TER SYSTEMES INDUSTRIELS

M1 FSI 2024/2025

ATTAQUE PAR SCRIPT

1 Propriétés du document

CLASSIFICATION DU DOCUMENT	Protégée
REFERENCE DU DOCUMENT	ATT_SCRIPT_24_25
DATE D'EMISSION DU DOCUMENT	20/06/2025
VERSION DU DOCUMENT	1
AUTEURS	DIA Mouhamadou Afiss
PROPRIETAIRE DU DOCUMENT	AGOPIAN Roland – Aix Marseille

Tableau 1 – Propriété du document

2 Historique des révisions

VERSION	DATE	AUTEUR	RESUME DES MODIFICATIONS
1	20/06/2025	DIA Mouhamadou Afiss	Version n°1

Tableau 2 - Historique des révisions

3 Distribution

NOM	FONCTION
AGOPIAN Roland	Professeur Encadrant Pédagogique
PARNET Cyril	Chef de Projet
BERREBIHA Nasserdine	Adjoint au chef de projet
YABDA Redouane	Membre de l'équipe
DIA Mouhamadou Afiss	Membre de l'équipe
DOUZI Youssef	Membre de l'équipe
KASMI Badreddine	Membre de l'équipe

Tableau 3 - Distribution

4 Approbation

NOM	FONCTION	SIGNATURE	DATE
PARNET Cyril	Chef de Projet	CP	19/06/2025
BERREBIHA Nasserdine	Adjoint au chef de projet	NB	19/06/2025
YABDA Redouane	Membre de l'équipe	RY	19/06/2025
DIA Mouhamadou Afiss	Membre de l'équipe	MaD	19/06/2025
DOUZI Youssef	Membre de l'équipe	YD	19/06/2025
KASMI Badreddine	Membre de l'équipe	KB	19/06/2025

Tableau 4 – Approbation

Sommaire

5 Présentation du document

Ce document présente une analyse des attaques sur systèmes industriels via les simulateurs CODESYS et FactoryIO, avec un focus sur les attaques par script, qui consiste à envoyer des commandes non autorisées au PLC, via des requêtes Modbus.

6 Attaques par Script

6.1 Presentation Simulateur

6.1.1 FactoryIO

FactoryIO est un simulateur industriel 3D en temps réel qui permet de modéliser visuellement des systèmes d'automatisation. Il offre une large bibliothèque d'objets (convoyeurs, moteurs, capteurs, bacs, interrupteurs, etc.) permettant de créer des scènes industrielles réalistes.

6.1.2 CODESYS

CODESYS (Controller Development System) est un environnement de développement intégré (IDE) destiné à la programmation d'automates programmables industriels (PLC), conforme à la norme IEC 61131-3.

CODESYS permet de programmer en plusieurs langages :

- Ladder Diagram (LD) : langage graphique en échelle, très utilisé en automatisme.
- Structured Text (ST) : langage textuel structuré, proche du Pascal.
- Function Block Diagram (FBD) : langage graphique par blocs fonctionnels
- Sequential Function Chart (SFC) : pour la logique séquentielle

Il est possible d'exécuter les programmes PLC sur un automate physique ou sur un PLC virtuel installé sur PC via CODESYS Control Win (soft PLC).

CODESYS peut aussi se connecter à des systèmes extérieurs via différents protocoles industriels, dont Modbus TCP, ce qui est crucial pour l'interconnexion avec FactoryIO.

6.1.3 Lien entre FactoryIO et CODESYS

L'interconnexion entre FactoryIO et CODESYS repose principalement sur le protocole Modbus TCP, où :

- FactoryIO agit comme un serveur Modbus : il expose ses capteurs et actionneurs sous forme d'entrées/sorties numériques ou analogiques accessibles via des adresses Modbus.
- CODESYS agit comme un client Modbus : il lit les données des capteurs (inputs) et écrit des commandes vers les actionneurs (coils, outputs) à travers son programme de contrôle.

Ce lien permet au programme PLC (dans CODESYS) de contrôler le comportement de la scène 3D dans FactoryIO. Par exemple :

- Si un objet est détecté par un capteur à l'entrée, CODESYS peut déclencher un moteur pour faire avancer un convoyeur.

- Si une condition logique est remplie, un vérin peut être activé, ou un système d'arrêt d'urgence déclenché.

Cette architecture permet également de simuler des attaques sur les trames Modbus (ex : forcer une sortie sans autorisation), ce qui est utile pour les travaux pratiques en cybersécurité industrielle.

6.2 Architecture

6.2.1 Modèle Purdue et flux d'attaque

L'environnement de simulation s'inspire du modèle de référence Purdue, couramment utilisé dans la segmentation des architectures OT (Operational Technology). Il permet de situer les différents composants du système industriel et de mieux comprendre les vecteurs d'attaque possibles.

- Niveau 3 – HMI / SCADA : représenté ici par FactoryIO, chargé de visualiser et piloter le processus industriel simulé.
- Niveau 2 – Contrôleur logique programmable (PLC) : implémenté à l'aide de CODESYS V3, il assure le contrôle des actionneurs selon les instructions reçues.
- Niveau 1 – Capteurs et actionneurs : gérés virtuellement dans FactoryIO, ils traduisent les commandes du PLC en actions sur le processus.
- Niveau 0 – Processus physique : simulé par un système de convoyeur, réagissant aux commandes comme l'allumage ou l'arrêt d'un moteur.

L'expérimentation cible principalement la liaison entre le SCADA (niveau 3) et le PLC (niveau 2), qui communiquent via le protocole Modbus TCP. C'est à ce niveau que l'attaque est injectée, en exploitant l'absence de sécurité native du protocole.

6.2.2 Chaîne d'attaque technique

Le scénario d'attaque repose sur l'utilisation d'un script Python s'appuyant sur la bibliothèque pymodbus, permettant de générer des requêtes Modbus TCP personnalisées.

La chaîne technique se déroule en plusieurs étapes :

- Envoi d'une trame Modbus TCP : l'attaquant émet une requête de type Write Single Coil (0x05), visant à modifier l'état d'un bit de sortie du PLC.
- Cible : le contrôleur CODESYS : la requête est adressée directement à l'automate virtuel, sans authentification, ce qui rend l'opération possible en clair.
- Impact sur le processus physique : une fois la requête acceptée, le PLC exécute la commande, ce qui entraîne par exemple l'activation ou l'arrêt du moteur du convoyeur dans Factory IO

Impact sur le processus physique : une fois la requête acceptée, le PLC exécute la commande, ce qui entraîne par exemple l'activation ou l'arrêt du moteur du convoyeur dans FactoryIO.

6.3 Protocoles Industriel

6.3.1 Modbus TCP

Le protocole Modbus TCP est au cœur des échanges dans cette simulation.

Il repose sur une architecture client/serveur, où le client (ici : un script Python) envoie des requêtes vers un automate agissant comme serveur (CODESYS).

Les principales caractéristiques techniques sont les suivantes :

- Port utilisé : TCP/502 (port standard Modbus).
- Sécurité : aucune sécurité native (pas de chiffrement, ni d'authentification).
- Fonctions Modbus testées lors de l'expérimentation :
    - FC1 – Lire les coils : permet d'interroger l'état de sorties binaires (par ex. état moteur).
    - FC5 – Écrire un coil : utilisé pour activer ou désactiver une sortie (ex. ON/OFF moteur).
    - FC15 – Écrire plusieurs coils : permet d'envoyer plusieurs commandes en une seule requête.

L'absence de contrôle d'accès ou d'authentification dans ce protocole le rend particulièrement vulnérable aux attaques à distance, dès lors que l'accès réseau est possible.

6.3.2 Pile protocolaire CODESYS

Le runtime CODESYS V3 repose sur une pile logicielle structurée en couches, allant de l'application de contrôle jusqu'à la couche réseau :

- ✓ Application IEC 61131-3 : Logique de contrôle (langages LD, ST, FBD…)
- ✓ CODESYS V3 Runtime : Interprétation et exécution du code automate
- ✓ Modbus TCP/IP : Communication industrielle
- ✓ Ethernet (IEEE 802.3 : Support réseau physique

Cette structure permet à CODESYS d'exposer des interfaces de communication compatibles Modbus, accessibles en réseau local (ou étendu), et directement manipulables via des outils tiers ou des scripts.

6.4 Attaque par Script

6.4.1 Principe de l'Attaque

Un script Python malveillant peut exploiter les failles inhérentes au protocole Modbus TCP pour perturber le fonctionnement d'un système automatisé piloté par CODESYS et simulé dans FactoryIO.

Ce type d'attaque repose sur l'envoi de commandes non autorisées directement au contrôleur, contournant ainsi les mécanismes de sécurité classiques.

6.4.2 Mécanisme de l'Attaque

Le script établit une connexion avec l'automate via le port 502, typiquement utilisé pour Modbus TCP. Sans nécessiter d'authentification, il envoie des requêtes répétées de type Write Coil pour forcer l'état d'une sortie, par exemple un moteur de convoyeur. En bombardant le contrôleur de commandes à haute fréquence, le script provoque une désynchronisation entre les entrées (capteurs) et les sorties (actionneurs), entraînant un comportement erratique de la simulation FactoryIO.

6.4.3 Conséquences Opérationnelles

L'impact immédiat se traduit par des dysfonctionnements visibles dans la simulation : le convoyeur peut s'arrêter brutalement, redémarrer de manière inattendue ou ignorer les signaux de sécurité. Dans un environnement industriel réel, cela pourrait endommager des équipements ou interrompre une ligne de production.

6.4.4 Failles Exploitées

Cette attaque est possible en raison de plusieurs vulnérabilités. D'abord, Modbus TCP ne chiffre pas les données et n'exige aucune authentification, permettant à un attaquant d'envoyer des trames malveillantes depuis n'importe quel point du réseau. Ensuite, FactoryIO et CODESYS, dans leur configuration par défaut, ne vérifient pas la légitimité des commandes reçues, faisant implicitement confiance aux instructions Modbus sans validation contextuelle.

6.4.5 Détection et Prévention

Pour se prémunir contre ce type d'attaque, il est essentiel de segmenter le réseau industriel afin d'isoler les équipements critiques. L'ajout d'un pare-feu filtrant les adresses IP autorisées à communiquer sur le port Modbus est une première étape. En complément, l'activation de mécanismes de supervision permet de détecter les flux anormaux, comme un nombre élevé de requêtes Write Coil en un temps réduit. Enfin, une approche défensive plus robuste impliquerait la migration vers des protocoles sécurisés, tels qu'OPC UA, qui intègrent chiffrement et authentification.

6.5 Déroulement d'une attaque

6.5.1 Création de la Simulation dans FactoryIO

Scénario choisi :

- Un convoyeur transportant des boîtes d'un point A à B

Capteurs :

- %IX0.0 : Capteur de présence en entrée (déclenche le moteur).
- %IX0.1 : Capteur de fin de course (arrête le moteur).

Actionneur :

- %QX0.0 : Moteur du convoyeur.

6.5.2 Programmation du PLC dans CODESYS

- **Programmation du PLC en Langage à Contacts (LD)**
  - Le programme PLC_PRG est codé en Ladder Diagram (LD) pour gérer la logique du convoyeur.
  - Le PLC et le Gateway Modbus sont démarrés dans CODESYS pour établir la communication.
- **Connexion Réseau et Configuration Modbus**
  - La connexion s'effectue en Ethernet entre le PC et le contrôleur.
  - Dans CODESYS, le PLC est configuré en tant que client Modbus TCP.
  - L'adresse IP du serveur Modbus (simulateur ou autre PLC) est définie sur 127.0.0.1 (localhost pour les tests locaux).
- **Programmation de la Simulation**
  - Logique implémentée :

    Si un carton atteint le capteur (%IX0.0), le tapis roulant s'arrête (%QX0.0 = FALSE).
  - Le code est compilé et téléversé dans le PLC pour exécution.
- **Vérification du fonctionnement :**
  - La simulation dans FactoryIO est lancée pour tester le comportement :

    Lorsqu'un objet est détecté, le convoyeur s'arrête conformément à la logique programmée.
  - Les entrées/sorties Modbus sont surveillées via le moniteur CODESYS.

6.5.3 Connection CODESYS et FactoryIO

- On lance la connexion à partir de CODESYS.
- Sur FactoryIO, on choisit comment Driver Modbus Server, et en Network adapter on choisit Software LoopBack Interface 1, et on établit la connexion.
- Le tapis avance, la connexion est établie.

6.5.4 *Préparation de l'Attaque*

- On analyse le point faible qu'est le moteur du convoyer, donc l'attaque se portera là-dessus.
- On crée le script malveillant.

6.5.5 *Exécution de l'Attaque*

- Le script envoie une commande Write Coil pour forcer %QX0.0 à True.
- FactoryIO reçoit la mise à jour et active le moteur sans détection de boîte.
- Le convoyeur démarre de manière intempestive, perturbant le processus.

6.5.6 *Traces de l'Attaque*

En lançant Wireshark, les trames de ModBus suspect sont visibles.

6.6 Contre-mesures recommandées

Face aux vulnérabilités identifiées dans le protocole Modbus TCP, plusieurs contre-mesures peuvent être mises en place pour renforcer la sécurité des systèmes industriels. Ces recommandations s'appuient sur les bonnes pratiques décrites dans la norme IEC 62443, largement reconnue dans le domaine de la cybersécurité OT.

Mesure	Références	Description
Segmentation réseau	Purdue + Firewall	Implémenter une séparation stricte entre les niveaux (ex. SCADA ↔ PLC) à l'aide de pare-feu industriels. Exemple : architecture ISA avec firewalls Cisco ISA.
Listes de contrôle d'accès	IEC 62443-3-3 – SR 1.4	Restreindre l'accès aux IP ou ports autorisés. Exemple : *IF clientIP NOT IN [IP_LIST] THEN DROP*.
Tunnel VPN	IEC 62443-4-2	Utiliser un tunnel chiffré (IPsec ou OpenVPN) pour encapsuler le trafic Modbus TCP entre les équipements distants.
Detection d'anomalies	Bonnes pratiques industrielles	Déployer des systèmes IDS industriels (Nozomi Guardian, Siemens SINEC IDS, etc.) capables de détecter les flux suspects ou anormaux sur le réseau OT
Signature applicative Modbus	Custom (niveau applicatif)	Implémenter une logique dans le PLC (ex. via CODESYS) pour détecter les fonctions Modbus non autorisées. Exemple : *IF FC NOT IN [3, 4, 16] THEN LOG_EVENT()*.

Ces mesures ne se substituent pas les unes aux autres, mais sont complémentaires dans une approche défense en profondeur. En combinant la segmentation, la surveillance, le filtrage et le chiffrement, il devient possible de réduire considérablement la surface d'attaque d'un système industriel exposé.

6.7 Conclusion

Cette expérience démontre la faible sécurité native du protocole Modbus TCP, encore largement utilisé dans les systèmes industriels. En s'appuyant sur des outils simples, un attaquant peut simuler des commandes légitimes et perturber le système OT.

La mise en œuvre de bonnes pratiques de sécurité issues de la norme IEC 62443 est indispensable pour atteindre un niveau de maturité supérieur, notamment à travers la segmentation réseau, le durcissement des contrôleurs, et la surveillance en temps réel des échanges.