données sensibles présentes dans les EEPROM, ou d'utiliser des modules cryptographiques protégés.

5 Conclusion et remerciement

5.1 Conclusion

Ce mémoire a permis de mettre en lumière la complexité et la diversité des menaces pesant sur les systèmes embarqués. Ces dispositifs, bien que conçus pour être autonomes, discrets et performants, sont loin d'être à l'abri des attaques. Nous avons

vu qu'une large gamme d'attaques peut viser leurs interfaces physiques (EEPROM, ports UART, JTAG), leurs communications radio (Wi-Fi, Bluetooth, NFC), ou leur firmware lui-même, souvent non chiffré ou mal protégé.

Les démonstrations réalisées au fil de ce travail — lecture d'EEPROM d'ECU moto, clonage de badge NFC, désauthentification de caméras IP ou encore analyse de rolling codes — ont montré que la compromission d'un système embarqué ne relève plus de la fiction. À l'inverse, elles soulignent aussi que la défense est possible, à condition d'adopter une approche multi-couches : blindage physique, chiffrement matériel, mise à jour sécurisée, désactivation des interfaces inutiles, ou encore détection d'anomalies radio.

Il est essentiel de retenir que les failles ne résultent pas uniquement d'un mauvais choix technologique, mais souvent d'un compromis assumé : performance versus sécurité, simplicité versus résilience. En ce sens, la sécurité embarquée ne peut être un ajout tardif ou facultatif : elle doit être pensée dès la conception. À une époque où tout est connecté — des voitures aux infrastructures industrielles — négliger la cybersécurité embarquée revient à exposer l'ensemble de l'environnement numérique à des risques systémiques.

Ainsi, ce mémoire se veut un point de départ pour ceux qui souhaitent comprendre, tester et améliorer la sécurité des systèmes embarqués. Dans un monde de plus en plus dépendant de l'IoT, sécuriser ces composants n'est pas une option, c'est une nécessité.

5.2 Remerciements

Je souhaite tout d'abord exprimer ma profonde gratitude envers ma famille et mes amis, dont le soutien indéfectible et les encouragements constants m'ont accompagné tout au long de ces deux années.

Je remercie chaleureusement l'équipe pédagogique de l'Université Paris Cité pour la qualité de leur enseignement et leur accompagnement tout au long de ce master.

Ma reconnaissance va tout particulièrement à mon maître d'apprentissage, Luc Bouganim, pour m'avoir fait confiance en me donnant l'opportunité d'intégrer son équipe, et plus précisément l'équipe de recherche PETRUS. Je tiens à remercier sincèrement Luc, Ludovic, Laurent, Philippe, ainsi que l'ensemble de l'équipe PETRUS pour leur accueil chaleureux, leur bienveillance, leur patience et leur pédagogie. Leur accompagnement tout au long de cette aventure a été précieux, et je suis profondément reconnaissant d'avoir été recruté dans cette équipe. Travailler à leurs côtés m'a permis de développer mes compétences dans un environnement professionnel aussi stimulant qu'humainement enrichissant.

Je remercie également Patrice Martin, mon tuteur enseignant, pour sa disponibilité, ses conseils avisés et son soutien constant durant la rédaction de ce mémoire.

Enfin, à toutes les personnes rencontrées au cours de ce parcours, qui ont contribué à faire de ces deux années une expérience aussi formatrice qu'épanouissante, je souhaite adresser mes plus sincères remerciements.

6 Figures

- Figure 1 : Accès aux broches UART sur un routeur 4G LTE
- Fligure 2 : Connexion UART à l'aide de picocom sur interface /dev/ttyUSB0
- Fligure 3 : Lecteur d'eprom sur un site marchant connu
- Figure 4 : Gèle de la ram avec spray pour conserver ses données
- Figure 5 : Exécution de l'outil bitunlocker.sh sur une image mémoire
- Figure 6 : USB rubber Ducky
- Figure 7 : OMG Cable
- Figure 8 : USB KILLER
- Fligure 9 : Schema du chiffrement d'un message avec WEP
- Figure 10 : Schéma du 4-Way Handshake WPA2 (protocole EAPOL)
- Fligure 11 : Exemple de lecture de paquet EAPOL sur wireshark
- Figure 12 : Diagramme utilisation protocole wifi (wpa , 2 , 3 et wep)
- Figure 13 : Badge vigik , carte nigo et carte mifare
- Figure 14: Fonctionnement du Rolling Code pour la sécurisation des télécommandes sans fil
- Figure 15: Attaque de désauthentification (Deauth Attack)
- Figure 15 bis : flipper zero avec le firmware "XTREME"
- Figure 16 : Un flipper zero qui lit le badge vigik, et une magic card pour copier le badge dessus
- Fligure 17 : Copie d'une clé protégé par le rolling code
- Figure 18 : Vehicule du programme scorpion protégé des attaques par ondes
- figure 19 : brouilleur de drone à 1578.99€ exemple de brouilleur d'onde militaire disponible à l'achat
- Figure 20 : ANFR Cartoradio exposition radiofréquences mobiles
- Figure 21 : portefeuilles anti-RFID
- Figure 22 : bunker transformer en data center
- Figure 23 : Ble Spam Iphone crash Attack sur IOS 17.0.2
- Figure 24 : module de lecture de badge nfc du flipper zero
- Figure 25 : Copie d'un badge nfc
- Figure 26 : nfc magic permettant de faire des copies de badge nfc
- Figure 27 : Fonctionnement normal du rolling code après hacking raté
- Figure 28 : Exploitation de la faille avec brouillage (hacking réussi)
- Figure 29 : Schema du deauth attack sur la camera wifi avec le flipper zero
- Figure 30 : Deauth attack sur la camera wifi avec le flipper zero

Figure 31 : Exemple de lecture de paquet EAPOL sur wireshark
Figure 31 bis : Site de conversion de paquet EAPOL pour le brut force avec hashcat
Figure 32 : Brute force mot de passe wifi avec hashcat
Figure 33 : Lecture d'une EEPROM d'ECU moto via interface I2C à l'aide d'un programmeur USB
Figure 34 : Hexadecimal de la red key

A Reference / Source

Figure 1 , 2 : [Accessing UART Console on 4G LTE Router - Hacking the Mercusys MR34L](#)

Figure 3 : [Lien du lecteur d'eeprom](#)

Figure 4 , 5 : [Cold Boot Attacks on Encryption Keys](#)

Figure 7 :<https://lab401.com/fr/products/o-mg-cable-programmer-usb-a>

Figure 6 :

<https://www.amazon.fr/HAK5-Canard-caoutchouc-Nouvelle-version/dp/B0C1HBSQS2>

Figure 8: <https://news.sophos.com/fr-fr/2017/03/27/le-usb-killer-dans-mauvaises-mains/>

Figure 9 :

<https://www.thesecuritybuddy.com/wireless-network-and-security/iv-attack-in-wep/>

Figure 10 : <https://networklessons.com/wireless/wpa-and-wpa2-4-way-handshake>

Figure 12: <https://wigle.net/stats>

Figure 13: [badge](#) / [navigo](#) / [carte](#)

Figure 18 : <https://www.defense.gouv.fr/dga/programme-scorpion>

Figure 19 : <https://www.skylifr.com/brouilleur-militaire.html>

Figure 20 :

<https://www.generation-nt.com/actualites/anfr-cartoradio-exposition-onde-telephonie-mobile-2057037>

Figure 21: [Porte feuille anti rfid](#)

figure 22 Bunker :

<https://www.leparisien.fr/paris-75/paris-le-centre-de-donnees-de-l-ancien-bunker-chauffe-150-logements-08-03-2019-8027646.php> et la
<https://www.scaleway.com/en/blog/meet-fr-par-3/>

Figure 23 : <https://www.youtube.com/shorts/Qjp3QRg1Xes>

Figure 26 : <https://lab401.com/fr/products/ultimate-magic-card-gen4>

<https://cyberphinix.de/blog/embedded-security-basics/>

<http://recode.fr/tablettes-industrielles/getac/f110>

<https://www.fortinet.com/fr/resources/cyberglossary/scada-and-scada-systems>

<https://github.com/Flipper-XFW/Xtreme-Firmware>

Figure 23 : <https://www.youtube.com/shorts/eoYNZBrlFak>

<https://www.it-connect.fr/mousejacking-piratage-de-claviers-et-souris-sans-fil/>

<https://www.tf1info.fr/justice-faits-divers/video-automobile-qu-est-ce-que-le-mouse-jacking-cette-nouvelle-facon-de-voler-une-voiture-2341859.html>

Figure 14 , 17 , 27  Rolling codes explained #flipperzero

Figure 31 bis : <https://hashcat.net/cap2hashcat/index.pl>