TER SYSTEMES INDUSTRIELS
M1 FSI 2024/2025

ATTAQUE DDoS SUR ICSSIM

1 Propriétés du document

CLASSIFICATION DU DOCUMENT	Protégée
REFERENCE DU DOCUMENT	CR006_OT_24_25
DATE D'EMISSION DU DOCUMENT	15/06/2025
VERSION DU DOCUMENT	1
AUTEURS	PARNET Cyril & DOUZI Youssef
PROPRIETAIRE DU DOCUMENT	AGOPIAN Roland – Aix Marseille

Tableau 1 – Propriété du document

2 Historique des révisions

VERSION	DATE	AUTEUR	RESUME DES MODIFICATIONS
1	15/06/2025	PARNET Cyril & DOUZI Youssef	Version n°1

Tableau 2 - Historique des révisions

3 Diffusion

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	11/06/2025
BERREBIHA Nasserdine	Adjoint au chef de projet	NB	11/06/2025
YABDA Redouane	Membre de l'équipe	RY	11/06/2025
DIA Mouhamadou Afiss	Membre de l'équipe	MaD	11/06/2025
DOUZI Youssef	Membre de l'équipe	YD	11/06/2025
KASMI Badreddine	Membre de l'équipe	KB	11/06/2025

Tableau 4 - Approbation

Table des matières

5 Présentation du document

Ce document présente une analyse des attaques sur systèmes industriels via le simulateur ICSSIM, avec un focus sur l'attaque DDoS. Cette attaque consiste à saturer le PLC simulé par des requêtes Modbus/TCP, le rendant incapable de piloter les capteurs/actionneurs. L'analyse couvre à la fois les mécanismes réseau bas niveau (sockets TCP, trames Modbus) et les impacts haut niveau sur le fonctionnement du système. Le tout illustre les vulnérabilités des protocoles industriels non sécurisés et l'importance d'une défense adaptée en milieu OT.

6 Présentation du Simulateur

6.1 Origine

Le simulateur ICSSIM (Industrial Control System SIMulator) est un environnement logiciel conçu pour reproduire de manière réaliste le fonctionnement d'un système industriel (ICS) dans un contexte sécurisé et contrôlé. Il a été développé par Ismail Khalil, chercheur au sein du Centre Interdisciplinaire de Sécurité Industrielle et des Réseaux (CISIR) de l'Université de Lille. Le projet est disponible en open source sur GitHub et a fait l'objet d'une publication scientifique détaillée en 2021.

L'objectif principal d'ICSSIM est de fournir une plateforme de simulation permettant de représenter le comportement d'un système de contrôle industriel, tout en étant capable de simuler diverses attaques informatiques ciblant les protocoles utilisés dans l'industrie, notamment Modbus/TCP et MQTT. Cette plateforme permet d'expérimenter différents scénarios d'attaques et de comprendre leur impact sur les composants industriels, sans exposer de véritables infrastructures à des risques.

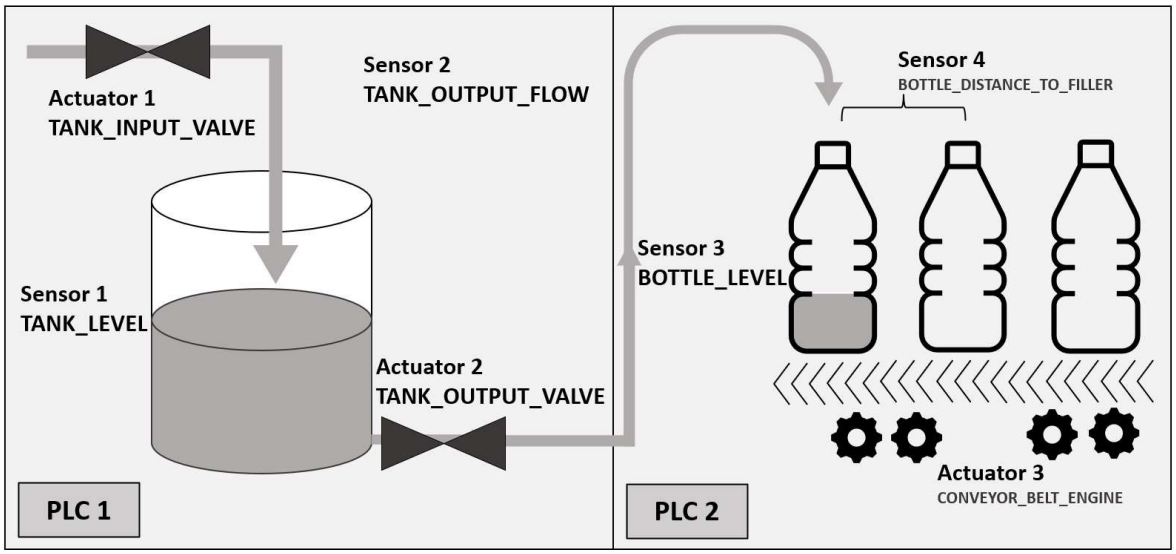
Techniquement, ICSSIM repose sur une architecture modulaire entièrement écrite en Python, avec une orchestration par Docker. Chaque composant du système industriel est représenté par un module indépendant : les automates programmables (PLC) sont simulés par des scripts comme PLC1.py ou PLC2.py, les interfaces opérateur (HMI) sont simulées par HMI1.py, HMI2.py ou HMI3.py, et la logique du processus industriel est centralisée dans FactorySimulation.py. Tous ces éléments interagissent entre eux via des protocoles standards (principalement Modbus/TCP), ce qui permet de reproduire des communications proches de celles d'un véritable réseau industriel.

ICSSIM permet également l'injection de différentes attaques comme le scan réseau, les attaques de replay, de MITM, d'injection de commande, ou encore des attaques DDoS, le tout dans un cadre pédagogique. L'utilisateur peut observer les effets de ces attaques sur le système simulé, analyser le trafic avec des outils comme Wireshark, et mettre en œuvre des stratégies de détection ou de mitigation.

Enfin, ICSSIM constitue un excellent outil d'apprentissage pour les étudiants et chercheurs en cybersécurité des systèmes industriels. Il permet d'expérimenter librement, de manière reproductible et sans risque, les faiblesses inhérentes aux protocoles industriels classiques et de développer des solutions de protection adaptées aux environnements OT.

6.2 Architecture

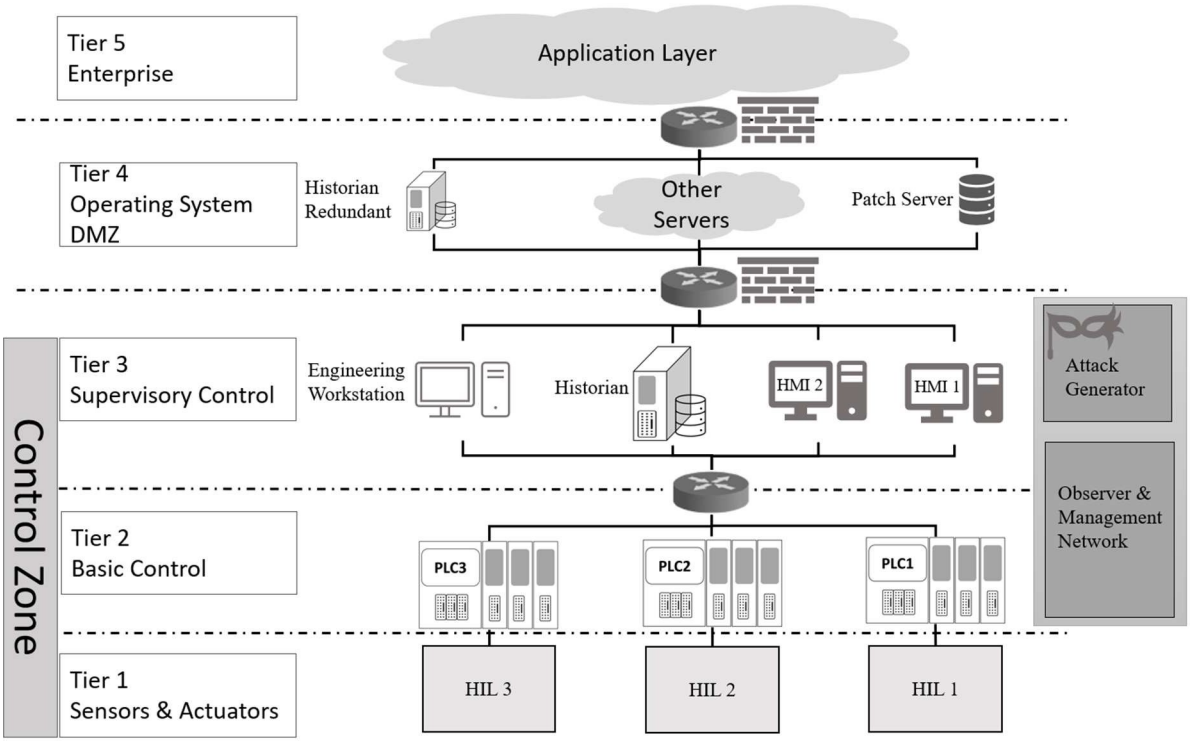
Dans le simulateur ICSSIM, les PLC1 et PLC2 représentent des automates programmables chargés de piloter le processus industriel. PLC1 contrôle la cuve principale (niveau de liquide, vannes d'entrée et de sortie) ainsi que le moteur du tapis roulant. Il reçoit des données de capteurs virtuels simulant le niveau du réservoir (tank_level), l'état des vannes (tank_input_valve, tank_output_valve) et la position des bouteilles. PLC2, de son côté, a la charge de la coordination en aval, notamment du remplissage des bouteilles, en interagissant avec d'autres capteurs (comme bottle_level ou bottle_distance_to_filler) et actionneurs. Les HMI (HMI1, HMI2, HMI3) servent d'interfaces opérateurs, permettant de visualiser l'état des capteurs ou de modifier manuellement certains paramètres. Tous ces composants sont reliés via des communications Modbus/TCP ou MQTT, ce qui permet de simuler un système industriel coordonné, mais vulnérable aux cyberattaques.



Architecture physique - Simulateur

6.3 PURDUE

Dans le modèle PURDUE, qui structure les systèmes industriels en couches, les composants d'ICSSIM se situent principalement entre le niveau 1 (contrôle de base) et le niveau 2 (supervision). Les PLC1 et PLC2 sont classés en niveau 1, car ils pilotent directement les capteurs et actionneurs simulés, et exécutent les logiques de contrôle en temps réel. Les capteurs virtuels (comme tank_level, bottle_level) sont en niveau 0, puisqu'ils représentent les objets physiques surveillés. Les HMI (interfaces opérateur), qui permettent la supervision locale ou distante, sont en niveau 2, tout comme les modules simulateurs type FactorySimulation.py. Enfin, les modules attaquants (Attacker.py, DDoSAgent.py) peuvent être considérés comme des entités intrusives simulant un accès frauduleux depuis l'extérieur du modèle PURDUE ou depuis un réseau compromis dans le niveau 3 IT/DMZ, soulignant les risques d'interconnexion entre réseaux industriels et informatiques.



Modèle PURDUE - Simulateur

7 Attaque DDoS

7.1 Introduction

Un DDoS, ou « Distributed Denial of Service », désigne une attaque visant à rendre un service indisponible en le submergeant de requêtes provenant de multiples sources. Le but est de saturer la cible pour qu'elle ne puisse plus répondre aux requêtes légitimes des utilisateurs.

Dans le cas d'un simulateur industriel comme ICSSIM, cette attaque est utilisée à des fins pédagogiques pour illustrer comment un protocole industriel — en l'occurrence Modbus/TCP — peut être exploité. En envoyant un grand nombre de requêtes vers un automate programmable (PLC), l'attaquant le rend indisponible. Cela entraîne un blocage de la communication avec les capteurs et actionneurs, perturbant ainsi l'ensemble de la simulation du système industriel.

7.2 Rappels de notions de réseau et programmation

- ### Qu'est-ce qu'un réseau, une adresse IP, un port ?

Un réseau IP est un ensemble d'ordinateurs ou de conteneurs connectés entre eux, capables d'échanger des paquets de données. Chaque machine du réseau possède une adresse IP unique, comme 192.168.0.5, qui permet de l'identifier.

Un port est un numéro attribué à une application ou un service sur une machine. Il permet de différencier plusieurs services sur une même adresse IP. Par exemple, le protocole Modbus/TCP utilise le port 502. Pour communiquer avec un automate (PLC) via Modbus, on établit donc une connexion TCP à son adresse IP, en précisant ce port.

- ## Socket TCP : comment un programme « parle » au réseau

Un socket est un concept utilisé en programmation réseau. Il s'agit d'une interface logicielle qui permet à un programme de communiquer à travers un réseau, un peu comme une prise réseau virtuelle.

Lorsqu'un programme joue le rôle de client TCP, il suit généralement les étapes suivantes : il commence par créer un socket (par exemple avec la fonction socket() en Python), puis il établit une connexion avec l'adresse IP et le port de la machine cible (comme le port 502 d'un automate). Ensuite, il envoie des données via ce socket (send ou sendall), lit éventuellement la réponse (recv), puis ferme la connexion (close).

Côté serveur — comme dans le cas du PLC dans ICSSIM — le processus commence aussi par la création d'un socket, mais cette fois « écoutant ». Le serveur l'associe à un port précis, comme le port 502, et se met en attente de connexions (listen). Lorsqu'une connexion entrante est détectée, le serveur l'accepte (accept) et crée un nouveau socket dédié à cette communication. Il peut alors lire la requête, la traiter, envoyer une réponse, et soit fermer la session, soit la garder ouverte selon le protocole.

- ## Processus et threads

Un processus est un programme en cours d'exécution, disposant de sa propre mémoire et de ses ressources indépendantes. Il fonctionne de manière isolée des autres processus du système.

Un thread, quant à lui, est une unité d'exécution qui s'exécute à l'intérieur d'un processus. Plusieurs threads peuvent coexister dans un même processus et partagent la même mémoire, ce qui permet une communication plus rapide entre eux.

Dans le cadre d'une attaque de type DDoS, l'utilisation de plusieurs processus ou threads permet d'envoyer un grand nombre de requêtes en parallèle. Cela permet de simuler plusieurs agents ou machines attaquantes, même à partir d'un seul script, et donc d'augmenter considérablement la pression exercée sur la cible.

7.3 Rappels sur Modbus/TCP

- ### Principe de Modbus/TCP

Le protocole Modbus/TCP fonctionne selon un modèle maître/esclave, également appelé client/serveur. Le client (souvent une interface HMI ou un système SCADA) envoie des requêtes, et le serveur (généralement un automate ou PLC) y répond. Ce protocole utilise le transport TCP, et plus précisément le port 502, qui lui est dédié.

Une trame Modbus/TCP est composée de deux parties : l'en-tête MBAP (Modbus Application Protocol) de 7 octets, suivi du PDU (Protocol Data Unit), qui contient la commande proprement dite.

L'en-tête MBAP comprend :

- Un identifiant de transaction (Transaction ID, 2 octets), qui permet d'associer les réponses aux requêtes.
- Un identifiant de protocole (Protocol ID, 2 octets), toujours égal à 0 dans le cas de Modbus.
- Une longueur (Length, 2 octets), indiquant la taille de la suite de données à lire.
- Un identifiant d'unité (Unit ID, 1 octet), utilisé pour adresser un esclave particulier, surtout utile dans le cas de passerelles vers des réseaux série.

Le PDU contient :

- Un code de fonction (Function Code, 1 octet), qui précise le type d'opération à effectuer (par exemple, 0x03 pour lire des registres).
- Et les données associées, comme l'adresse du registre ciblé, le nombre de registres à lire ou la valeur à écrire.

- Simplicité et absence de sécurité native

Le protocole Modbus/TCP présente plusieurs faiblesses majeures. Tout d'abord, il n'utilise pas de chiffrement, ce qui signifie que toutes les données circulent en clair sur le réseau. Ensuite, il n'y a aucune authentification : n'importe quelle application connectée au réseau peut ouvrir une connexion TCP sur le port 502 et envoyer des requêtes valides sans vérification. De plus, il n'existe pas de contrôle d'intégrité, ce qui laisse la porte ouverte à la modification des paquets en transit. Enfin, le protocole ne prévoit aucune limitation de fréquence, c'est-à-dire qu'il ne dispose pas de mécanismes de throttling ou de contrôle de flux au niveau applicatif. Ces caractéristiques rendent Modbus/TCP particulièrement vulnérable à différents types d'attaques, y compris aux attaques par déni de service distribué (DDoS).

7.4 Architecture ICSSIM et rôle du PLC

- Schéma global simplifié

L'attaquant, appelé ici DDoSAgent, envoie des requêtes Modbus/TCP vers le PLC. Le PLC, qui est programmé en Python, contient un serveur TCP qui écoute sur le port 502. Il dispose également d'une boucle interne qui, dans des conditions normales, lit et écrit des variables simulant des capteurs et des actionneurs. Ces valeurs sont ensuite publiées ou peuvent être consultées via le protocole Modbus.

Cependant, si le serveur Modbus est submergé par un grand nombre de requêtes, cette boucle interne est retardée voire bloquée. En conséquence, les capteurs et actionneurs simulés ne sont plus mis à jour correctement, ce qui provoque l'affichage, sur l'interface homme-machine (HMI), de valeurs figées ou erratiques.

- Comment le PLC traite une requête Modbus

Lorsqu'une requête Modbus arrive, le serveur accepte la connexion TCP. Ensuite, il lit la trame complète comprenant l'en-tête MBAP et la PDU. Cette trame est ensuite interprétée : par exemple, si la fonction demandée est 0x03, le serveur lit certains registres internes, qui sont en fait des variables Python simulant l'état du système. Puis, il construit une réponse en reprenant l'en-tête MBAP avec le même identifiant de transaction, ainsi que la PDU contenant les valeurs lues. Cette réponse est envoyée au client, après quoi le serveur peut soit fermer la connexion, soit attendre d'autres requêtes.

Parallèlement, le PLC exécute une boucle interne qui tourne périodiquement. Cette boucle met à jour les variables simulées représentant les capteurs, exécute la logique de contrôle, comme par exemple l'ajustement des vannes en fonction des niveaux mesurés, puis répond aux requêtes Modbus.

Si un trop grand nombre de requêtes arrivent, le serveur peut mettre plus de temps à accepter et traiter chaque connexion, ce qui repousse l'exécution de la boucle interne. Il peut aussi refuser ou délaisser certaines connexions lorsque la file d'attente TCP est saturée. En pratique, cela ralentit la boucle interne, ce qui provoque un gel de la simulation.

7.5 Vue d'ensemble de l'attaque DDoS dans ICSSIM

- High-level : rôle d'Attacker.py

Lorsque l'on choisit l'option DDoS (par exemple l'option 4 dans le menu), le script Attacker.py entre en action. Il commence par lire les paramètres fournis, notamment l'adresse IP du PLC ciblé, le port de communication (généralement 502), le nombre d'agents à lancer, ainsi que la durée ou les conditions d'arrêt de l'attaque. Ensuite, il crée et lance plusieurs instances du composant DDoSAgent, chacune étant un processus ou un thread chargé de générer du trafic.

Chaque agent exécute une boucle infinie dans laquelle il envoie des requêtes Modbus/TCP vers le PLC. Le script affiche en parallèle des messages dans les journaux pour indiquer l'état d'exécution, comme "INFO Created" ou "applied ddos attack successfully". L'utilisateur peut interrompre l'attaque à tout moment, par exemple en appuyant sur une touche ou en tuant les processus manuellement.

- ## Low-level : fonctionnement d'un DDoSAgent

Lors de l'initialisation, chaque DDoSAgent récupère les paramètres nécessaires comme l'adresse IP cible (target_ip), le port (target_port), un éventuel préfixe de nom, ainsi qu'un logger partagé pour tracer les événements. Il peut également choisir aléatoirement un registre ou une fonction Modbus à cibler, souvent une lecture simple pour générer rapidement du trafic sans modifier l'état du système.

Dans sa boucle d'attaque, chaque agent crée un socket TCP, se connecte au PLC sur le port 502, construit une trame Modbus valide (avec un en-tête MBAP et un PDU), puis envoie cette trame. Il peut éventuellement recevoir une réponse, mais celle-ci est souvent ignorée pour accélérer l'envoi des requêtes. Le socket est ensuite fermé, et la boucle recommence

Quelques précisions supplémentaires : le Transaction ID est souvent incrémenté ou aléatoire pour éviter les duplications. Le Unit ID est généralement fixé à 1. La fonction utilisée est souvent une lecture (code 0x03), car elle n'a pas d'effet sur l'état interne du PLC, mais elle mobilise tout de même les ressources du serveur. Le temps entre deux requêtes est volontairement réduit au minimum, voire nul, pour maximiser la charge générée. Cette boucle se répète indéfiniment jusqu'à arrêt manuel.

- ## Concurrence et distribution

Pour exécuter plusieurs agents en parallèle, deux approches sont possibles : le multiprocessing ou le threading.

Avec le multiprocessing, chaque agent est lancé comme un processus indépendant, ce qui lui donne son propre espace mémoire. Cela offre un bon niveau d'isolation mais consomme davantage de ressources système.

Avec le threading, tous les agents partagent le même processus et la même mémoire, ce qui est plus léger mais peut être limité par le GIL (Global Interpreter Lock) de Python en cas de traitements CPU intensifs. Toutefois, dans ce contexte orienté I/O réseau, le threading est généralement suffisant car le goulot d'étranglement réside dans les échanges réseau, non dans les calculs.

Le nombre d'agents est un facteur clé : plus il y en a, plus la saturation du PLC est rapide. Le serveur risque de voir sa file d'attente TCP (appelée backlog) se remplir, provoquant des délais de traitement ou des refus de connexions.

Enfin, pour simuler un botnet distribué, on peut répartir les agents sur plusieurs conteneurs Docker ou machines différentes dans ICSSIM. Cela reproduit une attaque DDoS provenant de multiples sources, rendant sa détection et sa neutralisation plus difficiles.

7.6 Niveau réseau : détails des trames et comportement TCP

- ### Établissement de la connexion TCP

Pour chaque communication entre un agent attaquant et le PLC via Modbus/TCP, une connexion TCP doit d'abord être établie. Ce processus suit le mécanisme classique du "3-way handshake" :

1. Le client (l'agent) envoie un paquet SYN au serveur (le PLC, sur le port 502).
2. Le serveur répond avec un SYN-ACK.
3. Le client termine l'établissement de la connexion en envoyant un ACK.

Chaque agent répète ce cycle à chaque nouvelle tentative de connexion, surtout s'il ferme la connexion après chaque trame envoyée. Ce processus, bien qu'invisible pour l'utilisateur, consomme des ressources côté serveur : il doit allouer de la mémoire, ouvrir un socket, et gérer les états de connexion, ce qui devient coûteux en cas d'attaque massive.

Une alternative consiste à garder la connexion TCP ouverte et à envoyer plusieurs requêtes consécutives sur la même session. Cela réduit le nombre de handshakes, mais le PLC doit tout de même traiter chaque requête reçue, ce qui le surcharge progressivement.

- ### Paquets Modbus/TCP

En analysant le trafic réseau avec des outils comme Wireshark ou tcpdump, on peut observer les paquets échangés lors d'une attaque DDoS Modbus. Chaque requête envoyée par un agent est un paquet TCP contenant une trame Modbus avec un en-tête MBAP et un PDU. Typiquement, la fonction utilisée est 0x03 (lecture de registres), accompagnée de l'adresse de départ et du nombre de registres à lire.

Le PLC répond par une trame similaire contenant les valeurs demandées. Toutefois, dans la majorité des cas, l'attaquant n'attend pas la réponse ou l'ignore complètement, car son but est seulement de générer de la charge réseau.

La fréquence d'envoi est très élevée : un seul agent peut envoyer plusieurs dizaines voire centaines de requêtes par seconde. Multiplié par le nombre d'agents, cela génère plusieurs milliers de paquets par seconde, inondant rapidement le serveur.

- ## Ressources consommées

Une attaque DDoS génère une consommation massive de ressources sur le serveur ciblé, ici le PLC.

- La file d'attente TCP (ou backlog) se remplit rapidement. Si elle atteint sa capacité maximale, les nouvelles connexions sont mises en attente ou rejetées, ce qui empêche les clients légitimes de communiquer avec le PLC.
- Chaque nouvelle connexion TCP consomme une entrée dans la table des sockets du système d'exploitation. Cela mobilise la pile réseau du PLC et finit par atteindre les limites du système.
- Le processeur (CPU) du PLC est aussi fortement sollicité. Pour chaque requête, le serveur Modbus doit lire l'en-tête MBAP, interpréter le PDU, exécuter la logique de lecture ou d'écriture des registres simulés, construire une réponse, et potentiellement l'envoyer. Même si la réponse n'est pas attendue, le travail est effectué.
- La mémoire vive (RAM) est également utilisée pour gérer les buffers de chaque socket et stocker temporairement les trames. Plus il y a de connexions, plus la mémoire est sollicitée.
- Enfin, la latence des requêtes légitimes augmente considérablement. Plus le PLC est saturé, plus il met de temps à répondre aux HMI ou à d'autres composants. Si cette surcharge persiste, la boucle de simulation interne du PLC (qui gère les capteurs et actionneurs simulés) est ralentie, voire bloquée, rendant la simulation incohérente ou figée.

7.7 Effet sur la boucle interne du PLC et la simulation de l'usine

- ### Boucle normale du PLC

Le PLC dans ICSSIM fonctionne selon une logique de cycle continu appelée boucle de contrôle. À chaque itération, il suit une séquence bien définie. Il commence par lire les données issues des capteurs internes simulés, comme les niveaux de liquide ou les positions de vannes. Ensuite, il applique une logique de régulation, par exemple : si le niveau du réservoir est trop bas, alors il faut ouvrir une vanne.

Une fois cette décision prise, il modifie l'état des actionneurs simulés pour appliquer cette logique. Ensuite, il met à jour les variables accessibles via Modbus, afin que d'autres composants comme les interfaces HMI ou d'autres PLC puissent récupérer l'état du système. Enfin, il attend un court délai (souvent 100 millisecondes) avant de recommencer ce cycle.

- ### Effet de la saturation Modbus/TCP

Quand le serveur Modbus du PLC est submergé de requêtes – comme lors d'une attaque DDoS – les ressources CPU sont principalement mobilisées pour accepter, lire, et traiter ces connexions réseau. Cela repousse ou empêche la boucle de contrôle interne de s'exécuter correctement.

La conséquence directe est que cette boucle ne se répète plus à une fréquence régulière. Les capteurs ne sont plus mis à jour, ou avec du retard, et les décisions de contrôle ne sont plus prises à temps. Cela peut entraîner des valeurs figées dans la simulation, ou des comportements incohérents.

Dans un automate réel, ce genre de perturbation peut activer des mécanismes de sécurité internes et entraîner un arrêt d'urgence du système. Dans ICSSIM, cela se manifeste par des dysfonctionnements dans l'affichage ou la logique de l'usine simulée. Par ailleurs, les HMI qui interrogent régulièrement le PLC via Modbus ou MQTT peuvent rencontrer des délais importants ou des erreurs de connexion, signalant des « timeouts » ou affichant des données obsolètes.

- Conséquences visibles dans ICSSIM

Plusieurs effets concrets peuvent être observés dans la simulation lorsque le PLC est saturé :

- Les valeurs restent figées, comme le niveau du réservoir (tank_level) qui n'évolue plus, même si la vanne est censée être ouverte.
- Des erreurs ou messages de timeout apparaissent dans les interfaces HMI, traduisant des problèmes de communication avec le PLC.
- Le système peut montrer un comportement erratique : par exemple, si le timing de la boucle est perturbé, une vanne peut rester ouverte trop longtemps, entraînant un débordement ou un arrêt imprévu.
- Les ressources simulées peuvent être bloquées : si d'autres composants comme PLC2 ou HMI2 communiquent via le même réseau ou broker MQTT, ils subissent également l'impact de la saturation, même s'ils ne sont pas directement ciblés. Cela montre bien que l'impact d'un DDoS peut se propager à tout le système.

7.8 Observabilité et diagnostic

- Sur l'attaquant

Côté attaquant, le script Attacker.py fournit plusieurs indications sur l'état de l'attaque. Lorsqu'un agent est lancé, on voit dans les journaux le message « Created », suivi de « applied ddos attack successfully », ce qui confirme que l'attaque a bien été initiée. En étudiant le code ou en ajoutant des compteurs, on peut également mesurer le nombre de requêtes envoyées par seconde pour évaluer la pression exercée sur la cible.

- Sur le réseau

À l'aide d'outils comme tcpdump ou Wireshark, on peut analyser le trafic réseau généré par l'attaque. En filtrant sur le port 502 (tcp.port == 502), on observe un flux continu et rapide de paquets Modbus aux contenus très similaires. Cela permet de visualiser le bombardement en cours. On peut également mesurer la latence entre chaque requête et sa réponse pour évaluer la saturation du serveur. En affichant un graphique de débit, on constate une montée brutale du trafic lorsque le DDoS est actif, ce qui confirme visuellement la surcharge du réseau.

- Sur le PLC (simulation)

Du côté du PLC simulé, plusieurs indicateurs témoignent de l'impact de l'attaque. En surveillant les processus avec top ou htop, on remarque une augmentation importante de l'utilisation du CPU, causée par le traitement des nombreuses requêtes. Si ICSSIM est configuré pour logger la durée de chaque cycle de contrôle, ces logs montrent que le temps d'exécution de la boucle interne augmente nettement, ce qui dégrade la performance de la simulation. Enfin, si le code du serveur Modbus enregistre les requêtes reçues, les journaux afficheront des timestamps très rapprochés, bien plus que dans une situation normale.

- Sur l'HMI

L'impact de l'attaque DDoS est également visible du côté des interfaces Homme-Machine (HMI). Tout d'abord, on peut remarquer une augmentation du temps de latence : les valeurs mettent plus de temps à s'afficher ou ne se mettent plus à jour du tout. Si le code de la HMI intègre une gestion des erreurs, des messages d'alerte comme « timeout » ou « connection refused » peuvent apparaître lorsque le PLC ne répond plus à temps.

Enfin, les HMI peuvent afficher des valeurs incohérentes ou figées, car elles continuent d'afficher les dernières données valides reçues, même si celles-ci ne reflètent plus l'état réel du système. Cela montre bien que l'attaque perturbe la synchronisation entre la supervision et le contrôle.

7.9 Éléments très bas niveau : comment une requête est formée et envoyée

- ### Particularités du handshake TCP

Chaque appel à connect() sur un socket TCP déclenche un handshake complet. Si le code effectue de nombreuses connexions successives, le PLC peut se retrouver avec une file d'attente de connexions incomplètes (état SYN_RECV) ou trop de connexions en attente d'acceptation. Cela conduit à des refus de connexion, qui se manifestent par des paquets TCP RST (reset), ou à un ralentissement global du serveur dans l'acceptation des nouvelles connexions.

- ### Réception de la réponse

Dans la plupart des cas, les agents DDoS n'attendent pas la réponse du PLC après l'envoi de la trame. Cela leur permet d'enchaîner les requêtes le plus rapidement possible. Toutefois, certains codes incluent une réception (recv) après l'envoi, suivie de la fermeture du socket. Même si cette étape consomme un peu plus de ressources, elle n'interrompt pas nécessairement la boucle d'envoi si elle est bien gérée.

- ### Temps entre requêtes

Les agents sont conçus pour envoyer les requêtes aussi rapidement que possible. Il n'y a généralement aucun délai (sleep) entre deux envois, ou alors un sleep(0.001) très court. Cela maximise la fréquence d'attaque et la pression exercée sur le PLC, ce qui est précisément l'objectif d'un DDoS.

7.10 Éléments de haut niveau : orchestration et paramètres

- ### Paramètres configurables

Plusieurs paramètres peuvent être ajustés lors de l'exécution de l'attaque DDoS dans ICSSIM. L'utilisateur peut définir l'IP cible, qui correspond à l'adresse du PLC attaqué. Le port est par défaut 502, mais peut être modifié si le serveur Modbus écoute sur un autre port. Le nombre d'agents détermine l'intensité de l'attaque : plus il est élevé, plus la pression exercée sera forte.

On peut également choisir le type de requête, souvent une lecture (non destructive) pour éviter d'altérer le fonctionnement du système. L'écriture, plus risquée, est moins fréquemment utilisée dans le simulateur. La durée de l'attaque peut être définie ou laissée indéfinie, dans ce cas l'utilisateur l'interrompt manuellement. Enfin, le script peut générer des logs qui enregistrent des statistiques utiles comme le nombre de requêtes envoyées, les erreurs rencontrées, ou la latence.

- Lancement

Le lancement de l'attaque se fait en exécutant la commande python3 Attacker.py. L'utilisateur sélectionne ensuite l'option correspondant à l'attaque DDoS. Le script affiche des messages confirmant le démarrage, comme « Created », puis initialise et lance chaque agent dans un processus ou thread distinct. Chacun démarre alors sa propre boucle d'envoi de requêtes vers le PLC.

- Arrêt

L'arrêt de l'attaque peut être réalisé en capturant un signal d'interruption, par exemple en appuyant sur Ctrl+C dans le terminal. Le script peut alors propager ce signal à tous les agents pour un arrêt propre. Sinon, l'utilisateur peut aussi mettre fin manuellement aux processus depuis le système d'exploitation.

7.11 Comparaison à un DDoS industriel réel

Dans un contexte réel, une attaque DDoS sur un système industriel présente des similitudes mais aussi quelques différences avec ce qui est simulé dans ICSSIM.

Similitudes :

- Il s'agit toujours d'une inondation de requêtes légitimes, ici des trames Modbus, dans le but de saturer le serveur cible (le PLC).
- L'attaque est distribuée : elle peut venir de multiples sources (ou agents), comme des bots ou des scripts tournant sur plusieurs machines.
- Elle entraîne une consommation excessive de ressources, en particulier du CPU, et remplit la file TCP des connexions entrantes.

Différences :

- Dans ICSSIM, l'environnement est écrit en Python et plus souple ; dans le monde réel, un PLC utilise un système embarqué limité et propriétaire.
- Dans une attaque réelle, les sources sont souvent des objets connectés compromis, des postes industriels ou des malwares.

Enseignement :

L'attaque DDoS met en lumière une faiblesse structurelle de Modbus/TCP : toute requête valide peut suffire à surcharger un système. Il n'est pas nécessaire d'exploiter une faille complexe ; le simple volume de requêtes légitimes suffit à paralyser le fonctionnement.

7.12 Observations et outils de mesure

Pour comprendre et mesurer l'effet de l'attaque, différents outils et méthodes peuvent être utilisés :

Monitoring CPU du PLC :

- Dans le simulateur Python, on peut observer une montée de la consommation CPU via des outils comme top ou htop.
- Sur un automate réel, des interfaces de supervision ou des sondes SNMP peuvent permettre une observation similaire.

Mesure de latence :

- En testant le temps de réponse d'une requête Modbus avant et pendant l'attaque, on constate un allongement net.
- Il est possible d'automatiser ce test avec un script client qui effectue périodiquement des lectures et mesure le délai de réponse.

Capture réseau :

- Avec Wireshark ou tcpdump, on peut filtrer sur le port 502 pour observer le trafic Modbus.
- On y verra le volume, la fréquence et la taille des paquets envoyés.

Logs du simulateur :

- ICSSIM peut enregistrer le temps nécessaire à chaque cycle de traitement interne du PLC. Une hausse brutale indique un ralentissement dû à l'attaque.

Visualisation HMI :

- Les HMI permettent de constater que certaines valeurs ne se mettent plus à jour, ou bien le font avec beaucoup de retard, preuve d'un dysfonctionnement.

7.13 Conséquences et recommandations

• Impact pédagogique

L'attaque DDoS simulée dans ICSSIM a une forte valeur éducative. Elle montre concrètement l'importance de mettre en place une défense en profondeur dans les réseaux industriels. Cela inclut la segmentation du réseau, le filtrage par pare-feu, l'utilisation de tunnels chiffrés (VPN), la mise à jour régulière des firmwares et l'ajout de solutions de détection d'intrusion (IDS/IPS) adaptées aux protocoles industriels comme SCADA.

Elle souligne aussi la valeur des tests en environnement simulé, qui permettent d'évaluer les limites d'un système sans risquer d'endommager une infrastructure réelle. Enfin, elle fournit un cadre idéal pour développer et tester des mécanismes de détection d'anomalies, par exemple via l'analyse de fréquence des requêtes ou la détection de comportements inhabituels dans les trames.

• Conseils pour contrer DDoS Modbus

Voici plusieurs bonnes pratiques pour protéger un environnement industriel contre les attaques DDoS sur Modbus :

- Limiter le nombre de connexions par seconde : certaines passerelles ou pare-feu permettent de configurer des seuils.
- Utiliser une liste blanche d'IP : seuls les clients autorisés peuvent accéder au port 502.
- Surveiller en temps réel : déclencher des alertes en cas de pic anormal de trafic.
- Encapsuler le trafic Modbus dans des tunnels sécurisés (TLS ou VPN) pour éviter l'accès direct.

- Segmenter les réseaux OT : isoler le réseau de contrôle des autres réseaux de l'entreprise pour limiter les points d'entrée.

7.14 Explications « évidentes » pour les novices

Pourquoi « inonder » le PLC de requêtes

C'est comme un guichet qui ne peut servir qu'une personne à la fois. Si 100 personnes arrivent en même temps, la file devient longue, et ceux qui ont vraiment besoin du service doivent attendre. Le PLC agit de la même manière : s'il reçoit trop de requêtes, il n'arrive plus à répondre à celles qui sont légitimes.

Qu'est-ce qu'un « agent » ?

C'est un petit programme autonome qui envoie des requêtes au PLC. Si on en lance plusieurs, cela revient à simuler plusieurs ordinateurs qui attaquent en même temps.

Pourquoi utiliser Python et des sockets ?

Python est un langage simple et accessible. Il permet de créer facilement des connexions réseau (sockets), de construire et d'envoyer des paquets. C'est parfait pour apprendre à comprendre et simuler des attaques.

À quoi sert le port 502 ?

C'est la porte d'entrée du PLC pour les communications Modbus. Si on bloque ou surcharge cette porte, on empêche toute communication avec le reste du système.

Que fait normalement un PLC ?

Il lit les capteurs (par exemple, le niveau d'un réservoir), applique une logique (ouvrir ou fermer une vanne), et envoie les résultats aux HMI. S'il est surchargé, il ne peut plus assurer ces tâches.

Pourquoi ICSSIM est écrit en Python ?

Pour reproduire le comportement d'un système industriel de manière sûre et modifiable, sans matériel physique. Cela permet d'expérimenter et de comprendre la cybersécurité OT dans un cadre totalement contrôlé.

7.15 Exemple pas à pas (simplifié)

1. **Avant l'attaque :**
   Le PLC fonctionne normalement. Il répond aux requêtes des HMI toutes les secondes. Sa boucle interne se répète toutes les 100 millisecondes, et l'utilisation CPU est faible, autour de 10 %.

2. **Lancement de l'attaque :**
   L'utilisateur exécute Attacker.py et choisit l'attaque DDoS. Cinquante agents sont lancés, chacun envoyant environ 100 requêtes par seconde, soit un total de 5000 requêtes par seconde. Le PLC commence à accumuler les connexions TCP, ce qui surcharge sa capacité de traitement.

3. **Pendant l'attaque :**
   Le CPU du PLC monte à 90-100 %. Les HMI mettent plusieurs secondes à obtenir une réponse, parfois rien du tout. La boucle de régulation est retardée. Par exemple, un réservoir peut rester figé ou se remplir jusqu'à déborder. Wireshark montre une avalanche de paquets Modbus sur le port 502.

4. **Arrêt de l'attaque :**
   En appuyant sur Ctrl+C, l'utilisateur interrompt le script. Les agents ferment leurs sockets. Le PLC peut alors reprendre un fonctionnement normal, son CPU redescend, et les cycles internes se rétablissent. La simulation retrouve un comportement cohérent.

7.16 Comment contrer une attaque DDoS dans un environnement industriel

- ### Défense réseau (périmétrique)

La première ligne de défense consiste à renforcer la sécurité périmétrique :

- Un pare-feu bien configuré permet de restreindre l'accès au port 502 aux seules adresses IP autorisées, comme les interfaces HMI ou les serveurs SCADA internes. Il bloque également les connexions non authentifiées ou suspectes.
- La segmentation du réseau OT est essentielle. Elle permet d'isoler le réseau de contrôle industriel du reste de l'entreprise ou d'Internet, réduisant ainsi la surface d'attaque. L'utilisation de VLAN ou de DMZ permet aussi de limiter la propagation d'une attaque.
- La mise en place d'une liste blanche d'adresses IP permet de n'autoriser que certains clients spécifiques à communiquer avec le PLC.
- Des outils de détection d'anomalies (IDS/IPS), comme Snort avec des règles adaptées à Modbus, permettent de repérer des comportements anormaux comme un nombre inhabituel de requêtes ou des motifs répétitifs suspects.

- ### Résilience du serveur Modbus (PLC)

Le serveur lui-même peut être renforcé pour mieux résister :

- Il est possible de limiter le nombre de connexions TCP simultanées, soit via le système d'exploitation, soit via la configuration du serveur Modbus, afin d'éviter les débordements.
- Des délais de réponse artificiels peuvent être ajoutés en cas de fréquence de requêtes trop élevée, pour décourager les attaques automatisées.
- Si le matériel le permet, on peut activer une authentification renforcée, notamment en utilisant une version sécurisée du protocole, comme Modbus sur TLS ou via des passerelles industrielles sécurisées.

- ## Surveillance et réaction

Un système de surveillance efficace permet de réagir rapidement en cas d'attaque :

- Des logs détaillés et du monitoring en temps réel permettent de repérer rapidement un comportement anormal, comme une augmentation soudaine du trafic.
- Des mesures correctives comme le redémarrage automatique du service Modbus ou la bascule vers un automate de secours peuvent être mises en place pour assurer la continuité du service.

- ## Approche proactive

Enfin, une stratégie proactive est indispensable à long terme :

- Il est utile de réaliser des tests réguliers en environnement simulé, par exemple avec ICSSIM, afin d'anticiper les vulnérabilités.
- La formation du personnel est également un levier important. Les opérateurs et ingénieurs doivent être sensibilisés aux risques DDoS et aux bonnes pratiques de sécurité OT.
- Enfin, il est crucial de maintenir les firmwares à jour, afin de corriger les vulnérabilités connues et de bénéficier des améliorations de sécurité fournies par les fabricants.

En combinant ces différentes approches – protection réseau, renforcement du serveur, surveillance active et prévention – il est possible de limiter fortement l'efficacité d'une attaque DDoS, même sur un protocole aussi simple que Modbus/TCP.

7.17 Conclusion

L'attaque DDoS simulée dans ICSSIM démontre de façon claire et détaillée comment un protocole industriel simple et non sécurisé comme Modbus/TCP peut être détourné à des fins malveillantes. Cette simulation explore toutes les couches de l'attaque, depuis les aspects bas niveau — comme la création de sockets, l'établissement du handshake TCP, et la construction des trames Modbus (MBAP + PDU) — jusqu'aux aspects haut niveau, comme l'organisation d'agents multiples, l'impact sur la boucle de contrôle interne du PLC, et les effets visibles dans la simulation complète de l'usine.

L'ensemble repose sur deux éléments clés : la simplicité du protocole et l'absence de mécanismes de sécurité intégrés dans Modbus/TCP. Cela rend l'attaque facile à implémenter, même avec de simples scripts en Python.

Cette démonstration, accessible aussi bien aux débutants qu'aux professionnels, permet de :

- Comprendre concrètement le fonctionnement des communications réseau (connexion, envoi, réception),
- Voir comment une attaque puissante peut être menée avec un outil aussi basique qu'un script Python,
- Souligner l'importance d'une stratégie de défense en profondeur dans les environnements OT, reposant sur la segmentation du réseau, le filtrage, la surveillance active, et le chiffrement des communications.

À retenir : une requête Modbus valide, en soi inoffensive, devient dangereuse lorsqu'elle est répétée massivement. Ce type d'attaque montre à quel point la sécurisation des protocoles industriels est cruciale dans les infrastructures réelles.

8 Acronymes et définitions

Table des acronymes

ACRONYME	SIGNIFICATION
OT	Operational Technology
ICS	Industrial Control System
PLC	Programmable Logic Controller
SCADA	Supervisory Control And Data Acquisition
HMI	Human-Machine Interface
MBAP	Modbus Application Protocol (Header)
PDU	Protocol Data Unit
MQTT	Message Queuing Telemetry Transport
DDoS	Distributed Denial of Service
VPN	Virtual Private Network
DMZ	Demilitarized Zone
API	Application Programming Interface
CPU	Central Processing Unit
IP	Internet Protocol address
GIL	Global Interpreter Lock (Python)
ModBus/TCP	Modbus over TCP/IP
TCP/IP	Transmission Control Protocol / Internet Protocol
IDS/IPS	Intrusion Detection / Prevention System

Table 5 – Table des acronymes

9 Sources

1. Le code source du dépôt GitHub : https://github.com/ikhalil/ICSSIM
2. NIST SP 800-82 (Guide to Industrial Control Systems Security)
3. ISA/IEC 62443 (Security for Industrial Automation and Control Systems)
4. Spécification officielle : Modbus Application Protocol v1.1b3 (Modbus.org)
5. Documentation : « Modbus TCP/IP Protocol Overview » (Schneider Electric)