# Model-based IDS design pour ICS

Mohamad-Houssein Monzer

Communauté
UNIVERSITÉ Grenoble Alpes
UNIVERSITÉ LIBANAISE
FACULTÉ DES SCIENCES

# THÈSE

Pour obtenir le grade de

**DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES préparée dans le cadre d'une cotutelle *entre la Communauté Université Grenoble Alpes* et l'université Libanaise**

Spécialité : **INFORMATIQUE - AUTOMATIQUE PRODUCTIQUE**

Arrêté ministériel : le 6 janvier 2005 – 25 mai 2016

Présentée par

**Mohamad Houssein MONZER**

Thèse dirigée par **Jean-Marie FLAUS** et **Alaa GHAITH** codirigée par **Kamal BEYDOUN** préparée au sein des **Laboratoires G-SCOP et MECRL** dans **les Écoles Doctorales EEATS et Faculté de sciences**

# Model-based IDS design pour ICS

Thèse soutenue publiquement le **12 novembre 2020** ,
devant le jury composé de :

**M. Abed Ellatif Samhat**
Professeur à l'université Libanaise, Président du jury
**M. Kratz FRÉDÉRIC**
Enseignant-chercheur chez Institut National des Sciences Appliquées Centre Val de Loire, Rapporteur (visio conférence)
**M. Rida KHATOUN**
Professeur associé à l'université Télécom Paris, Rapporteur (visio conférence)
**M. Youssef LAAROUCHI**
Chef de projet R&D chez EDF France, Examinateur (visio conférence)
**M. Mohamad NASSAR**
Professeur associé à l'université Américaine de Beyrouth, Examinateur
**M. Kamal BEYDOUN**
Professeur associé à l'université Libanaise, Co-directeur de thèse
**M. Alaa GHAITH**
Professeur à l'université Libanaise, Directeur de thèse
**M. Jean-Marie FLAUS**
Professeur à l'Univ. Grenoble Alpes (UGA), Directeur de thèse (visio conférence)

*À ma famille,*

*À mes amis,*

*À mes rencontres de la vie, d'aujourd'hui et d'hier,*

*À tous ceux qui m'aiment,*

*À tous ceux que j'aime,*

# Remerciements

Le travail est terminé, cependant, avant de tourner la page, je tiens à remercier Dieu, le tout puissant et miséricordieux, qui m'a donné la force, l'intelligence et la patience d'accomplir ce modeste travail. Ensuite, je voudrais remercier certaines des personnes merveilleuses, qui étaient comme un morceau de sucre dans une tasse de thé donnant un goût merveilleux.

Le travail présenté dans ce mémoire a été effectué au Laboratoire des Sciences pour la Conception, l'Optimisation et la Production (G-SCOP) en cotutelle avec le laboratoire (MECRL). A ce titre, je tiens à remercier tout d'abord Monsieur François Villeneuve et Monsieur Hussein Chible, respectivement les directeurs du laboratoire G-SCOP et du laboratoire MECRL, de m'avoir accueilli et permis d'effectuer mes travaux de recherche tout au long de cette phase de thèse.

J'exprime toute ma gratitude à mon directeur de thèse, Monsieur Jean-Marie Flaus, professeur à l'Université de Grenoble Alpes. En tant que Directeur de thèse, il m'a soutenu et guidé dans mon travail et, m'a aidé à trouver des solutions pour avancer, et pour dépasser les barrières. Ainsi que pour l'inspiration et le temps qu'il a bien voulu me consacrer, je dis merci.

Je tiens également à adresser mes remerciements Monsieur Alaa Ghaith, professeur à l'Université Libanaise, et Monsieur Kamal Beydoun, Professeur associé à l'Université Libanaise ; respectivement Co-directeur et Encadreur de ce travail. Ils sont toujours montrés à l'écoute et très disponible tout au long de la réalisation de cette thèse.

Je continuerai en remerciant les membres de mon jury. Tout d'abord, Monsieur Ellatif Samhat, Professeur de l'université Libanaise pour avoir accepté de présider mon jury, J'exprime également ma gratitude à Messieurs Frédéric Kratz, professeur des Universités à l'INSA Centre Val de Loire, et Rida Khatoun, professeur associé à l'Institut Polytechnique de Paris, pour m'avoir fait le privilège d'être rapporteurs de ma thèse et surtout pour leurs critiques constructives. Enfin, je souhaiterais aussi remercier Messieurs, Youssef Laarouchi, gestionnaire de projet R&D du groupe Innovation & Recherche en Cybersécurité chez EDF, et

Mohamad Nassar, professeur associé à l'Université Américaine de Beyrouth, pour avoir évaluer scientifiquement mon travail de recherche et pour avoir pris part au jury. Vos remarques et vos conseils ont contribué à parfaire la version finale de ce manuscrit.

Je souhaite également exprimer mes remerciements à l'ensemble du personnel des services informatique et administratif du laboratoire G-SCOP pour avoir participé à mon intégration au sein du laboratoire, pour la gestion des missions.
Je remercie également l'ensemble administratif de l'Université Libanaise pour l'organisation de ma soutenance de thèse.

Je voudrais aussi remercier mes amis et coéquipiers, qui ont eu à supporter les hauts et les bas, Houssein Abdo, Mohamad Kaouk, Amine Awada. Aussi, au moment de conclure de cette thèse, je souhaite également remercier mes anciens collègues de l'université Libanaise Mohamad Harissa, Hussein Taha pour leur intérêt constant à mon travail. Je souhaiterais notamment dire merci à Mostafa Raed, mon frère à qui ma mère n'a pas donné naissance, pour son soutien tout au long de ma vie.

Je voudrais remercier tout particulièrement mon oncle Houssein Bazzi, qui a été une des causes principales de ce travail. Je voudrais remercier pareillement mon oncle Ali Bazzi, l'un des personnes les plus intéressés à terminer mes études supérieures.

Enfin, je dédie ce paragraphe à ma merveilleuse famille, mon père Imad, ma mère Iman, mes frères Ali, la personne inspirante, Ahmad, la personne pleine de vie, et Mostafa, la personne merveilleuse. Merci pour tout ce que vous avez fait pour moi, pour être toujours à mes côtés et pour vos soutiens.

À ma chérie, mon compagnon et mon amie au cours de mon long voyage. A toi mes remerciements mêlés aux couleurs de l'amour.

Vous m'avez apporté force et courage pour achever cette thèse et poursuivre vers de nouvelles aventures.

Mohamad-Houssein

# Table de matière

# Introduction Générale

La révolution technologique dans le domaine industriel a permis d'ajouter les capacités informatiques pour les systèmes industriels. Par conséquent, de nouveaux moyens de communication ont été amenés à partir des systèmes informatique (IT) vers les systèmes de contrôle industriels (ICSs). De nos jours, la plupart des ICSs sont connectés à des réseaux locaux ou à l'internet. La connexion au réseau informatique ajoute beaucoup d'avantages, tel que l'évolutivité, la rentabilité, et l'accès à distance.

Cette révolution a été accompagnée par des risques de sécurité liées aux vulnérabilités informatiques (toutes les vulnérabilités informatiques sont applicables à ces systèmes de contrôle). Les ICSs sont devenus donc facilement accessibles aux logiciels malveillants cherchant à en prendre le contrôle pour effectuer des actions dangereuses ou désactivantes pour les installations. Et contrairement aux systèmes informatiques, les conséquences de telles malveillances sur les systèmes industriels peuvent comporter des risques importants pour la santé et la sécurité des vies humaines et des graves dommages à l'environnement.

La prise en compte des aspects de sécurité pour ces systèmes est devenue indispensable. Les solutions de sécurité on a commencé par adopter les approches IT aux systèmes de contrôle tel que le chiffrement, l'utilisation des pares-feux, la surveillance des trames de réseau par des systèmes de détection d'intrusion (IDSs), etc.

Le traitement des problèmes de sécurité par une vision IT n'était pas suffisant, puisqu'il néglige une très importante spécificité des systèmes industriels, qui est la relation profonde entre le système informatique et le système physique. Il est donc important d'interpréter les informations à la lumière du comportement physique qu'elles représentent.

Le travail que nous présentons dans ce manuscrit propose d'apporter une contribution au domaine de la surveillance d'actes de malveillance dans les systèmes de contrôle-commande. Notre apport principal se situe dans la proposition d'une méthodologie permettant la

conception des systèmes de détection d'intrusions à base modèle pour les ICS. Cette conception commence par une spécification du système de contrôle et du comportement attendu du système physique à partir de la connaissance de l'ICS. Ensuite, les spécifications seront traduites en des règles d'IDS et en un module complémentaire permettant la prédiction de l'état du process. Enfin, les règles de l'IDS sont implémentées par un IDS standard, qui est aussi relié au module de prédiction de l'état de process.

Ce manuscrit s'articulera autour de 3 parties de la façon suivante :

La première partie présente d'une manière générale le cadre de l'étude ainsi que la problématique. Après avoir décrire et définir les systèmes de contrôle industriels, leurs vulnérabilités et leurs différences avec le domaine informatique, une présentation de l'état de l'art des approches de sécurité appliquées est menée afin de mettre en évidence les approches et techniques à retenir pour notre contribution. Les approches de détection d'intrusion à base de modèle pour les systèmes industriels semblaient avoir des résultats intéressants, surtout celle qui prend en considération le comportement du système physique. Sur cette base, nous exposons notre approche générale et proposons une approche permettant la conception des IDS à base modèle pour les ICSs.

La deuxième partie est consacrée à la présentation de notre approche sur le plan méthodologique et algorithmique. Dans un premier temps, nous proposons notre méthode pour générer un modèle permettant la spécification de l'ICS. La démarche proposée consiste à créer des modèles génériques pour l'ICS comprenant son architecture réseau, le comportement de la partie de contrôle et le comportement attendu du système physique à partir des spécifications de l'ICS et en utilisant des méthodes et des formalismes bien défini. Ces modèles sont traduits plus tard en des règles de détection d'intrusion qui peuvent être implémentées par un IDS standard, ainsi qu'un module de prédiction de l'état du système physique qui alimente l'IDS par des informations complémentaires qui sont nécessaires pour prendre une bonne décision sur la nature des événements observés.

La troisième et dernière partie, présente un exemple d'application basé sur des cas d'étude académiques et sur une plateforme hybride (où l'ICS est simulé par un réseau de machines

virtuelles qui utilisent des vraies communications réseau entre elles) et représentative d'un système de production industrielle. Après avoir présenté le cas d'étude**,** nous présentons différents scénarios d'attaque que nous avons réalisés sur le système. Une fois appliqués, nous présentons les comportements de nos algorithmes afin de tester et d'évaluer les capacités de détection des mécanismes développés.

# Chapitre 1 Contexte de l'étude

## 1.1 Motivation

De nos jours, les systèmes industriels présentent des risques de sécurité liés à leurs vulnérabilités informatiques. D'après l'équipe d'intervention d'urgence en systèmes industriels du laboratoire Kaspersky ("Threat landscape for industrial automation systems. H2 2018," 2019), les systèmes industriels répartis sur le monde entier continuent à être cible d'attaques. La Figure 1 présente la répartition géographique des attaques sur les systèmes d'automatisation industriels, où Vietnam, Algérie, Tunisie, Égypte, Indonésie et China (à l'exception de Hong Kong) étaient parmi les pays ayant les plus grands pourcentages d'ordinateurs ICS contenant des objets malveillants en S2 2018 dépassent 58%.

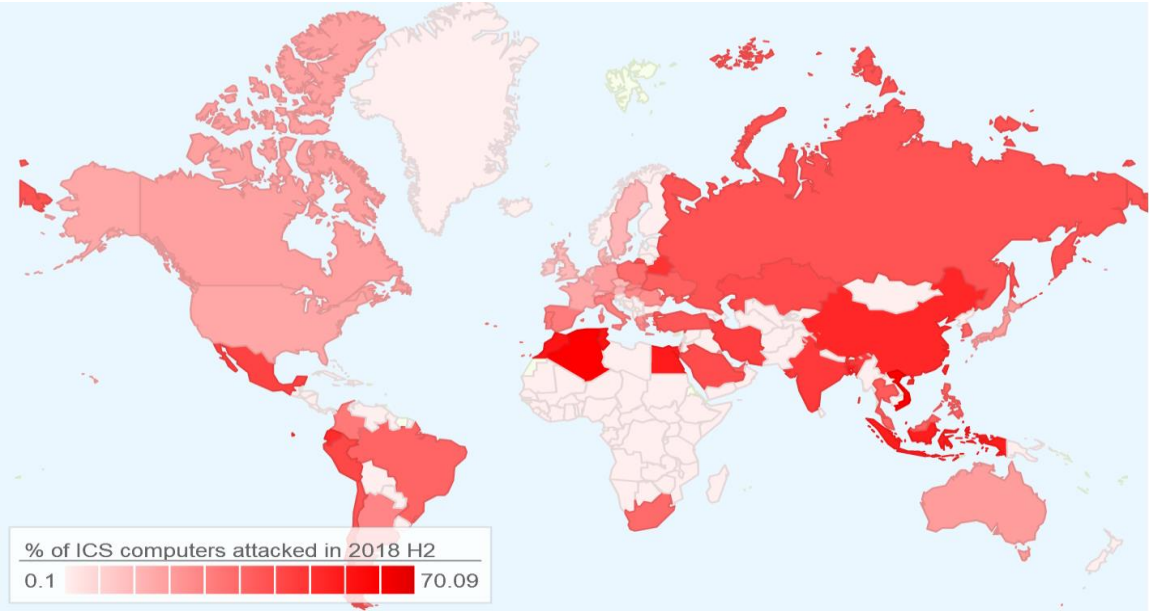



Figure 1: Répartition géographique des attaques sur les systèmes d'automatisation industriels

Le nombre de vulnérabilités sur les différents composants des ICS continue à évoluer. En effet, le site de l'ICS-CERT ("ICS-CERT Landing | CISA") a publié que les vulnérabilités identifiées en 2018 étaient 415, c'est-à-dire 93 plus que celle détectées en 2017. Ainsi, ces vulnérabilités sont réparties sur plusieurs secteurs tel que la fabrication critique, le secteur d'énergie, les systèmes d'approvisionnement de l'eau et d'autres secteurs comme c'est illustré dans la Figure 2.

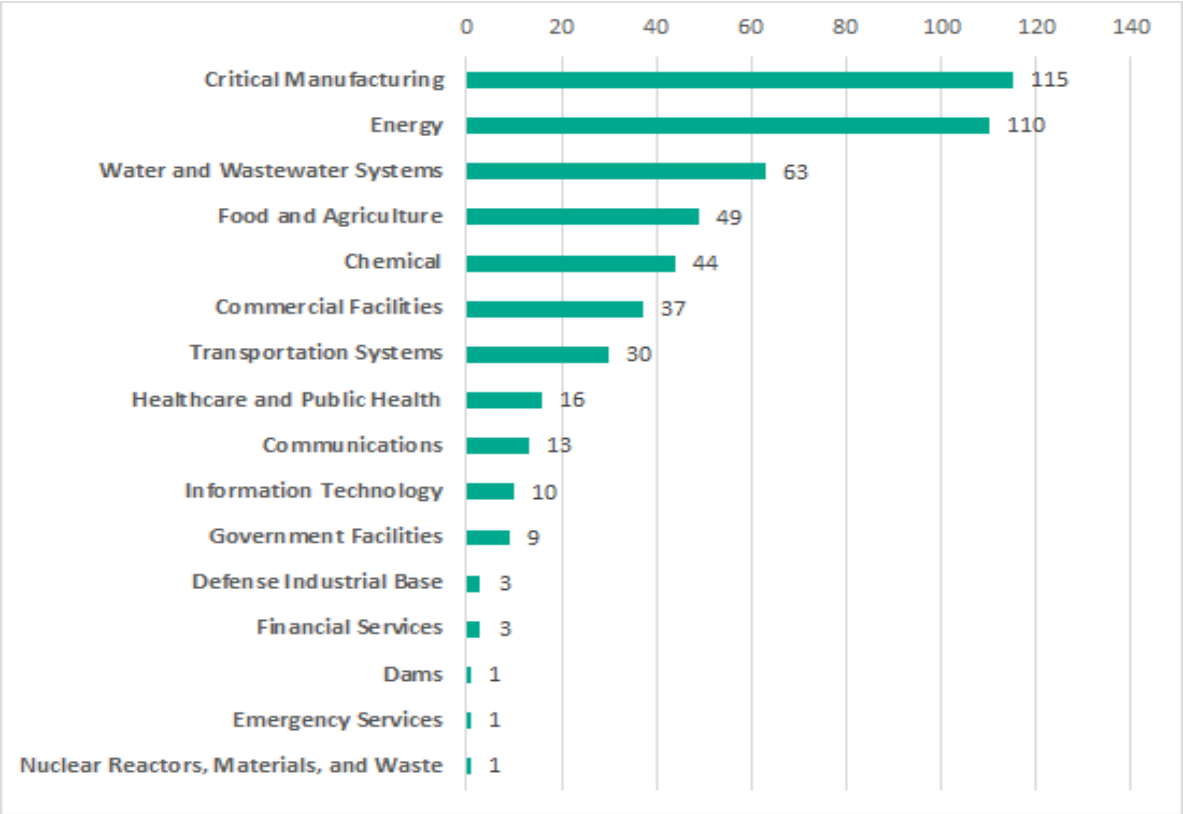



Figure 2: nombre de produits vulnérables utilisés dans différentes industries (selon la classification US ICS-CERT) publiées en 2018

Ceci montre que la sécurité des systèmes industriels est toujours un sujet d'actualité surtout en tenant compte des effets qui peuvent résulter suite à une attaque sur des tels systèmes qui commencent par une dégradation de production, mais peuvent atteindre des résultats catastrophiques sur la santé et la vie des personnels.

## 1.2 Système de contrôle industriel (ICS)

### 1.2.1 Introduction

Le système de contrôle industriel (ICS) est un terme général utilisé pour décrire l'intégration des logiciels et du matériel, ainsi que la connexion réseau pour atteindre un objectif industriel (Stouffer et al., 2015a). Un ICS est composé des composantes numériques et physiques qui interagissent à travers d'un réseau de communication, pour permettre la collaboration des éléments informatiques de contrôler et commander les entités physiques (Cardin et al., 2016). Très souvent, les éléments de l'ICS sont exploités par des logiciels spécifiques, comme les logiciels SCADA.

Un ICS est composé d'une part, d'un système de traitement de l'information qui ressemble à un système d'information classique et composé de postes de travail, de serveurs et d'équipements réseau et des systèmes de stockage et de sauvegarde, d'autre part, un ensemble d'équipement spécifiques permettant d'agir avec le monde physique par la réception des grandeurs mesurées (tel que la température, la pression ou la présence d'un objet), et l'application des commandes de contrôle (tel que l'ouverture d'une vanne, la mise en action d'un vérin ou l'ouverture et la fermeture d'un circuit). Ces équipements contiennent par exemple les automates programmables (PLC), les capteurs et les actionneurs.

L'architecture d'un système de contrôle industriel est illustrée sur la Figure 3. Pour désigner un ICS, on peut rencontrer plusieurs termes tel que DCS (Distributed Control System) qui est réservé aux systèmes provenant d'un fournisseur unique, SCADA (Supervisory Control and Data Acquisition) utilisé pour désigner les systèmes constitués d'appareils et de logiciels de différent fabricants et mise en place par un intégrateur, IACS (Industrial Automation and Control System) et ICS (Industrial Control System). Le terme « IACS » a été proposé par l'ISA dans les années 2000 et a été repris sous une forme simplifiée « ICS » dans le guide NIST 800-

82 (NIST, 2008) en 2008. Cette nouvelle dénomination rend progressivement obsolètes les termes « DCS » et « SCADA ».

Les systèmes de contrôle industriel sous-tendent l'infrastructure nationale critique et sont essentiels à la réussite des industries telles que : la télécommunication, la production alimentaire, la production chimique et pharmaceutique, la fabrication de composants et

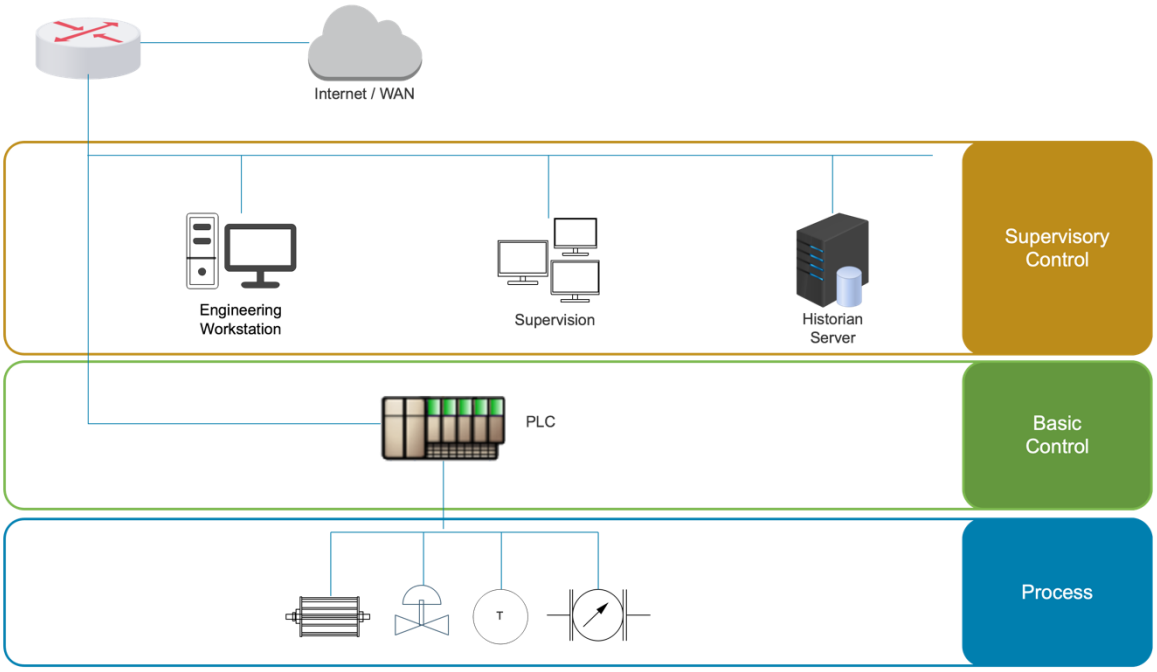



Figure 3: Système de contrôle industriel

produits finis, la production et distribution d'électricité, l'approvisionnement et traitement de l'eau et la production et approvisionnement en pétrole et gaz (Stouffer et al., 2015a).

### 1.2.2 Système de contrôle et d'acquisition de données (SCADA)

SCADA (*Supervisory Control And Data Acquisition*) est un système de télégestion permettant de contrôler des équipements d'automatisation et d'acquisition des données. Il désigne une catégorie de logiciels destinés à centraliser le pilotage de processus, la collecte et la présentation de données sur une interface homme-machine.

### 1.2.3 Système de contrôle distribué (DCS)

Un DCS, *Distributed Control System*, est un système de contrôle industriel automatisé répartie géographiquement, et connecté en réseau avec une unité centrale permettant de réaliser la supervision.

### 1.2.4 Automate programmable industriel

Un automate programmable ou bien Programmable Logic Controller (PLC) en anglais est un équipement essentiel dans l'automatisation d'un système. Les automates sont utilisés dans les systèmes SCADA et DCS en tant que composants de contrôle d'un système hiérarchique global afin de permettre une gestion locale des processus grâce au contrôle par retour. Un automate permet de contrôler un système physique, en appliquant soit un fonctionnement séquentiel prédéterminé, soit par esclavage ou bien en régulant des grandeurs selon une consigne fixée.

Généralement, un PLC est composé d'une mémoire, d'une unité de traitement (processeur), d'une unité d'alimentation électrique, d'interfaces d'entrée/sortie et d'une interface de communication, comme c'est illustré sur la Figure 4.

Un PLC exécute un algorithme bien déterminé pour contrôler et gérer la production. Cet algorithme est codé dans un langage de programmation de PLC tel qu'il est spécifié dans la norme IEC-61131-3. Il fonctionne selon un cycle régulier, pendant lequel il :

- Lit des entrées dans la mémoire d'entrée et les copie dans la mémoire d'entrées,
- Exécute le programme,
- Exécute les tâches de diagnostic et de communication,
- Met à jour les résultats dans la mémoire de sortie et envoie ces valeurs au dispositif physique.

La mémoire du PLC contient le système d'exploitation et le programme de contrôle. L'unité de centrale de traitement est à base de microprocesseur. Cette unité lit les signaux d'entrées et

écrit les signaux de sortie. L'interface de communication permet au PLC de communiquer avec les autres automates, le système de supervision (SCADA) et la station de programmation.

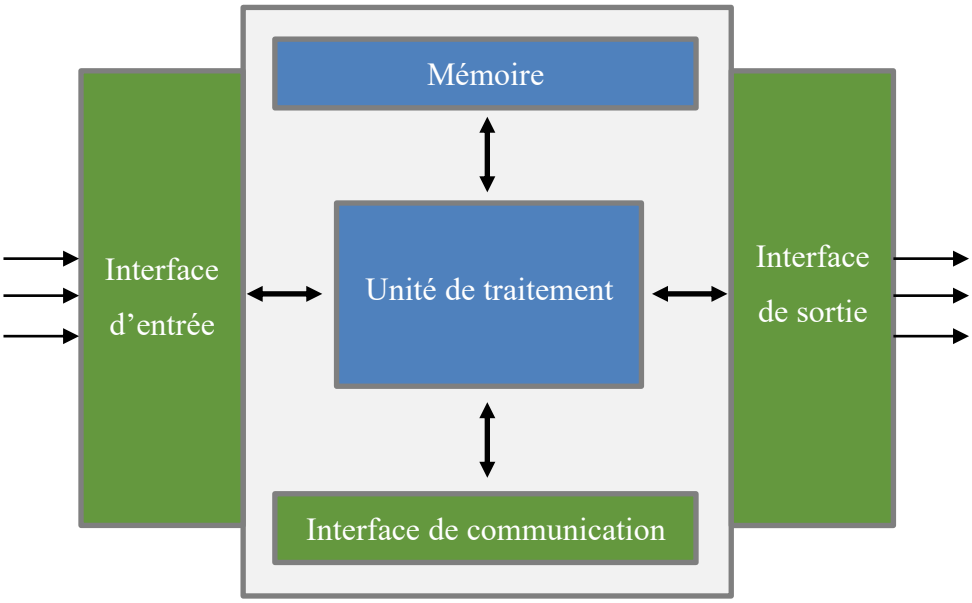



Figure 4: Architecture du PLC

La norme IEC-61131-3 ("IEC 61131–3," 2003, p. 20033) définit cinq langages de programmation pour les PLCs, qui sont :

- **Langage Ladder ou bien Ladder Diagram (LD)**: est une représentation graphique d'équation booléennes combinant des contacts (en entrée) et des relais (en sortie). Il ressemble aux schémas électriques, et permet la manipulation des données booléennes, à l'aide de symboles graphiques.
- **Boîte fonctionnelle** ou bien **Functional Block Diagram (FBD)** en anglais. C'est un langage graphique, permettant la construction d'équations complexes à partir des opérateurs standards, de fonctions ou de blocs fonctionnels.
- **Liste d'instruction ou bien Instruction List (IL)**, qui est langage textuel de bas niveau, proche au langage assembleur.
- **Texte structuré Structured Text (ST)**, qui est un langage textuel de haut niveau dédié aux applications d'automatisation. Il est généralement utilisé pour décrire des procédures complexes, difficilement modélisables avec les langages graphiques.

- **Diagramme fonctionnel séquentiel ou bien Sequential functional chart** (**SFC**): est un langage graphique dérivé de GRAFCET et utilisé pour décrire les opérations séquentielles. Ce langage représente le procédé comme une suite connue d'étapes (états stables), reliées entre elles par des transitions, où :
    - Une condition booléenne est attachée à chaque transition.
    - Les actions dans les étapes sont décrites avec les langages **ST**, **IL**, **LD**, ou **FBD**.

### 1.2.5 Particularité des systèmes industriels

Un certain nombre de modèles a été proposé pour structurer et organiser un ICS d'une façon hiérarchique permettant de simplifier un peu la réalité. Le modèle de Purdue (PERA) (Williams, 1994), le modèle IEC-62443, et le modèle CPwE (Converged Plantwide EtherNet) ("CPwE," 2013).

Le modèle Purdue présenté sur la Figure 5 est le premier de ces modèles. Il structure l'architecture de l'ICS en six niveaux :

- Niveau 0 (processus physique) : contient les systèmes physiques utilisés pour la production et une grande variété de capteurs et d'actionneurs impliqués dans le processus de fabrication de base. Ces périphériques prennent en charge les périphériques de contrôle du niveau 1 du modèle logique et communiquent leurs statues.
- Niveau 1 (contrôle de base): correspond au contrôle local ou de base. Ce niveau se compose de contrôleurs qui dirigent et manipulent le processus de fabrication tel que les automates programmables (PLC) et les unités terminales distantes (RTU) dont la fonction principale consiste à assurer une interface avec les dispositifs de niveau 0.
- Niveau 2 (contrôle de supervision): concerne les serveurs et les stations opérateurs utilisés pour observer et contrôler à distance les appareils de terrain. On y trouve dans ce niveau les interfaces hommes-machines (IHM) et les postes de travail en salle de contrôle.

- Niveau 3 (gestion des opérations) : représente le plus haut niveau de l'IACS. Il est le responsable de la gestion de la production et de ses opérations connexes pour produire les produits souhaités. Ce niveau comprend les systèmes de gestion des opérations MOM (Manufacturing Operation Management), les systèmes d'exécution de la fabrication MES (Manufacturing Execution System) et d'autre composants tel que les bases de données d'historiques et les systèmes de gestion de qualité à l'échelle du site.
- Niveau 4 (Planification des activités et logistique du site) : souvent considéré comme une extension du niveau 5, regroupe les systèmes informatiques traitant des fonctions de reporting, de planification, de gestion des stocks, de planification des capacités, de gestion des opérations et de la maintenance, des services de courrier électronique, téléphonique et d'impression. Les services, systèmes et applications des niveaux 4 et 5 sont normalement gérés et exploités par le service informatique (Cisco et Rockwell Automation, 2011).
- Niveau 5 (Système d'entreprise) : est l'emplacement des fonctions et des systèmes qui gèrent les activités liées à la fabrication et à la transformation. On y trouve l'ERP (Enterprise Resource Planning) qui fournit au MES une liste d'ordre de fabrication.

Les autres modèles se basent sur le modèle de Purdue et créent leurs propres adaptations ; d'une part, le modèle IEC-62443 fait explicitement apparaître les systèmes instrumentés de sécurité (SIS) et de protection dans le niveau 1 et fusionne les niveaux 4 et 5 en le niveau « systèmes d'entreprise ». D'autre part, le modèle CPwE ajoute une zone DMZ entre le niveau 3 et 4 afin de séparer les réseaux industriels, appelé aussi « Operational Technology » (OT), et le réseau informatique de l'entreprise (IT).

Dans la suite, nous référons sur ce modèle pour analyser les risques et réaliser les modèles de l'ICS.

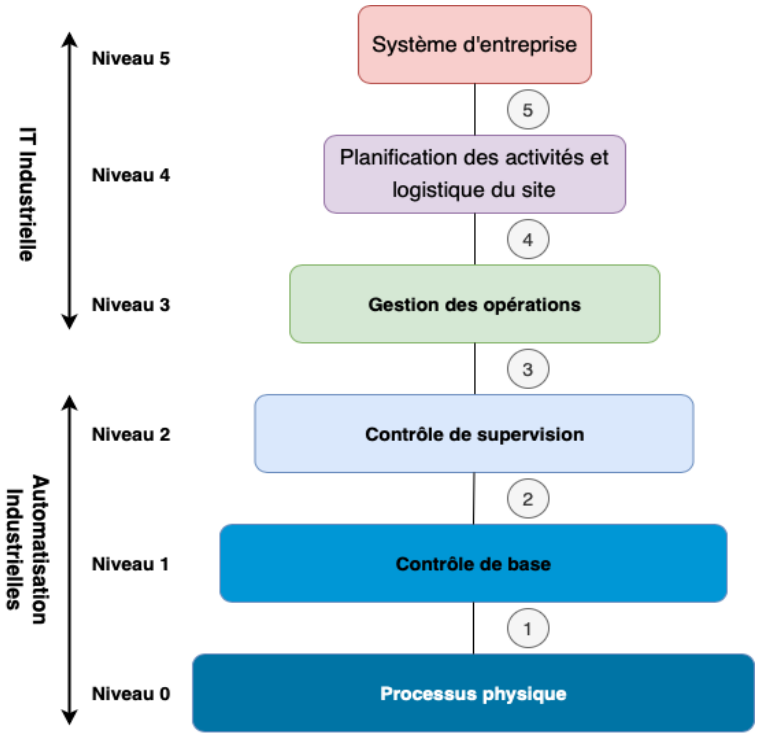



Figure 5: Modèle de Purdue

## 1.3 ICS et sécurité

### 1.3.1 Introduction

Aujourd'hui, la plupart des systèmes de contrôle industriel (ICSs) et de nombreux systèmes embarqués sont connectés en réseau local ou même via Internet. L'utilisation de la connectivité Internet offre l'évolutivité, la rentabilité, l'accès à distance et de nombreux autres avantages (Lu et al., 2014). Cependant, ces systèmes n'ont jamais été conçus en tenant compte de la connectivité du réseau et de la sécurité (Zamaï et al., 2007). La protection de ces systèmes a été assurée par l'isolement en utilisant des standards propriétaires (Stouffer et al., 2015a). Le contrôle de supervision et l'acquisition de données (SCADA) est l'un de ces systèmes. Il est utilisé dans la plupart des processus industriels (Horkan, 2015). Depuis longtemps, ICS / SCADA était une zone qui s'appuyait sur différents périphériques embarqués et des communications en clair comme Modbus / TCP, sans prendre en compte l'approche de sécurité

qui la rendait vulnérable aux différents types d'attaques et devient une cible de menaces cybernétiques. Ces systèmes, par conséquent, présentent des risques de sécurité liés à leurs vulnérabilités informatiques et sont relativement facilement accessibles aux logiciels malveillants qui cherchent à prendre le contrôle de celui-ci pour effectuer des actions dangereuses ou invalides sur les installations (Lu et al., 2014) (Zhu and Sastry, 2010).

Un ICS présente de nombreuses caractéristiques qui diffèrent des systèmes informatiques traditionnels, y compris les différents risques et priorités. Certains d'entre eux incluent des risques importants pour la santé et la sécurité des vies humaines, des dommages sérieux à l'environnement et des problèmes financiers tels que les pertes de production et les effets négatifs sur l'économie d'une nation. ICS a des exigences de performance et de fiabilité différentes, et utilise également des applications qui peuvent être considérées comme non-conventionnels dans un environnement de réseau informatique typique.

Les protections de sécurité doivent être mises en œuvre de manière à maintenir l'intégrité du système pendant les opérations normales ainsi que pendant les périodes de cyber-attaque.

Il existe de nombreux exemples d'accidents survenus sur des systèmes physiques causés par des logiciels malveillants, comme Stuxnet, qui a été utilisé pour détruire les installations nucléaires iraniennes (Stouffer et al., 2015a). Stuxnet est créé pour infecter PC utilisant le système d'opération Windows, puis pour cibler un contrôleur spécifique de Siemens connecté au PC infecté via Ethernet, bus Pro ou un lien de communication propriétaire de Siemens. Quand il a trouvé le contrôleur, Stuxnet a chargé un code malveillant et a causé la destruction.

Même si les systèmes de contrôle et les systèmes informatiques présentent des vulnérabilités partagées, ils ont des conséquences différentes. Par exemple l'attaque déni de service (DoS) sur un système informatique rend le service indisponible qui pourrait coûter à l'entreprise, mais elle peut avoir des conséquences désastreuses sur l'économie et la vie dans un système industriel (Han et al., 2014).

### 1.3.2 Vulnérabilités des protocoles de communications industriels

Les protocoles de communication industriels ont été principalement conçus en tenant compte de l'efficacité des systèmes intégrés sans prendre en considération la sécurité des échanges. La sécurité a été traditionnellement assurée par l'isolation. Cependant, avec l'évolution des ICSs, ils sont devenus de plus en plus exposés. Ce qui a permis l'apparition des risques de sécurité pour les ICSs causés par les vulnérabilités des protocoles utilisés.

Les communications utilisant le protocole **Modbus** ne présentent aucun mécanisme de chiffrement ou d'authentification. Par conséquent, les attaquants puissent analyser les adresses Modbus et les codes de fonction, puis voler ou altérer les données de communication. L'absence de mécanismes de cryptage et d'authentification rend la communication Modbus efficace, mais peut également entraîner de graves problèmes de sécurité.

De même, le protocole **DNP3** (Distributed Network Protocol) n'utilise toujours pas de mécanisme d'autorisation ou de cryptage et est vulnérable aux attaques d'interception (MitM).

D'après cette analyse sur les protocoles industriels utilisés, nous trouvons un manque de considération de la sécurité dans la conception des protocoles industriels, et donc il existe un grand risque de sécurité provenant des protocoles industriels. Et par conséquent, la technologie de détection d'intrusion basée sur l'analyse de protocole émerge.

Outre les protocoles publics, certains protocoles industriels propriétaires ont été utilisés pour développer des techniques IDS dans des industries spécifiques. Hong et al. ont analysé des systèmes automatiques dans les sous-stations d'un réseau intelligent et ont détecté des anomalies ou des comportements malveillants dans des messages de multidiffusion basés sur les normes IEC 61850 (par exemple, la technologie GOOSE (Generic Orient-Orient Substation Event) et la technologie Sample Value (SV)) publié par la IEC en 2004. Cette méthode, basée sur des spécifications de protocole propriétaires, permet de détecter efficacement les attaques malveillantes telles que la falsification de paquets, les attaques répétées et le déni de service (DoS).

### 1.3.3 Vulnérabilités ICS et attaques contre ICS

Les ICS présentent de nombreuses vulnérabilités du point de vue de la sécurité (Sicard et al., 2019). La cause de ces vulnérabilités comprend des pare-feu mal configurés, des failles logicielles, une authentification faible ou un manque de mécanismes d'authentification, l'utilisation de protocoles non-fiables tels que Modbus et DNP3 sur TCP / IP (Flaus, 2019). McLaughlin et al. (McLaughlin et al., 2016) regroupe les vulnérabilités ICS en cinq couches : matériel, firmware, logiciel, réseau et processus ICS. Une discussion des caractéristiques de chaque couche, des attaques, des menaces et des vulnérabilités possibles est expliquée dans (Keliris et al., 2016).

Ces vulnérabilités peuvent être exploitées par des pirates informatiques qui cherchent à endommager physiquement le système et l'environnement. Différents types d'attaques pourraient être appliqués, notamment :

- Divulgation d'informations et exécution de commandes non autorisées en raison d'un manque d'authentification.
- Attaques DoS (déni de service) ou DDoS (déni de service distribué) rendant les appareils non-disponibles.
- Sollicitation excessive des équipements en répétant des combinaisons spéciales de commandes de contrôle causant des dommages à l'équipement ICS.
- Attaques Man-in-the-Middle (MitM) qui interceptent et/ou injectent des données et rejettent des attaques qui retournent une séquence déjà envoyée.
- Attaques séquentielles qui utilisent des commandes légitimes dans un ordre différent pour conduire progressivement le système à un état critique.
- Attaques temporelles qui envoient des commandes/rapports violant les contraintes de temps (périodicité, timing).

Certaines de ces attaques sont inhérentes à l'informatique et peuvent être détectées par des solutions de sécurité avec des détails de haut niveau sur le système à sécuriser. D'autres attaques

sont plus avancées et nécessitent une connaissance approfondie de la loi de contrôle et du dispositif du système.

### 1.3.4 Différences entre ICS et systèmes informatiques

Comme cela à déjà été expliqué précédemment (dans le paragraphe 1.1), l'introduction des capacités informatiques sur les systèmes de contrôles a été la cause d'un large nombre de vulnérabilités. Cependant, les systèmes informatiques et les systèmes de contrôles industriels possèdent des caractéristiques différentes, ce qui nécessite des mécanismes de sécurité et de surveillances différent les uns des autres. Nous procédons en expliquant les différences entre ces derniers systèmes en termes de cybersécurité répartie sur trois aspects :

- Les fonctionnalités attendues.
- La technologie utilisée.
- Le cycle de vie du système.

#### 1.3.4.1 Différences dans les fonctionnalités attendues

D'abord, du côté des **fonctionnalités attendues**, les systèmes de contrôle industriels sont généralement conçus pour appliquer des tâches spécifiques, optimisés pour une faible consommation d'énergie avec un coût minimal.

- Les **fonctionnalités principales** d'un ICS sont d'une part de piloter un processus physique selon le mode opératoire prévu, et d'autre part, d'assurer des fonctions de sécurité en cas de fonctionnement anormal. Cependant, un système informatique ne manipule que des données n'évolue que dans un monde digital (Flaus, 2019).
- **Un ICS doit généralement rester en opération de façon continue,** il est donc conçu pour un fonctionnement continu de 24h/7j. Ainsi, les niveaux 1 et 0 d'un ICS sont soumis à des contraintes temps-réels (de l'ordre de milliseconde). Par conséquent, il apparaît clairement que la disponibilité ; c'est-à-dire la capacité à accomplir une fonction à un moment donné, est considéré comme une caractéristique essentielle des ICS. Ceci est différent du cas des systèmes IT, qui s'intéressent moins aux contraintes

temps réels. Par exemple, le contrôle du freinage d'un train est plus important en termes des contraintes de temps réel que d'avoir une réponse assez rapide d'un serveur mail ou d'un serveur web.

#### 1.3.4.2 Différences dans les technologies utilisées

Du côté **aspects technologiques**, les ICS diffèrent des systèmes IT sur les points suivants :

- Au niveau des **ressources informatiques**, les systèmes IT sont composés essentiellement de postes informatiques, de serveurs et d'équipements relativement récents. Tandis que les ICS s'appuient sur des équipements qui peuvent être assez anciens. Par exemple, les différents équipements de terrain sont essentiellement des automates et des systèmes embarquées. Ceux-ci sont conçus pour assurer un fonctionnement temps réel efficace, mais en utilisant des ressources limitées en mémoire, capacité de calcul et en consommation d'énergie. Ces limites en ressources sont dues à une conception ancienne des ICSs, ce qui a causé un manque d'intelligence pour ces composantes, et par conséquent, une faible capacité à assurer les fonctions de sécurité tel que l'authentification des nœuds communicants et le chiffrement des données communiquées.
- **Durée de vie des composants** : pour un ICS, la durée de vie est identique à celle de l'installation et peut atteindre jusqu'à vingt ans (Stouffer et al., 2015a). Tandis que dans les systèmes IT, la durée de vie des composantes est généralement de 3 à 5 ans.
- Au niveau de la **communication**, les systèmes IT utilisent des protocoles de communication très standardisés tel que les protocoles TCP/IP et UDP (Cheminod et al., 2012) en utilisant des réseaux câblés avec certaines fonctionnalités sans fil localisées. En revanche, les ICSs utilisent beaucoup des protocoles propriétaires tel que Profibus (Siemens), CAN (automobile) et des protocoles standards tel que Modbus sur TCP/IP. En plus, plusieurs types de supports de communication sont utilisés par les ICSs, y compris les réseaux filaires et sans fil (radio et satellite) dédiés (Stouffer et al., 2015a).

- **Hétérogénéité des technologies** : dans les systèmes IT, la compatibilité des composants entre eux est normalisée. On parle des composants homogènes et interopérables. Cependant, un ICS est un système complexe, et le plus souvent, les équipements proviennent de fournisseurs différents avec des technologies qui évoluent progressivement en fonction de l'évolution du système industriel et de sa durée de vie. Ce qui résulte en une superposition des technologies matérielles, logicielles et de communication, qui sont de plusieurs générations différentes, ce qui les rend complexes à gérer.

#### 1.3.4.3 Différences dans le management de la sécurité

Dans le côté sécurité, les ICSs et les systèmes informatiques diffèrent dans le but de la sécurité, et les mécanismes qui peuvent être utilisés pour l'assurer. Ceci est généralement due à cause de la différence des technologies utilisées.

- **Objectif de sécurité** : L'une de plus grandes différences entre la sécurité informatique des ICSs et des systèmes informatiques est l'objectif principal de chacune en matière de sécurité. Les systèmes informatiques ont pour objectif de sécurité principal la confidentialité des données ; c'est-à-dire, s'assurer que l'information n'est accessible qu'à ceux dont l'accès est autorisé. Cependant, la protection de l'information est importante pour un ICS, mais son objectif principal l'intégrité de son processus de production et la disponibilité de ses composants (Neitzel and Huba, 2014).
- **La gravité en cas d'attaque** : les systèmes informatiques gèrent des informations, ainsi une attaque dans un système informatique cause généralement des problèmes financiers à une entreprise (par exemple le serveur web arrête son fonctionnement et l'entreprise perd ses clients). Cependant, une attaque dans un ICS peut être catastrophique, surtout puisqu'ils gèrent des systèmes physiques et des infrastructures critiques. Par conséquent ces effets commencent par des dégradations de production et des problèmes financières, et peuvent entraîner de graves problèmes de santé, de sécurité et la vie des personnes (Galloway and Hancke, 2012)

- **Mise à jour des logiciels et des systèmes d'exploitation** : les systèmes informatiques sont conçus pour être utilisés avec des systèmes d'exploitation typiques. Ainsi, leurs mises à jour sont rendues simples avec la disponibilité d'outils de déploiement automatisés. En revanche, les ICSs utilisent des systèmes d'exploitation différents et éventuellement propriétaires. Les modifications et les mises à jour des logiciels doivent être effectuées avec soin, généralement par les éditeurs, à cause des algorithmes de contrôle spécialisés et peut-être du matériel et des logiciels modifiés impliqués. En plus, une mise à jour dans les ICSs nécessite un arrêt de production pour un certain temps afin d'appliquer les corrections nécessaires. En outre, beaucoup des ICSs utilisent des systèmes d'exploitation qui ne sont plus prise en charge tel que Windows NT et Windows XP. Ceci entraîne le fait que les ICS seront rarement mis à jour.

#### 1.3.4.4 Bilan de différences entre IT et ICS

Le Tableau 1 résume les différences entre les systèmes IT et les ICSs. Comme nous pouvons le constater, ces différences sont dues des conceptions différentes entre ces systèmes ce qui cause des conséquences importantes du point de vue de sécurité.

Tableau 1: Bilan de différences entre IT et ICS

| | Systèmes informatiques | Système industriels |
|---|---|---|
| **Les fonctionnalités attendues** | | |
| Fonctionnalités principales | Manipuler des données dans un monde digital. | Piloter un processus physique et assurer des fonctions de sécurité en cas de fonctionnement anormal. |
| Disponibilité et temps-réels | Importantes mais non critique | Critique |
| **Les technologies utilisées** | | |
| Ressources informatiques | Capacité importante en mémoire et capacité de calcul.<br>Composants plus intelligent. | Capacité limitée en mémoire et capacité de calcul.<br>Pas d'intelligence. |
| Durée de vie des composants | 3 à 5 ans. | Peut dépasser 20 ans. |

| | | |
|---|---|---|
| Communication | Protocoles standardisés. | Protocoles standardisés et nombreux protocoles propriétaires. |
| Hétérogénéité | Composants homogènes / interopérables. | Une superposition des technologies matériel, logiciels et de communications, qui sont de plusieurs générations différentes |
| **Management de la sécurité** | | |
| Objectif principal de sécurité | Confidentialité | Intégrité et la disponibilité |
| Gravité en cas d'attaque | Problèmes financières | Dégradation de production, problèmes financière, problèmes de santé et dangers sur la vie des personnels. |
| Mise à jour | Mise à jour régulière | Mise à jour rare ou bien n'existe pas. |

### 1.3.5 Vers la cybersécurité ICS

Il existe plusieurs normes et guides rédigés par des agences gouvernementales, notamment NERC CIP, IEEE 1613, International Society for Automation SP99, NIST Sp800, ANSSI, ENISA. Tous ces éléments tentent de guider vers une meilleure cybersécurité (Manson and Anderson, 2019) grâce à la formation et aux certifications des employés, en appliquant des techniques de sécurité telles que le chiffrement au stockage et aux communications de données ICS, et en appliquant une stratégie de défense en profondeur. Cette dernière comprend la sécurisation de l'architecture du réseau en mettant en œuvre une topologie de réseau multicouche, la séparation des réseaux ICS à l'aide de pare-feu, l'emploi de DMZ (Stouffer et al., 2015a).

Certaines approches s'appuient sur ces normes et guides et tentent d'étudier les vulnérabilités des CPS, Shin et al. (Shin et al., 2015) a développé un modèle de risque de cybersécurité qui peut trouver le point faible de la cybersécurité intégré deux modèles de cyber-analyse en utilisant le réseau bayésien. La première porte sur une évaluation qualitative de la cybersécurité

en vérifiant que les guides réglementaires et les normes sont respectés par les développeurs et les opérateurs du CPS. Alors que le second se concentre sur l'évaluation quantitative et qualitative des effets des vulnérabilités spécifiques au système et des mesures d'atténuation contre celles-ci sur la cybersécurité.

Chatterjee et al. (Chatterjee and Thekdi, 2020) a présenté une approche d'apprentissage itérative pour évaluer les propriétés dynamiques de vulnérabilité du système cyber-physique. Cette approche fait la distinction entre la vulnérabilité du sous-système et la sécurité globale du système. Enfin, il présente quatre mesures de posture du système : stabilité, anti fragilité, sécurité et dispersion qui fournissent collectivement une caractérisation holistique de la sécurité globale du système. Zang et al. (Zang et al., 2019) propose de révéler les mécanismes de propagation des défauts en modélisant la défaillance en cascade du CPS électrique. Sur la base de ce modèle, deux graphiques (graphes de propagation et d'attaque) sont construits pour analyser la vulnérabilité du CPS. Enfin, ces graphiques sont utilisés pour construire les indices de vulnérabilité du CPS électrique.

De nombreuses approches sont créées et sont spécifiques aux logiciels, firmware et matériels utilisés par les composants ICS afin de protéger ICS.

Sur la couche matérielle, Ren et al. (Ren et al., 2016) a proposé une approche basée sur l'apprentissage automatique pour sécuriser le port JTAG. Dans cette approche, l'exécution des instructions JTAG est surveillée et des machines à vecteurs de support (SVM) sont utilisées pour identifier les séquences d'instructions anormales.

Sur la couche firmware, Basnight et al. (Basnight et al., 2013) a proposé une méthode d'analyse de firmware basée sur l'ingénierie inverse pour identifier la faiblesse du contrôleur et proposer des méthodes défensives contre les attaques de modification de firmware. De même, Schuett et al. (Schuett et al., 2014) a utilisé une méthode d'ingénierie inverse pour exploiter l'automate pour les vulnérabilités et les portes dérobées et a suggéré des recommandations de conception pour atténuer la faiblesse potentielle dans le développement futur du firmware.

La couche logicielle est également étudiée pour assurer la protection ICS. McLaughlin et al. (McLaughlin et al., 2014) propose une approche pour protéger la couche logicielle. Dans cette approche, un vérificateur de sécurité fiable (TSV) est développé pour la vérification du code critique de sécurité avant son exécution sur des contrôleurs programmables.

D'autres approches seront expliquées dans le chapitre suivant, et proposent la surveillance des ICS en utilisant des systèmes de détection / prévention des intrusions qui réussies de contourner les mesures de cybersécurité citée ci-dessus.

Notre travail se concentre sur ce contexte et s'intéresse de la détection des attaques sur le réseau et la couche physique qui visent à endommager / dégrader le processus de production. Dans ce qui suit, nous présentons l'état de l'art sur les approches de détection des intrusions dans les systèmes industriels. Ensuite, nous proposons une solution pour modéliser les ICS et concevoir des IDS spécifiques pour les systèmes industriels. Un générateur de règles IDS basé sur un modèle de l'ICS est proposé, qui convertit un modèle système en règles IDS basées sur des anomalies.

## 1.4 Bilan du chapitre

Ce premier chapitre a permis d'avoir une vue d'ensemble des systèmes de contrôles industriels (ICS) et de leurs particularités en face des systèmes d'information classiques. Les systèmes de contrôle industriels présentent des risques de sécurité liées à leurs vulnérabilités informatique. Ces systèmes, répartis dans le monde entier, continuent d'être la cible d'attaques. Ce qui peut entraîner des pertes financières, mais aussi des risques graves sur la sécurité et la vie des personnels. Bien que ces systèmes partagent des vulnérabilités communes avec les systèmes informatiques, cependant, ces systèmes ont plus de contraintes que les systèmes informatiques. Ces contraintes rendent les approches proposées pour les systèmes d'information (IT) insuffisantes au contexte des systèmes industriel, avec une possibilité supplémentaire qui consiste à s'appuyer sur le comportement spécifique des ICS. Par conséquent, la nécessité d'avoir des solutions de sécurité plus adaptés pour le domaine industriel.

# Chapitre 2 État de l'art des approches de surveillance et de détection d'attaques dans les ICSs

## 2.1 Introduction

Ce chapitre est consacré à l'étude bibliographique de la cybersécurité dans les systèmes de contrôle-commande industriels. En premier lieu, nous identifions les différentes techniques présentes dans la littérature pour sécuriser les systèmes industriels. Ensuite, nous focalisons sur une technique particulière : les systèmes de détection d'intrusions (IDS). Ces systèmes étaient d'abord développés pour les systèmes d'informations (IT), puis sont progressivement adaptés pour répondre aux besoins des systèmes de contrôle industriels. Enfin, nous terminons cette étude avec les approches de détection à base de modèles de la partie opérative (OT).

## 2.2 Méthodes et outils de sécurisation des ICS

Historiquement, les systèmes de contrôle industriels ont été déployés en tant que plates-forme autonomes utilisant des protocoles de communication propriétaires aux fournisseurs séparés des réseaux informatiques externes. Cette isolation a permis d'éliminer les risques des cyber intrusions et donc la protection des ICS était assurée par le l'emplacement des composants ICS dans des zones physiquement sécurisées sans être connectés à des réseaux ou des systèmes informatiques. Cependant, les appareils IP largement disponibles et à faible coût remplacent désormais les solutions propriétaires. En plus, ICSs commence d'adopter des solutions informatiques pour promouvoir la connectivité des systèmes d'entreprise et les capacités d'accès à distance, qui sont conçus et mis en œuvre à l'aide d'ordinateurs, de systèmes d'exploitation (OS) et de protocoles réseau standard. Les ICSs commencent donc à ressembler à des systèmes informatiques. Ceci a causé une exposition des infrastructures critiques à des menaces externes et internes relatives aux vulnérabilités informatiques (Stouffer et al., 2015b).

Par conséquent, plusieurs propositions ont été créés afin de construire une architecture sécurisée en passant par un découpage en zones, filtrage et surveillances des flux entre les zones et surveillances des activités dans les zones (Flaus, 2019).

**Le concept de découpage en zones de sécurité** introduit par la norme IEC-62443 consiste à découper le réseau en zones de sécurité séparées par des pare-feu (firewalls), où une zone de sécurité est définie comme un regroupement physique et/ou logique ou de ressources ayant des exigences similaires en matière de sécurité. Ce partitionnement permet d'empêcher la propagation rapide des attaques (Fiaidhi and Gelogo, 2012). Dans une architecture ICS sécurisée il est recommandé de gérer l'accès à l'environnement ICS via une zone démilitarisée (DMZ) (Automation, 2011). Une zone démilitarisée (DMZ) est un sous-réseau situé entre un réseau interne et un réseau externe équipé par un ou deux pare-feu avec des politiques permettant d'autoriser les connexions depuis le réseau interne et le réseau externe vers cette zone, mais ne permettant pas de connexion depuis cette zone vers le réseau interne. Le but principal de cette zone est de fournir des services au réseau externe tout en protégeant le réseau interne ("Control System Security DMZ | CISA," n.d.). Cette notion est appliquée dans les réseaux de l'ICS afin d'isoler les systèmes et les données qui ont besoin d'être accessibles à partir du réseau IT et OT de l'ICS.

**Le pare-feu (firewall)** : est un dispositif de protection physique et/ou logiciel qui permet d'appliquer la politique de partitionnement. Il s'agit d'un séparateur de zones permettant de filtrer les paquets entrants et / ou sortant d'une zone (ou un réseau) en se basant sur des règles afin de s'assurer la politique réseau (Maheshwari and Dagale, 2018). Il permet ainsi de réaliser un monitoring des paquets et une journalisation. Ces règles peuvent par exemple limiter le trafic entrant vers des postes internes spécifiques et / ou provenant d'une (ou plusieurs) adresses spécifiques tel que les postes de maintenances (en se basant sur le champ IP et port). Mais aussi des règles dynamiques basées sur l'étude des séquences, ou bien applicative orienté compréhension des données transportées de l'application (niveau application du couche OSI) (Nurika et al., 2012).

Dans le même esprit, la **diode de données** est un dispositif de sécurité informatique qui restreint la communication entre deux ordinateurs ou deux réseaux afin que les données ne puissent pas être transmises que dans un seul sens (Jeon and Na, 2016). Ceci permet à un réseau informatique plus sensible de recevoir des données directement à partir d'une source moins sécurisée tout en interdisant la transmission de données dans le sens opposé (Stevens and Pope, 1999).

Enfin, les **systèmes de détection d'intrusions (IDSs)** permettent de surveiller les activités dans une zone. Cette technologie a été développée pour sécuriser les systèmes et les architectures IT, et s'est adapté pour répondre aux besoins des systèmes industriels (Zhu et al., 2011). Cette technologie sera présentée en détail dans la suite et elle constitue la base de notre approche de sécurité appliquée pour les systèmes industriels.

## 2.3 Systèmes de détection d'intrusion dans les ICSs

Un système de détection des intrusions (IDS) est un élément clé de la stratégie de défense en profondeur pour les systèmes d'information. La détection d'intrusion consiste à surveiller les événements survenant dans un système ou réseau informatique et les analyser afin de détecter les éventuels incidents qui constituent des failles de sécurité, de violation de règles ou d'autres problèmes susceptibles de compromettre le système d'information ou l'installation. Dans un premier temps, nous détaillons cette technologie, puis nous identifions les travaux utilisant cette technique pour développer des approches sécurisant les ICSs. Ces dernières sont fortement basées sur l'analyse de réseau de communication.

### 2.3.1 Généralité sur les IDS

Le système de détection d'intrusion (IDS) est une technique efficace pour détecter les attaquants lorsqu'ils arrivent à contourner les mesures de sécurité. Il peut détecter les activités malveillantes ou les violations des règles en surveillant les trafics du réseau ou sur les hôtes du système. Le cadre d'utilisation de cette technique a été pour la première fois énoncé par

(Denning, 1987) et de nos jours, il sont utilisés pour sécuriser les systèmes industriels. Un IDS est normalement un périphérique de veille ou un logiciel de troisième partie qui ne demandera pas beaucoup de modifications au système actuel. Il a été adapté aux systèmes à ressources limitées pour protéger leur sécurité réseau (Axelsson, 2000).

Les systèmes de détection d'intrusions sont composés de :

- La source de données qui doit être utilisée par l'IDPS pour faire l'analyse, tel que les paquets réseaux et les logs des hôtes.
- Le moteur d'analyse : ce composant prend les informations de la source de données, et vérifie l'existence des symptômes d'attaques, de violations de politique ou bien de présence d'anomalie dans le système.
- Le responsable de réponse : qui agit lors de la détection d'une incertitude (possible intrusion), en informant une personne, une autre machine, ou bien en faisant un certain processus comme réponse à l'éventuelle intrusion. Deux types de réponse sont applicables suite à la détection d'intrusion :
    - Une réponse passive où les systèmes de surveillance répondent en notifiant l'autorité approprié (généré un alerte), mais ne cherchent pas à régler ou atténuer les problèmes. Dans ce cas on parle d'un système de détection d'intrusion.
    - Une réponse active. Au contraire des systèmes passifs, les systèmes actifs cherchent à réagir contre l'attaque soit en exerçant des contrôles sur le système attaqué par exemple en modifiant l'état du système attaqué pour atténuer l'effet de l'attaque ou bien en exerçant leurs contrôles sur le système d'attaque pour tenter d'éliminer la plateforme d'opération de l'attaquant.

### 2.3.2 Les différents types d'IDS

Les systèmes de détection d'intrusion peuvent être classifiés selon leurs sources de données. Les classifications les plus communes des IDSs sont : IDS basés sur le réseau (NIDS) et IDS basés sur les hôtes :

- Un système de détection d'intrusion basé sur le réseau (*Network-based Intrusion Detection System*, NIDS) utilise le réseau comme source de donnée. La Figure 6 illustre un exemple d'implémentation d'un NIDS dans un réseau IT.
- Un système de détection d'intrusion basé sur l'hôte (*Host-based Intrusion Detection System*, HIDS) est installé sur les postes et surveille les activités du poste. La Figure 7 illustre un exemple d'implémentation d'un NIDS dans un réseau IT.

Un NIDS utilise des sondes déployées sur le réseau à surveiller pour collecter et analyser les données. Il surveille le réseau et cherche des actions hostiles ou bien des actions causant une violation des politiques d'utilisation employées. Un NIDS possède les avantages suivants :

- Couverture de détection d'intrusion en nécessitant moins de ressources systèmes.
- Possibilité de détection un pirate avant qu'il ne soit en mesure de faire une intrusion non autorisée.
- Indépendant des systèmes d'exploitation.
- Un coût inférieur de déploiement, de maintenance et de mise à jour que ceux d'un HIDS.

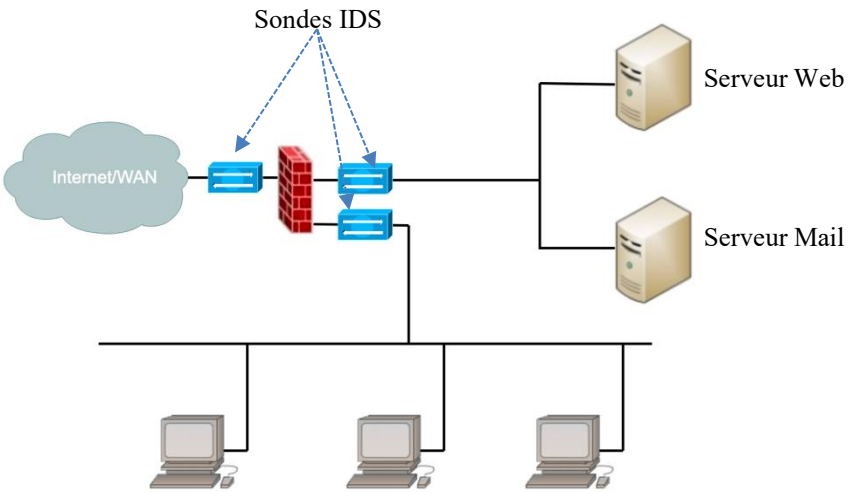



Figure 6: Système de detection d'intrusion basé sur le réseau (NIDS)

Cependant, un NIDS possède des inconvénients suivants :

- Nécessité de disposition d'une capacité de traitement relativement importante pour ne pas ralentir le fonctionnement.
- Inefficace lorsque le trafic est chiffré.

Les systèmes de détection d'intrusion peuvent être déployés sur les hôtes elles-mêmes. Cette implémentation est appelée HIDS. Les HID examinent des actions spécifiques basées sur l'hôte, telles que les applications utilisées, les fichiers en cours d'accès et les informations contenues dans les fichiers journaux. Ils examinent ainsi le trafic réseau entrant ou sortant d'un hôte spécifique.

Même que ces systèmes sont intéressants et peuvent avoir une excellente visibilité de l'état du système, mais ils sont lourds à déployer. Ainsi, il existe nombreux équipements sur lesquels déployer un HIDS n'est pas possible. Par conséquent, les HIDS sont considérés moins adaptés aux ICS et les NIDS sont davantage utilisés dans la littérature en raison des ressources limitées en mémoire, en CPU et en consommation d'énergie pour les équipements ICS.

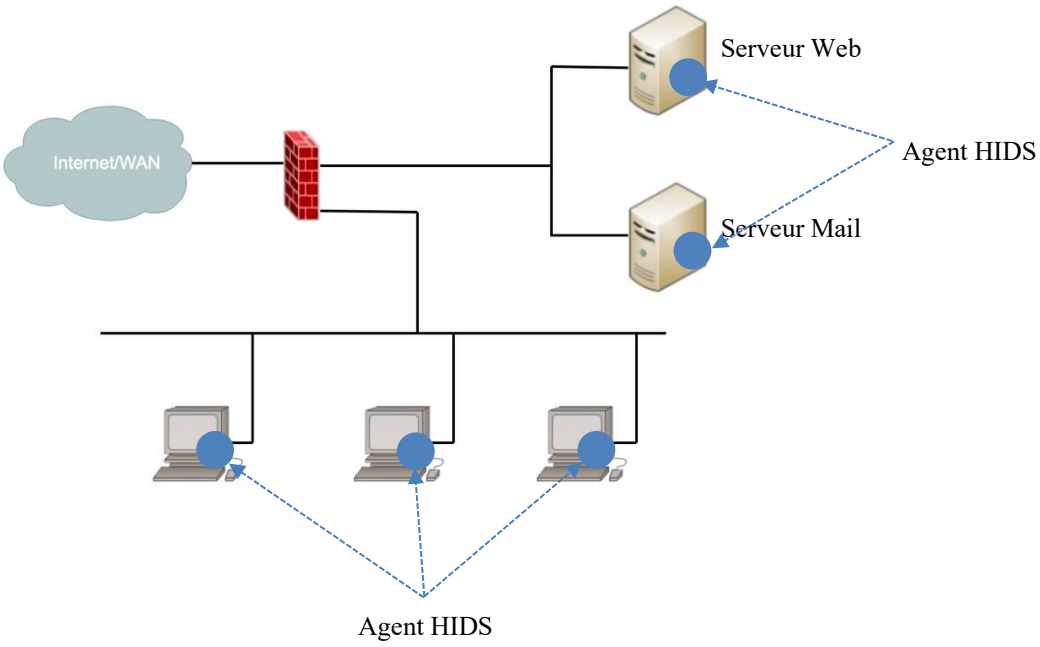



Figure 7: HIDS

### 2.3.3 Les méthodes communes de détection d'intrusions

Deux grandes catégories peuvent être identifiées pour détecter les intrusions dans l'environnement protégé (Evangelista, 2004). La première se base sur la connaissance des attaques (ou bien les comportements anormaux), et ici, on parle d'une détection à base de signature d'attaque. D'autres IDS se basent sur la connaissance (ou bien sur un modèle) du comportement normal et cherchent une déviation du comportement de référence, et ici, on parle d'une détection à base d'anomalie.

#### 2.3.3.1 La détection à base de signature

Les systèmes de détection d'intrusion à base de signature, également connus sous le nom d'utilisation abusive, détectent les attaques connues en s'appuyant sur une définition précise du comportement malveillant (Kumar and Spafford, 1994). Il s'appuie sur une base de données contenant la signature d'attaques connues et les compare aux données surveillées (également appelées événements). Les événements qui correspondent à une signature sont considérés comme des attaques lorsque d'autres événements sont considérés comme légitimes.

Une signature est un modèle qui correspond à une menace connue. La détection par signature est le processus consistant à comparer les signatures aux événements observés afin d'identifier les incidents possibles.

Selon (Ding et al., 2009) le principal avantage des systèmes basés sur une mauvaise utilisation est qu'ils produisent généralement très peu de faux positifs. Les **avantages** de la détection par signature sont :

- Méthode de détection la plus simple car elle compare simplement l'unité d'activité actuelle, telle qu'un paquet ou une entrée de journal, à une liste de signatures utilisant des opérations de comparaison de chaînes.
- Très efficace pour détecter les menaces connues (Ding et al., 2009).

Le principal inconvénient de la détection par signature est qu'il est inefficace pour détecter les menaces que ne sont pas encore connues et par conséquent, les IDS se basant sur les signatures des attaques ne peuvent par détecter les attaques du jour zéro (zero-day attacks) (Hu et al., 2018).

#### 2.3.3.2 La détection à base d'anomalie

Un système de détection/prévention d'intrusion à base d'anomalie se base sur des modèles représentant le comportement normal des utilisateurs, hôtes, connexions réseaux, du système physique ou du système de pilotage pour vérifier le comportement actuel du système surveillé. Par conséquent, un écart important entre le comportement actuel et le modèle de comportement normal est considéré comme anomalie.

Du point de vue opérationnel, les éléments suivants sont nécessaires pour suivre et analyser le système à surveiller (Flaus, 2019):

1. Choisir les éléments caractéristiques de l'activité. Le modèle de trafic du réseau, le comportement du protocole, les indicateurs d'activités des postes et le comportement du système physique et du système de pilotage sont des exemples des éléments caractéristiques.
2. Un langage formel permettant de créer un modèle du comportement acceptable du système à surveiller. Il existe deux façons pour créer ce modèle :
    a. Spécification par l'analyste.
    b. Spécification par apprentissage automatique. Dans ce cas, on a besoin de choisir la méthode d'apprentissage, qui peut être une approche statistique ou à base de deep Learning par exemple.

## 2.4 Approches de détection des intrusions dans les ICS

Dans la littérature, plusieurs approches se distinguent pour la détection des intrusions dans le domaine des ICSs. Ce manuscrit regroupe les approches de détection par le suivant :

- Les approches basées sur la vérification syntaxique et grammaticale des protocoles de communication industriels.
- Les approches de détection basées sur l'analyse de réseau de communication, et peuvent être regroupées en deux sous-catégories :
    - Les approches basées sur la spécification des propriétés du réseau.
    - Les approches basées sur l'apprentissage automatique du trafic.

Certains travaux se basent sur la vérification syntaxique et grammaticale des protocoles de communication industriels. L'approche créée par (Cheung et al., 2007) se base sur la spécification du protocole Modbus pour détecter les intrusions sur les ICS. Cette spécification concerne les codes fonctions, exceptions ou identificateurs de protocoles. Ensuite, Des règles pour l'IDS Snort sont développées pour détecter des violations des critères simples (qui ne nécessitent pas une détection répartie sur plusieurs paquets) tel que les code de fonctions invalide (ne sont pas dans la liste des fonctions de la spécification), codes des exception, identificateurs de protocoles, longueur de paquets, et les adresse de données. Les travaux de (Lin et al., 2013) sont équivalent à celles de Cheung mais sur le protocole DNP3. Dans leurs travaux, ils ont conçu un analyseur de paquets pour l'IDS **Zeek** pour l'adapter aux systèmes SCADA. L'analyseur prend en charge des protocoles industriels tel que DNP3 et analyse les valeurs légales de différents champs dans un paquet afin de concevoir des politiques de sécurité correspondant au protocole.

(Morris et al., 2012) proposent de convertir le protocole terrain Modbus RTU/ASCII en trafic TCP afin de l'analyser avec des sondes **SNORT** configurées avec des règles respectant la politique réseau. Les auteurs ont utilisé 11 des règles SNORT créés par Digital bond pour examiner les données illégales. Cependant, la précision de détection de cette méthode repose en grande partie sur la définition précise des règles de SNORT. En 2013, ils ont amélioré cette approche ne proposant 50 règles de signature en analysant les failles du protocole Modbus et ont considérablement amélioré la précision de la détection.

Toutefois, cette méthodologie de détection permet de vérifier les violations dans l'utilisation du protocole de communication, mais ne s'intéresse pas à la détection des autres types

d'attaques, que ce soit les attaques venant du domaine informatique (tel que les attaques de déni de service et l'attaque homme au milieu (MitM)), ou bien les attaques respectant les protocoles utilisés et utilisant des trafic légitimes en se basant sur des connaissances avancées sur le système pour causer des effets malveillants en utilisant des commandes acceptables (attaques sémantiques).

D'autres approches se basent sur l'analyse de réseau de communication pour détecter les attaques sur les systèmes industriels. Ces approches supposent que le trafic dans un ICS est relativement stable dans les conditions normales ce qui permet d'utiliser le trafic comme un miroir reflétant le statut de sécurité dans un ICS. Ces approches peuvent être regroupées en deux catégories : (i) les approches de détection se basant sur la spécification des propriétés du réseau et (ii) les approches de détection se basant sur l'apprentissage automatique du trafic.

Plusieurs chercheurs ont proposé des approches basées sur la spécification du comportement normal de la communication et des contraintes à respecter. (Hadeli et al., 2009) proposent une méthode de détection d'intrusion qui crée un modèle du trafic à partir de la spécification du système. Ils exploitent les fichiers de configuration présent dans les ICS (fichier de configuration de sous-station (SCD) utilisées dans les sous-stations IEC 61850) en plus des informations implicites (tel que les discussions des experts de l'IEC 61850 codé en format XML) pour identifier les informations nécessaires tel que des informations sur le sous-réseau, les adresses IP des composants, les paramètres du bloc de contrôle GOOSE tel que l'adresse mac de diffusion, $t_{min}$, $t_{max}$, le dispositif permit d'envoyer les messages GOOSE. Ensuite, il transforme les spécifications en règles SNORT pour détecter les paquets requis manquant dans la communication en plus des règles pour les pares-feux du réseau.

(Barbosa et al., 2013) proposent une approche de détection basée sur une liste blanche des flux du réseau. Une modélisation du comportement normal du réseau caractérisé par les adresses du client et du serveur, le numéro de port de communication et le protocole utilisé sont utilisé pour caractériser les flux. Le modèle décrit le trafic légitime est créé d'une façon hybride en se basant sur un apprentissage qui se fait sur un enregistrement des paquets du réseau, en plus des spécifications de l'administrateur du système.

D’autres chercheurs utilisent les techniques d’apprentissage automatique sur les informations issues de la communication pour détecter les anomalies.

(Cheung et al., 2007) appliquent un apprentissage bayésien pour les modèles relatifs à la disponibilité des serveurs et des services pour identifier les composants communiquant et les codes fonction utiliser et détecter le changement de disponibilité des services ou bien l’apparition des services suspect.

(Linda et al., 2009) proposent une approche basée sur l’apprentissage par réseau de neurones pour caractériser la structure de l'échange (adresse IP, nombre de paquets échangés, intervalle entre deux trames, protocoles, ...).

Pareillement, (Vollmer and Manic, 2009) ont utilisé des techniques de réseau de neurones (NN) artificiel pour extraire les caractéristiques du trafic réseau (taille de paquet, ID de protocole ICMP, numéro de séquence ICMP, code ICMP, type ICMP, ID de protocole IP, option de protocole IP, temps de survie IP) afin de construire des vecteurs d'entrée pour l'apprentissage du modèle NN. Par conséquent, des attaques telles que le DoS et l'espionnage pourraient être détectées.

(Linda et al., 2011b) (Linda et al., 2011a) (Linda et al., 2012) ont réalisé un mécanisme de détection d'intrusion basé sur la logique floue (fuzzy logic). (Linda et al., 2011b) ont utilisé des règles floues pour représenter les schémas de comportement normaux du ICS. Les règles floues peuvent être extraites de la séquence de paquets du réseau en utilisant un algorithme de clustering en ligne ajusté le plus proche.

Luo (Luo, 2013) a conçu une approche de détection d'intrusion basée sur un SVM multi-classe. Plusieurs classificateurs SVM (tel que …) sont combinés pour déterminer la catégorie exacte d'une intrusion.

Khoury et Nassar (Khoury and Nassar, 2020) ont proposé un cadre basé sur une approche hybride utilisant la théorie des jeux (Game theory) et l’l'apprentissage par renforcement pour modéliser un jeu contradictoire pour la propagation de virus entre couches dans les réseaux CPS.

Maglaras et Jiang (Maglaras and Jiang, 2014) ont proposé un algorithme de détection d'intrusion ICS basé sur OCSVM (One-Class Support Vector Machine), qui n'exige aucune donnée de formation étiquetée ni une connaissance préalable des catégories d'attaque et peut être formé en mode hors connexion. Cette méthode est capable de construire des modèles de trafic pour plusieurs protocoles qui détectent divers comportements d'intrusion contre ICS, par exemple, Man-in-the-Middle et SYNFlood.

## 2.5 Études des IDS orientés Connaissance de procédé

Les systèmes basés sur l'analyse de procédé se basent sur les informations sémantiques et la particularité des systèmes de contrôle industriels pour détecter les intrusions. Sur l'ensemble des travaux présents dans la littérature, plusieurs axes de recherche ont pu être dégagés. Nous prenons les grandes catégories d'étude portant sur :

- L'identification des variables du process à partir des échanges.
- La détection se basant sur la spécification des états critiques et/ou les états autorisé.
- La détection basée sur les séquences d'événements.
- Création des filtres de détection à partir des modèles comportementaux du système et du contrôleur.
- Injection de données – modèle comportemental complet.

L'ensemble des approches énoncées par la suite repose sur une modélisation du système à l'aide d'un ensemble de règles, de représentations quantitatives (équations, représentation d'état) ou encore d'un apprentissage.

### 2.5.1 Approche de détection basées sur l'identification des variables du process à partir des échanges

(Hadziosmanovic et al., 2014) proposent d'identifier les variables du process à partir des paquets de communications. Leur approche commence par une phase de prétraitement, où elle extrait les valeurs des variables du procédé, et les classifie automatiquement en trois

catégories : (i) variables continues, (ii) variables discrets et (iii) variables constantes. Les valeurs observées des variables de type discret et constant seront sauvegardées. Alors que les variables continues sont modélisés par le modèle de régression « le processus auto régressif » et leurs limites maximales et minimales sont sauvegardé. Cette approche utilise une combinaison de l'IDS Zeek qui fait le « data extraction » et les enregistre dans des fichiers log avec du code écrit en C++ qui s'intéresse à caractériser, modéliser les données et détecter les attaques. Enfin, la détection d'attaque se fait soit en observant une valeur non déjà vue pour les variables discrets et constants, soit en violant le processus auto régressif ou bien en dépassant les limites pour les variables continues.

### 2.5.2 Approche de détection basées sur la notion des états critiques

D'autres travaux introduisent la notion de détection par l'analyse des états critiques. Cette méthode se base sur des spécifications manuelles sur les états interdits et/ou le comportement autorisé dans le procédé permettant de détecter des anomalies dans l'ICS en vérifiant l'état du système et permettent l'analyse de plusieurs paquets. L'approche de (Carcano et al., 2009) utilise une stratégie basée sur la signature d'un seul paquet pour détecter les paquets illicites envoyés aux systèmes de contrôle (PLCs et RTUs) et propose une technique de détection d'intrusion basée sur l'état, permettant de garder une trace de l'état du système industriel et d'identifier si un ensemble de commandes Modbus licites envoyées aux appareils de terrain peut amener le système dans un état critique. Ensuite, (Fovino et al., 2010) étend l'approche de (Carcano et al., 2009) pour qu'elle supporte le protocole DNP3 en plus du protocole Modbus. Dans le même sens, (Mitchell and Chen, 2014) ont développé un IDS qui s'appuie sur les règles de comportement des actionneurs et des capteurs. Cette approche propose de transformer ces règles en trois catégories d'états de machine : les états sécurisés, les états d'avertissements et les états non sécurisés. Le comportement de système est décrit sous forme d'automate fini probabiliste où les probabilités de transitions dépendent du type d'attaque. En plus, un degré de conformité est utilisé pour différencier entre attaque et comportement normale. Ces approches utilisent la notion des états du système prenant en compte l'analyse de plusieurs paquets et sont meilleurs que celles qui utilisent des règles de style basé Snort qui ne permettent

que d’analyser des paquets individuels. Cependant, ces approches négligent les contraintes temporelles sur les événements puisque leur langage de spécification des règles (ou du comportement) ne permet pas de spécifier des modèles temporels sur le comportement du système.

### 2.5.3 Approches créant des filtres de détection à partir des modèles comportementaux du système et du contrôleur

De l'autre côté, les approches (Fovino et al., 2011; Sicard et al., 2019, 2018) utilisent des modèles comportementaux du système et du contrôleur, puis convertissent ces connaissances en filtres. Fovino et al. (Fovino et al., 2011) propose une approche de filtrage de réseau basée sur le concept des états critiques et la distance entre l'état actuel du système et les états critiques. Cette approche commence par l'identification des états sûrs et critiques pour chaque variable du système. Ensuite, une image virtuelle du système est suivie par un logiciel qui reproduit les automates et les mémoires maîtres en utilisant deux méthodes : (i) extraire des données du réseau, et (ii) interroger les automates du système surveillé pour maintenir une synchronisation étroite entre les l'image virtuelle et le système réel. Enfin, les commandes PLC sont simulées à l'aide de l'image virtuelle du système et vérifiées avant d'être transmises au système physique. Cette approche se concentre uniquement sur les commandes envoyées au système physique et ne fournit aucune vérification des rapports (les mesures des capteurs) provenant du système physique.

Dans le même sens, Sicard et al. (Sicard et al., 2019, 2018) ont proposé une approche de filtrage basée sur des modèles comportementaux et des notions de distance d'état critique. Dans cette approche, les contraintes temporelles sont prises en compte et deux filtres sont déployés pour analyser les échanges entre le contrôle et le dispositif de l'ICS. Le premier analyse la commande envoyée du contrôleur au système physique lorsque le second analyse le rapport renvoyé au contrôleur. Cependant, cette approche se concentre uniquement sur les systèmes à événements discrets et ne prend pas en compte les événements continus. De plus, cette approche ne considère pas la vérification de l'évolution du système entre deux états stables.

### 2.5.4 Approches de détection basées sur les séquences d'évènements

D'autre approches se basent sur les séquences d'événements pour détecter les intrusions. Différemment des réseaux à grande variété de trafic tel que le réseau Internet, les systèmes de contrôle industriels montrent des communications régulières et cohérentes. Cette hypothèse est la base des approches « sequence-aware » qui cherchent des communications irrégulières dans le trafic. Ces approches sont complémentaires à celles présentées récemment de façon qu'elles prennent en considération les aspects timing et l'ordre des évènements reçus. Les auteurs de (Caselli et al., 2015) proposent un IDS « sequence-aware » qui s'intéresse à détecter un type spécifique d'attaques sémantiques provenant par des séquences d'événements individuellement légitimes. Ces attaques ne cherchent pas à endommager le système en premier lieu, mais plutôt à modifier l'ordre ou le timing des événements pour le dégrader progressivement. L'approche propose d'apprendre le comportement normal du système dans le temps et le représente par le modèle de chaîne de Markov (DTMC). Ensuite, un autre modèle est construit en ligne et est comparé au modèle de référence pour détecter les anomalies qui marquent des violations des séquencements et des contraintes temporelles de la commande. Dans le même sens, (Ferling et al., 2018) considère que les modèles de détection d'intrusion « sequence-aware » sont souvent volumineux et difficile à manipuler ce qui résulte en des analyses qui prennent beaucoup de temps. En conséquence, leur approche propose de réduire la taille des modèles de trafic en combinant des états du modèle DTMC qui diffèrent uniquement par la plage d'adresses IOA (Information Object Adresses) utilisées dans le protocole IEC-104.

### 2.5.5 Approches de détection des attaques d'injection des données

(Cárdenas et al., 2011) proposent une approche de détection qui se base sur la connaissance du système physique et s'intéresse surtout à l'objectif final de l'attaque au lieu de s'intéresser aux mécanismes particuliers d'exploitation des vulnérabilités et de masquage de l'attaque. Dans cette approche, la détection est axée sur le déni de service et la corruption des mesures envoyées par les capteurs notamment, en comparant les mesures reçues avec celle prédite obtenues avec un modèle précis du système.

(Papa et al., 2012) proposent un système de détection d'intrusion basée sur une fonction de transfert qui crée une relation entre les signaux d'entrées et les signaux de sortie. Dans cette approche, les auteurs considèrent qu'un signal de sortie peut être estimé en se basant sur un ou plusieurs signaux d'entrées, et Par conséquent, la comparaison d'un signal de sortie observé par un système peut être comparé avec un autre signal généré par simulation pour vérifier la possibilité d'avoir une injection (ou changement) de données erronées créé par un attaquant MitM. Cette approche permet de modéliser les relations linéaires et logiques combinatoire pour créer les signaux de simulation.

(Sridhar and Govindarasu, 2014) proposent une approche de détection d'intrusion dans le fonctionnement du système d'alimentation. Cette approche se base sur la connaissance du fonctionnement de système de régulation électrique et permet de détecter l'injection de données malveillantes en se basant sur la prévision de charge en temps réel.

## 2.6 Aperçu sur les IDSs existants

La littérature montre que plusieurs IDSs sont créés, et ont déjà prouver leur puissance et leur efficacité dans la détection des intrusions et sont applicables pour les systèmes industriels. Ces IDSs permettant aux utilisateurs de créer l'algorithme de détection le plus convenable pour leurs systèmes à partir des langages flexibles qui sont généralement appelés les règles de détection.

**Snort** est un système de détection d'intrusion open source, basé sur le réseau. Créé en 1998, Snort est capable d'effectuer une analyse au niveau du protocole (tel que vérifier une violation des spécifications d'un protocole de communication dans les événements observés), de chercher un contenu et peut être utilisé pour détecter une variété d'attaques. Snort a un langage de règles pour spécifier les signatures d'utilisation abusive et d'attaque (Roesch, 1999).

Le code ci-dessous présente un exemple d'une règle Snort sur le protocole de communication Modbus :

```
alert tcp any any -> any 502 (flow:established, from_client; msg:"Modbus Function Code 3 requested registers under 100"; content:"|03|"; offset:7; depth:1; byte_test:2,<,100; threshold:type threshold, track by_src, count 5, seconds 60; sid:1111101);
```

Cette règle permet de détecter un paquet Modbus demandant une lecture des registres Modbus "Read Multiple Registers" et de vérifier si le nombre de registres requis est inférieur à 100 parmi les sessions établies des clients. Les règles sont exprimées par rapport au contenu du flux et il n'est pas possible d'effectuer un filtrage avec état « stateful filters», sauf en utilisant des variables de flux telles que "Flowbits" et "Threshold" limités à la vérification d'une variable par rapport à un flux. Par exemple, vérifier que l'utilisateur est déjà connecté (en utilisant la variable Flowbits), ou qu'on a déjà essayé de nous connecter plus de 5 fois (en utilisant la variable de threshold).

**Suricata** (Albin and Rowe, 2012) IDS est créé par Open Information Security Foundation (OISF). Il a été présenté comme une amélioration open source sur Snort. Il a des opérations multithreads natifs pour traiter un plus grand volume de réseau que Snort. Suricata peut aussi faire du filtrage en utilisant les débits et le seuil comme Snort.

**Zeek** (Paxson, 1999) (le nouveau nom de l'IDS Bro) est un IDS Open source d'analyse de réseau. Il a été initialement développé en 1994. Il utilise des méthodes de détection basées sur les signatures et les anomalies. Il définit des attaques spécifiques en termes d'événements permettant de détecter un comportement inhabituel du réseau. Zeek utilise des signatures plus sophistiquées au moyen de son langage de politique, ce qui lui permet d'analyser le trafic réseau à un niveau d'abstraction beaucoup plus élevé et de disposer de puissantes fonctionnalités pour stocker des informations sur les activités passées et les intégrer à l'analyse de nouvelles activités (Thongkanchorn et al., 2013).

Plusieurs chercheurs implémentent leurs approches de détection en utilisant un de ces IDSs, ce qui permet de ne pas réinventer la roue, et de gagner beaucoup de temps par l'utilisation d'un IDS qui a déjà prouvé son efficacité.

## 2.7 Positionnement de l'approche proposée

Comme cela a été expliqué précédemment dans ce chapitre, plusieurs approches ont été proposées pour la détection d'intrusion dans les systèmes industriels. Cependant, ces approches assurent plusieurs niveaux de détection selon leur catégorie.

Sur le premier niveau, les approches de détection basée sur le protocole s'intéressent à la vérification de l'utilisation correcte du protocole de communication, et permet de détecter des tentatives d'utiliser les lacunes de protocoles pour appliquer des attaques au système.

D'autres approches se basent sur l'hypothèse de la périodicité du trafic dans les systèmes industriels et considèrent les changements dans le comportement général du trafic comme indice d'action suspecte.

Ces approches proposent une adaptation des techniques de détection provenant des systèmes informatiques. Du point de vue technique, ces approches peuvent être implémentées soit en utilisant des IDSs standards avec des règles génériques tels que les règles développées par Digital Bond, soit un plug and play des IDSs basés sur les technologies d'apprentissage automatiques nécessitant seulement une spécification des caractéristiques à prendre en considération. Et dans les deux cas, l'implémentation peut être considérée comme relativement simple. Cependant, elles ne peuvent être considérées que comme un premier niveau de détection des intrusions dans les systèmes industriels, puisqu'elles ne prennent pas en considération les connaissances du système physique. Les analyses appliquées dans ces approches négligent la partie donnée (les valeurs des quantités du système et les commandes envoyées depuis les contrôleurs vers le système physique). Ceci résulte par un manque de visibilité des IDSs sur les commandes envoyées à partir des contrôleurs vers le système physique, et les réponses du système physique contenant les valeurs des quantité mesurées par

les capteurs. Un pirate pourrait donc exploiter ce manque de vision pour injecter des commandes non-acceptables ou bien pour compromettre des capteurs du système sans qu'il soit détecté par l'IDS.

D'autres approches se basent sur l'analyse du procédé et des commandes de contrôle pour vérifier l'état du système. Ces approches prennent en considération les connaissances du système de contrôle et du système physique et cherchent la présence des commandes inacceptable ou bien des mesures hors de portée dans les valeurs des quantités du système physique. La mise en œuvre de ce type d'IDS nécessite généralement la création des règles métiers qui prennent en considération les valeurs échangées entre le système de contrôle et le système physique. Ceci nécessite la création d'un plus grand nombre des règles spécifique à l'ICS pour le système de détection. On peut parler donc d'une ou plusieurs règles à créer pour chaque quantité à vérifier qui doivent être créé pour chaque ICS, ce qui serait plus difficile à créer, prône à des erreurs humaines et difficile à gérer.

Ainsi, la sécurité des système industriels ne peut être assurée sans prendre en considération leur spécificité représentée par l'interaction entre la partie cyber et la partie physique du système. Un attaquant peut utiliser une connaissance approfondie des protocoles, du logiciel, du matériel et du système physique, ou pour endommager le système, à l'aide de commandes acceptables, ou pour maximiser les dommages causés au monde physique. Ce type d'attaque est appelé attaque sémantique (Caselli et al., 2015). D'ici vient l'importance des approches à base modèle du procédé qui détectent les attaques en cherchant des violations dans le comportement du process. Dans ce cas, l'IDS vérifie le respect de la loi de la dynamique du système lors de l'évolution des variables du système. Une telle méthodologie de détection nécessite la création de règles plus avancées et qui seront dynamiques en fonction des valeurs du process.

La création et la maintenance de telles règles est beaucoup plus complexe et difficile. Il faut donc avoir un générateur de règles qui permet de créer automatiquement les règles en se basant sur la connaissance du process.

Afin de résoudre ces problèmes, nous proposons :

- Une approche de synthèse basée sur un modèle du système pour gérer la complexité de la création et de maintenance des règles d'IDS pour ICS.
- Un algorithme de détection en ligne, qui de façon à ne pas réinventer la roue, s'appuie sur un IDS standard tel que Snort, Suricata et Zeek qui ont déjà prouvé leur capacité de détection.
- et un module d'extension à l'IDS classique pour ICS qui permet de prendre en compte l'état du procédé afin de prendre une décision de la validité des commandes reçu par le système physique.

## 2.8 Cadre de l'étude

L'approche développée se concentre sur la protection du procédé dans un ICS et est complémentaire aux approches orientées IT. Cette approche considère les attaques informatiques qui peuvent être exécutées ayant des effets sur la partie OT (operational technoloy) de l'ICS. Elle surveille les attaques ayant comme cibles :

- Le Control channel : canal liant la supervision au PLC ou bien liant le PLC au système physique, où l'attaquant cherche à corrompre les commandes envoyées ou bien injecter d'autres commandes pour commettre des actions malveillantes sur le système physique.
- Le report channel : canal utilisé pour envoyer les mesures du réseau de capteurs jusqu'à la partie commande, ou bien pour remonter les mesures depuis le PLC vers la supervision, où l'attaquant cherche à corrompre les informations renvoyées pour tromper la supervision afin d'introduire des ordres non-acceptables par rapport à l'état physique du système.
- Le PLC ou bien le système de supervision, en attaquant sa mémoire ou son intégrité logicielle, ou bien la disponibilité de leurs services, le pirate cherche à endommager la partie opérative de l'ICS.

## 2.9 Bilan du chapitre

Dans ce chapitre, une vision globale des moyens de sécurisation des systèmes industriels a été présentée. Ensuite, les systèmes de détection des intrusions adaptés aux systèmes de contrôle industriels ont été ciblés. Plusieurs catégories d'IDSs ont été présentées en détails ainsi que leur application aux systèmes industriels. Ensuite, une attention particulière pour la littérature traitant l'utilisation de la connaissance de procédé pour créer des mécanismes de détection d'intrusions a été considéré.

Ce chapitre a permis de souligner les points faibles et forts des approches de détection d'intrusion adaptés pour les systèmes industriels. On constate que les approches à base de modèle du procédé permettent une détection des intrusions intéressante pour les systèmes industriels. Des actions malveillantes se cachent entre les commandes acceptables du système de contrôle-commandes et ne peuvent pas être détecter par les approches de détection génériques. Celles-ci nécessitent l'utilisation des connaissances des systèmes de contrôles et du comportement légitimes du procédé par les systèmes de détections pour les détecter. Cependant, la conception, la configuration et la maintenance de telles systèmes de détection avancés nécessitent la création d'un très grand nombre de règles complexe d'IDSs, ce qui peut être très difficile à faire manuellement et rend nécessaire des outils pour la conception des IDSs qui se basent sur la connaissance de l'ICS et pour la génération automatique des règles de détection des intrusions.

# Chapitre 3 Modélisation pour ICS

## 3.1 Introduction

Ce chapitre est consacré à l'exposé de la méthodologie de modélisation des différentes composantes de l'ICS qui permettent d'apporter les connaissances nécessaires à la génération des règles de l'IDS et d'implanter des mécanismes de détection des actions malveillantes sur le système. Le point d'entrée de cette démarche est l'analyse de risques qui identifie les zones critiques pour le système à protéger. Ensuite, une modélisation des éléments représentant l'ICS est détaillée. La modélisation du système passe par les étapes suivantes :

- Définir le périmètre de l'ICS à protéger.
- Identification des paramètres descriptifs du comportement du système.
- Modélisation représentative des composantes de l'ICS.

Dans ce qui suit, nous présentons les différentes surfaces de risque pour un ICS. Puis nous identifions les paramètres descriptifs pour chacune ces surfaces possibles de risques. Ensuite, nous détaillons la modélisation des ces surfaces. Enfin, nous créons un modèle générique pour l'ICS.

## 3.2 Les attaques sur les systèmes industriels

### 3.2.1 Introduction

Un ICS est l'objet de menaces, générées par des sources qui utilisent un vecteur d'attaque pour mener une attaque. Une menace représente tout phénomène, chose ou personne qui peut potentiellement générer des dommages à l'ICS (Flaus, 2019).

Une attaque est une menace particulière et qui consiste à des actions malveillantes qui ont pour but de compromettre la sécurité du système d'information. En 2011, les informaticiens de la

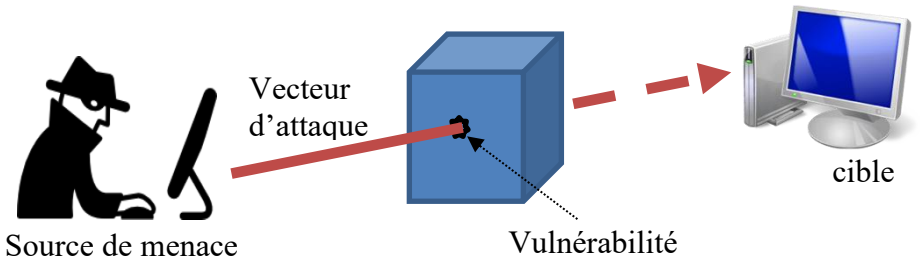



Figure 8: Vecteur d'attaque et vulnérabilité

société Lockheed-Martin introduisent un modèle de chaîne anti-intrusion « Cyber Kill Chain » (Hutchins et al., 2011) pour la défense des réseaux informatiques. L'institut SANS ont adapté ce modèle en 2015 pour les besoins des systèmes industriels et ont créé le modèle ICS Cyber Kill Chain ("ICS Cyber Kill Chain," 2015) qui consiste de deux étapes principales:

**Intrusion Cyber, préparation et exécution** : cette phase consiste à compromettre la sécurité. Elle peut être appliquée en suivant les étapes suivantes :

- Planification : consiste à collecter des informations sur le système et trouver les faiblesses qui peuvent être exploitées pour créer une attaque sur l'ICS.
- Préparation : consiste à décider de la cible d'attaque et à préparer une arme malveillante tel qu'un virus pour l'utiliser ensuite pour appliquer l'action malveillante.
- Cyber intrusion : consiste à tenter d'obtenir un accès initial. Cette phase inclut la transmission de l'arme par exemple par un courriel de phishing et l'exploitation qui lui permet de déclencher l'arme. Ensuite, quand l'exploitation réussit, l'adversaire installera (ou bien modifiera) un fonctionnalité tel qu'un cheval de Troie (Trojan horse) d'accès distant.
- Gestion et activation : à cette étape, l'adversaire obtient un accès distant et donc serait capable de commencer à appliquer son objectif.
- Maintien, enracinement, développement et exécution : cette étape décrit un certain nombre d'objectifs finaux qu'un adversaire pourrait avoir tel que la découverte de

nouveaux systèmes ou nouvelles données, le déplacement latéral sur le réseau, l'installation et l'exécution de fonctionnalités supplémentaires, le nettoyage des traces de l'activité d'attaque.

**Développement et exécution d'attaques ICS** : Cette étape consiste à utiliser les connaissances acquises dans l'étape précédente pour développer et tester spécifiquement une capacité capable d'attaquer l'ICS. Elle peut être appliquée en suivant les étapes suivantes :

- Développement et réglage d'attaques : l'adversaire (attaquant) développe une nouvelle capacité adaptée à une implémentation ICS spécifique et à l'impact souhaité.
- Validation : l'adversaire teste sa capacité sur des systèmes similaire pour vérifier qu'elle peut avoir un impact significatif et fiable.
- Attaque sur ICS : consiste à délivrer la capacité, l'installer et exécuter l'attaque.

Par conséquent, on peut distinguer deux types d'attaques selon la source de menace :

- Les attaques provenant de menaces externes qui doivent utiliser des vecteurs d'attaque pour collecter les informations et dépasser la sécurité appliquée.
- Les menaces internes provenant d'une source qui dispose d'un accès autorisé (en fonction de leurs droits d'accès) et qui peuvent causer des dommages sans avoir besoin de passer une vulnérabilité technique.

Quelle que soit la catégorie des menaces, l'objectif d'une attaque informatique est donc de créer des dommages sur l'un des biens de l'installation après avoir dépassé la couche de sécurité employé.

### 3.2.2 Les surfaces possibles d'attaques et les risques liées

Les points d'entrée pour une attaque sur un ICS incluent mais ne sont pas limitées à :

- Les **serveurs SCADA et les HMI (Human Machine Interface)**. Par exemple, un pirate peut avoir un accès sur le système de contrôle et donc il pourra modifier les paramètres de contrôle de façon à causer des dommages sur le système.

- Les **contrôleurs** tels que le PLC et le RTU (Remote Terminal Unit). La modification du programme de contrôle d'un PLC est un exemple d'attaque qui peut être utilisé sur ce niveau.
- Le **système physique** lui-même est exposé à des attaques, comme la modification physique d'un des composants ou bien le changement d'un équipement physique par un autre qui est compromis.
- **Les réseaux de communication** entre le SCADA et le contrôleur, et entre le contrôleur et le système physique.

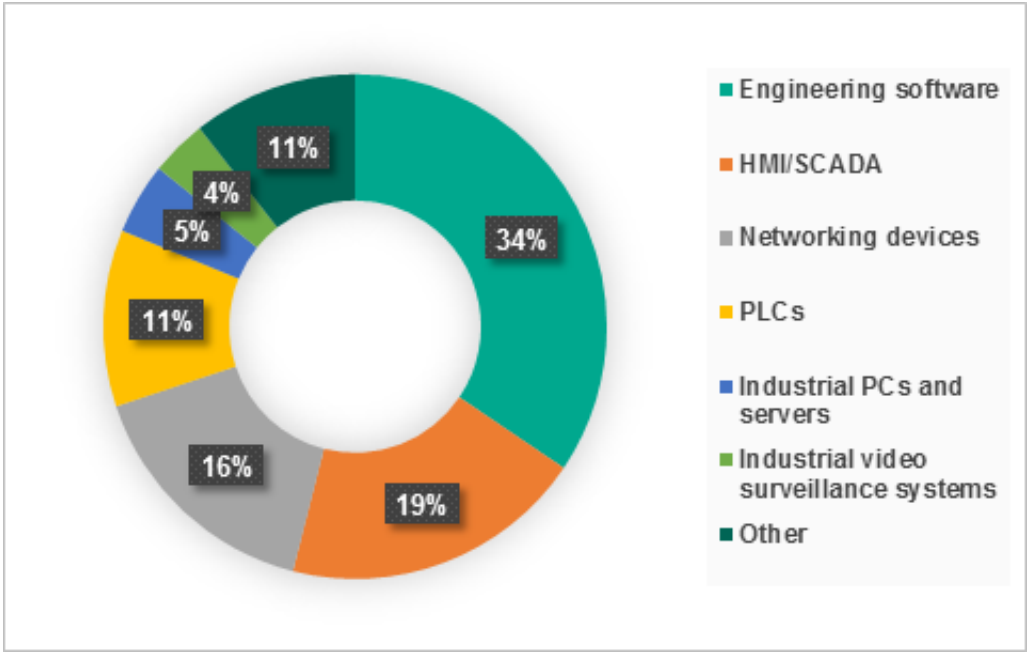



Figure 9: Distribution des vulnérabilité sur les composantes de l'ICS ("Threat landscape for industrial automation systems. H2 2018," 2019)

Le laboratoire Kaspersky ICS CERT a publié dans son rapport annuel ("Threat landscape for industrial automation systems. H2 2018," 2019) une distribution des vulnérabilités sur les composantes de l'ICS comme c'est illustré sur la Figure 9. Les logiciels d'ingénierie ont eu la plus grande part des vulnérabilités (143 vulnérabilités). En second lieu viennent les vulnérabilités sur les composantes SCADA/HMI avec 81 vulnérabilités. Ensuite, les dispositifs de réseau conçus pour les environnements industriels avec 66 vulnérabilités et 47 vulnérabilités sur les PLC. D'autres vulnérabilités sont réparties sur les ordinateurs et serveurs industriels

(5%), les systèmes de surveillance vidéo industriels (4%) et les divers appareils de terrain et les relais de protection.

Une analyse du risque du système montre que les risques comportent :

- **les risques provenant du SCADA et HMI** :
  - Un utilisateur SCADA tente de saboter le système ou dégrader la production en utilisant les commandes du système. Par exemple, une répétition successive d'une commande « acceptable » peut saboter un équipement ou bien une succession de commandes dans un ordre ou bien un timing spécifique.
  - Un utilisateur du système (ayant une adresse IP acceptable) tente de saboter le système à l'aide d'un logiciel externe.

À noter qu'un utilisateur du système peut être soit un utilisateur légitime qui utilise le système d'une façon erronée en le conduisant à des états critiques, soit un utilisateur légitime avec une intention malveillante, ou bien un pirate qui a été capable de dépasser les mesures de sécurité appliquées et a pris contrôle sur un serveur SCADA ou bien sur une HMI.

- **Les risques provenant du réseau de communication entre SCADA et PLC** : cette communication est habituellement assurée par un réseau informatique et utilise les protocoles industriels. La communication est susceptible à des attaques tel que Man-in-the-Middle (MitM), DoS (tel que l'attaque de l'inondation du réseau) et l'injection de paquets.
- **Les risques provenant du contrôleur (PLC)** : le contrôleur est susceptible des attaques tel que le téléchargement d'un logiciel non autorisé pour le contrôle du système physique, une attaque de déni de service (DoS, DDoS), l'usurpation d'identité du PLC et l'injection des commandes sur le système physique et des attaques physiques sur le PLC lui-même.
- **Les risques provenant du réseau de communication entre le PLC et le système physique** : cette communication peut être assurée par un réseau électrique (comme dans le cas des systèmes de contrôle traditionnels), ou bien par un réseau informatique (comme dans le cas de l'internet des objets industriels, IIOT). Les risques relatifs aux réseaux informatiques sont similaires aux risques sur le réseau de communication entre SCADA et PLC.

- **Les risques sur le système physique** : ce système est sensible à des attaques telles que la manipulation physique des équipements, le compromis des actuateurs et des capteurs.

Suite à la définition des surfaces possible d'attaques, nous procédons pour la description de chacune et la méthode de modélisation adaptable.

## 3.3 Modélisation de l'ICS

### 3.3.1 Introduction

Un système de détection d'intrusion a besoin d'avoir une connaissance sur le système à surveiller. Cette connaissance peut être fournie à l'IDS à partir des modèles représentant le fonctionnement de l'ICS. Par conséquent, notre but est d'avoir une connaissance du processus de production et sa configuration pour créer des filtres capables de détecter un comportement non prévu représentant un indice d'actions malveillantes sur le système.

Une chaîne de production dans un ICS peut être définie comme un ensemble d'équipement et de logiciels qui collaborent entre eux en suivant un processus bien défini pour créer le produit final ; d'une part, les contrôleurs sont programmés pour appliquer des algorithmes bien défini qui se basent sur des commandes SCADA et sur les mesures du système physique pour envoyer des commandes aux actionneurs et d'autre part, le procédé suit une loi de dynamique suite à des actions appliquées par les actionneurs. Par exemple, le volume de solvant dans une cuve change suite aux flux des vannes d'entrées et des vannes de sorties.

La vérification qu'un processus de production d'un système est « légitime » (fonctionne comme prévu) peut être faite en vérifiant que le système physique se comporte normalement en suivant la loi de dynamique du système et que les commandes reçues sont des commandes légitimes et leurs effets ne causent pas une violation des contraintes du système.

La première étape est donc de créer un modèle du comportement du système physique, de ses commandes d'entrée et des contraintes à respecter.  La modélisation du PLC fournit donc un moyen important pour la vérification des commandes envoyées vers le système physique.

Par ailleurs, dans le cas où seules des stations spécifiques doivent avoir accès au système de contrôle du processus de production, la plupart des implémentations actuelles des systèmes de contrôle ne présentent aucun des mécanismes et d'autorisation, ce qui permet à une personne ou bien un logiciel délivré ayant accès au réseau ICS d'exécuter des actions malveillantes sur le système. La surveillance des nœuds communicants et leurs droits d'accès au système de contrôle peut être considérée donc comme une responsabilité de l'IDS. Ceci nécessite donc d'avoir une modélisation de l'architecture du réseau et des droits d'accès des nœuds.

Par conséquent, deux types de modèles seront à spécifier :

- Un modèle statique pour décrire l'architecture des nœuds de la chaîne de production.
- Un modèle fonctionnel pour décrire le fonctionnement du contrôleur et du procédé (le système cyber physique CPS).

Ces deux catégories de modèles seront utilisées comme une base pour les filtres de détection du système de surveillance.

### 3.3.2 Modélisation de l'architecture

D'après (Hu et al., 2018), les systèmes ICS ont des objets et des processus métier relativement fixes, une topologie de réseau simple et statique et un petit nombre d'applications. Cependant, la plupart des implémentations des ICSs ne fournissent aucun mécanisme de sécurité pour le réseau local, et se limitent à appliquer les mécanismes de sécurité sur les frontières du réseau en utilisant les pares-feux par exemple. Ceci permet donc à un adversaire ayant eu la chance de se connecter au réseau local du système de contrôle d'avoir un accès non limité sur le process et de pouvoir facilement exercer des actions malveillantes sur le système de production. Par conséquent, il est indispensable que le système de surveillance complète les mesures de sécurité en surveillant les communications internes sur le réseau.

Pour modéliser l'architecture, ce manuscrit utilise un modèle sous forme d'un graphe G représentant les nœuds communicants sur le réseau, illustré sur la Figure 10 et représenté par le doublet $G = (N, L)$, avec :

- $N$ est un ensemble fini de sommets appelé nœuds et représente les adresses IP des composantes de l'architecture.
- $L$ est un ensemble fini d'arêtes appelées liens et définies par le couple $(n, n')$, où $n, n' \in N$.

Une communication depuis un nœud $n$ vers un autre nœud $n'$ est considéré acceptable si et seulement s'il existe un lien $l \in L$ avec $l = (n, n')$.

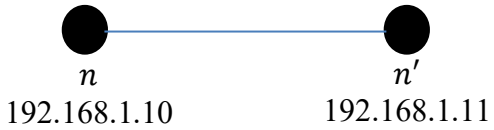



Figure 10: Modèle de l'architecture

### 3.3.3 Le modèle fonctionnel

#### 3.3.3.1 Introduction

Un modèle fonctionnel pour un ICS décrit le comportement normal (prévu) de l'ICS. En d'autres termes, ce modèle décrit le fonctionnement normal du contrôleur suite à la réception des commandes depuis les serveurs SCADA, mais aussi suite à la réception des mesures depuis le procédé. Ce modèle décrit aussi le fonctionnement normal du procédé suite à la réception des commandes de contrôle depuis le contrôleur.

#### 3.3.3.2 Choix du modèle

Plusieurs travaux de recherche ont été proposés pour la modélisation du CPS. Des langages non formels ont été utilisés tel que : Modelica (Liping et al., 2012), Architecture Description Langage (ADL) (Banerjee et al., 2012), Business Process Model and Notation (BPMN) (Seiger et al., 2018) et les profils UML (Schamai et al., 2013). D'autres chercheurs ont utilisé des modèles formels pour la spécification des CPS tel que réseau de Pétri (Basile et al., 2016),

automates temporisés (Pajic et al., 2014) et hybrides (Li et al., 2014) . Une statistique faite par l'auteur de (Graja et al., 2018) montre que les modèles formels sont les plus utilisés parmi 50 articles étudiés sur la modélisation des CPS. Par conséquent, nous serons plus intéressés à la modélisation en utilisant les modèles formels dans notre approche. Nous aurons donc à choisir un formalisme le plus complet pour décrire un CPS.

Les formalismes basés sur la logique temporelle telle que le réseau de Petri temporisé et les automates temporisés permettent de représenter les propositions qualifiées en termes de temps. Autrement dit, seul le temps est la variable continue représentée par ce type de formalismes. Donc, ces caractéristiques rendent ces formalismes insuffisants dans le contexte du CPS.

Nous avons donc choisi le modèle de l'automate hybride qui est un modèle mathématique qui combine des graphes de contrôle discrets à des variables évolutives continues, décrites par un ensemble d'équations différentielles ordinaires.

#### 3.3.3.3 Le modèle d'automate hybride

Un automate hybride est une extension de l'automate fini par l'ajout des variables continues qui constituent le vecteur d'état continu. Une première définition pour des systèmes autonomes a été donnée par (Alur et al., 1995). Par la suite, ce manuscrit se base sur cette dernière définition pour définir le modèle d'automate hybride.

Un automate hybride, illustré sur Figure 11, est défini par l'octuple H = $(L, X, E, I, \Sigma, \mathrm{l}_0, \mathrm{Q}, \mathrm{F})$, avec :

- $L$ est un ensemble fini de sommets appelé les places.
- $X$ est l'espace d'état continu des automates, $X \subset \mathbb{R}^n$. Un état est une paire $(l, x)$ faite d'une place $l \in L$ et d'un vecteur $x \in X$. On note $\Omega$ l'ensemble des états.
- $E$ est un ensemble d'arêtes appelé transitions défini par un septuple $(l, event, Guard, Jump, l', valid, internalEvent)$, avec :
    - $l, l' \in L$.
    - $event \in \Sigma$.

- $Guard$ est une expression conditionnelle dépendant de $X$. C'est une fonction de $\mathbb{R}^n \rightarrow \{true, false\}$.
- $Jump$ est une fonction de $\mathbb{R}^n \rightarrow \mathbb{R}^n$ qui permet de définir le nouveau vecteur d'état
- $Valid$ est une expression conditionnelle dépendant de $X$ et des évènements. C'est une fonction $\mathbb{R}^n \rightarrow \{true, false\}$ qui marque la condition de validité d'une transition.
- $internalEvent$ est un événement interne qui se produit suite au déclanchement de la transition.

Une transition doit être franchie lorsqu'un des événements spécifiés reçus ou bien que $Guard$ soit vérifié. Une $Guard$ non spécifiés est considéré comme toujours false. Si aucun événement n'est spécifié la transition serait franchi en ayant la $Guard$ vérifiée.

- $\Sigma$ est un ensemble fini d'événements marquant les arêtes. Chaque arête est associée à zéro, un ou plusieurs événements par la fonction d'étiquetage $E \rightarrow \Sigma$. Les événements d'une arête sont reçus et ils peuvent être utilisés pour synchroniser plusieurs automates hybrides.
- $I$ est une fonction qui dans chaque place $l \in L$, associe une collection d'expressions logiques conditionnelles dépendant de $X$. Chaque expression est notée $Inv(x)$. C'est une fonction de $\mathbb{R}^n \rightarrow \{true, false\}$.
- $Q \subset X$ appelé l'ensemble des états initiaux.
- $l_0 \in L$ est la place initiale de l'automate. Elle est représentée par une flèche qui pointe vers une place.
- $F$ est une fonction qui à chaque place $l \in L$ associe un vecteur $F(X)$ appelé l'activité de la place $l$. Chaque composante notée $f_i(x)$ permet de déterminer l'évolution de la composante $x_i$ du vecteur d'état.

Les temporisateurs sont considérés des cas particuliers des états continus ayant la fonction $f_i(x) = 1$.

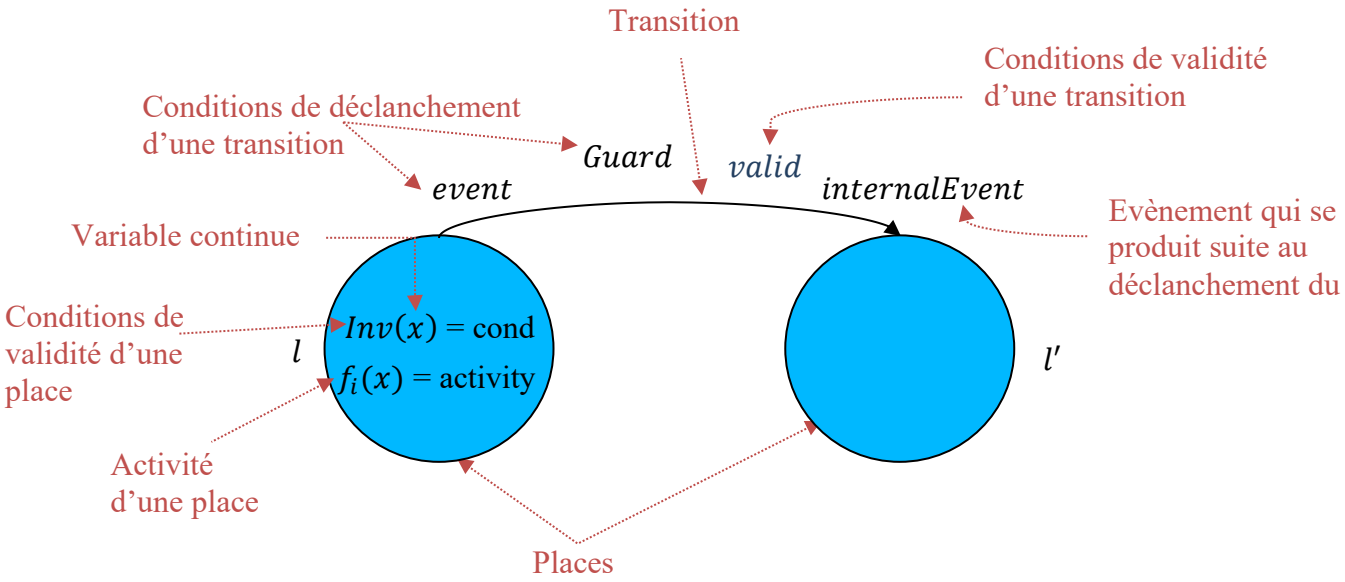



Figure 11: Automate hybride

### 3.3.3.4 Modèle fonctionnel générique

La modélisation par des automates hybrides va être utilisée pour la spécification du comportement acceptable du système, du contrôleur et du réseau.

Cependant, la création d'un seul modèle spécifiant tout le système est complexe. Les auteurs de (Larsen et al., 1997) proposent de modéliser le système par un réseau d'automates (temporisé) qui sont synchronisés entre eux et qui peuvent communiquer soit via des variables entières, soit via des canaux de communication. Une telle modélisation modulaire permet de représenter les systèmes complexes d'une façon plus simple (par exemple en modélisant chaque élément à part). Ce manuscrit utilise cette modélisation modulaire pour la modélisation du CPS et en utilisant des modèles d'automates hybrides synchronisés. Par conséquent, notre

approche commence par modéliser chaque entité du CPS par un ou plusieurs modèles d'automates hybrides. Ensuite, une synchronisation entre les modèles d'automate hybrides par le moyen des événements internes, des événements externes et des variables globales est appliquée pour représenter le comportement complet du CPS.

### 3.3.4 Modélisation du PLC

Comme c'est expliqué précédemment dans la section 1.2.4, un PLC exécute un algorithme bien déterminé pour contrôler la gérer la production. Cet algorithme est codé dans un langage de programmation de PLC tel que celle spécifié dans la norme IEC-61131-3. Ceci peut être utilisé donc pour connaître le fonctionnement attendu du PLC. SFC ou bien « Sequential Functional Chart » est un de ces langages qui est trop utilisé pour programmer le PLC. Par conséquent, ce manuscrit propose de modéliser le fonctionnement du PLC en traduisant le programme écrit en langage SFC en un modèle en automate hybride.

#### 3.3.4.1 Le langage SFC

Selon la définition donnée dans la norme IEC 1131-3 , un graphe SFC est constitué d'étapes et des transitions interconnectés par des liens comme c'est montré dans la Figure 12. Une étape peut être soit active, soit inactive. Zéro ou plusieurs actions sont associés à chaque étape. Le changement d'une étape à une autre est représenté par un ensemble de transitions qui doivent être déclenchées simultanément. Une transition serait déclenchée suite à la satisfaction d'une condition résultante d'une expression booléenne des variables pouvant être influencées soit par les actions associées aux étapes, soit aux événements externes.

Dans ce qui suit, nous simplifions la définition des actions en se référant à (Hassapis et al., 1998) qui limite la définition d'une action à une fonction qui attribue une valeur de type booléen ou réelle à une variable interne et aux variables de sortie.

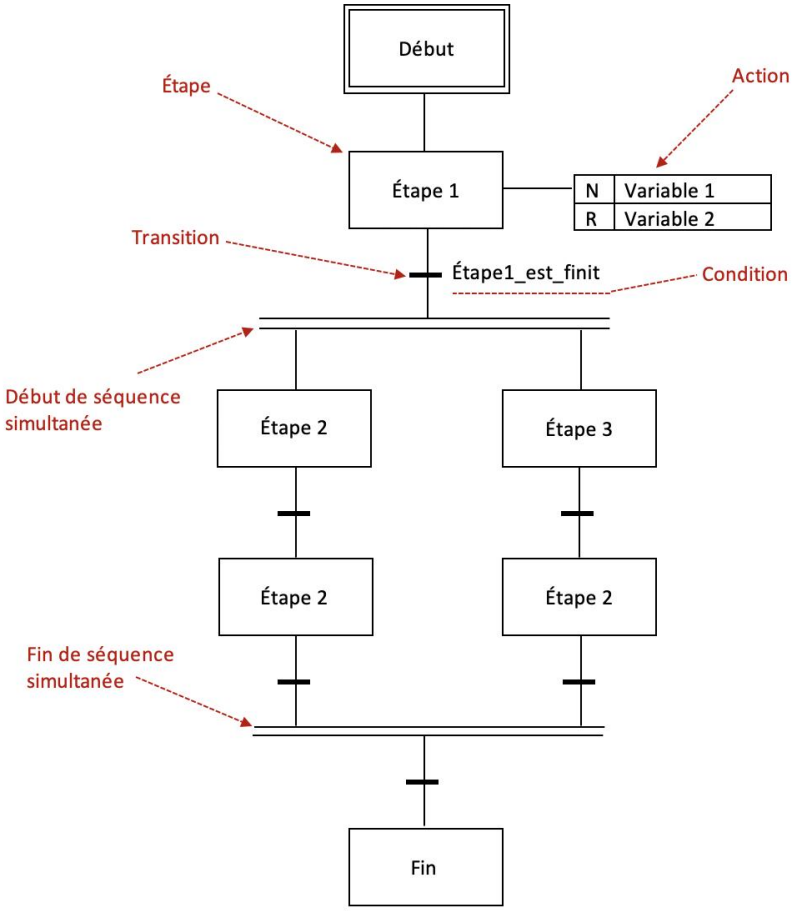



Figure 12: SFC de base, avec des éléments importants étiquetés

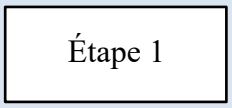
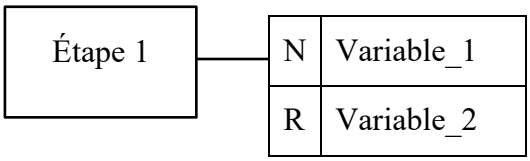
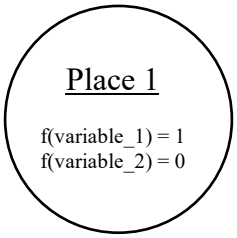
### 3.3.4.2 Traduction du langage SFC vers Automate hybride

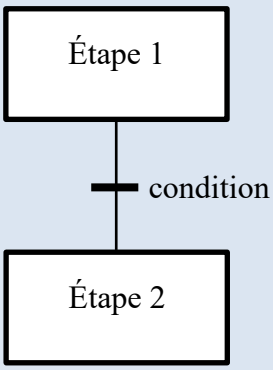
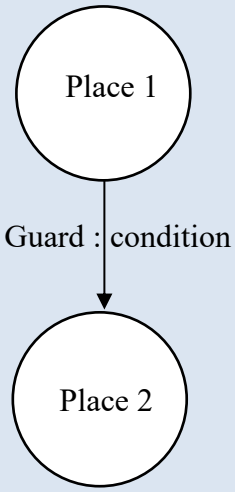
Le langage SFC peut être facilement traduit vers un modèle d'automate hybride (HA) en considérant que les deux sont constitués d'étapes et de transitions. Cependant, la tâche la plus difficile consiste à gérer les opérations parallèles (séquences simultanées). (Bauer et al., 2004) proposent de modéliser un programme SFC avec des automates temporisés en utilisant l'outil UPPAAL (Larsen et al., 1997) et créent des règles de transformations qui peuvent être appliquées au graphe SFC pour réduire sa complexité. Nous utilisons cette méthode pour traduire un SFC avec séquences simultanées vers plusieurs graphes synchronisés, sans séquences simultanées et équivalent aux graphes initiaux.

L'algorithme de traduction de SFC vers HA est illustré dans le Tableau 2, et est constitué des étapes suivantes :

1. Réduire la complexité du graphe SFC avec transitions simultanées en créant des graphes « logiquement synchronisés » équivalents (expliqué ci-dessous).
2. Une étape dans SFC est traduite en une place dans un automate hybride.
3. Une action associée à une étape dans un SFC est traduite en activité dans une place.
4. Une transition SFC est traduite en transition dans HA. Une condition de transition SFC est traduite en « guard » de la transition dans notre définition de HA.

Tableau 2: Conversion des élèments SFC en élèments HA

| # | Conversion SFC → HA | SFC | Automate hybride (HA) |
|---|---|---|---|
| **2** | Etape → place |  |  |
| **3** | Action → activité |  |  |

| | | | |
|---|---|---|---|
| **4** | Transition SFC<br>→ Transition HA |  |  |

Nous expliquons l'algorithme de traduction en présentant la traduction de l'exemple illustré sur la Figure 13.

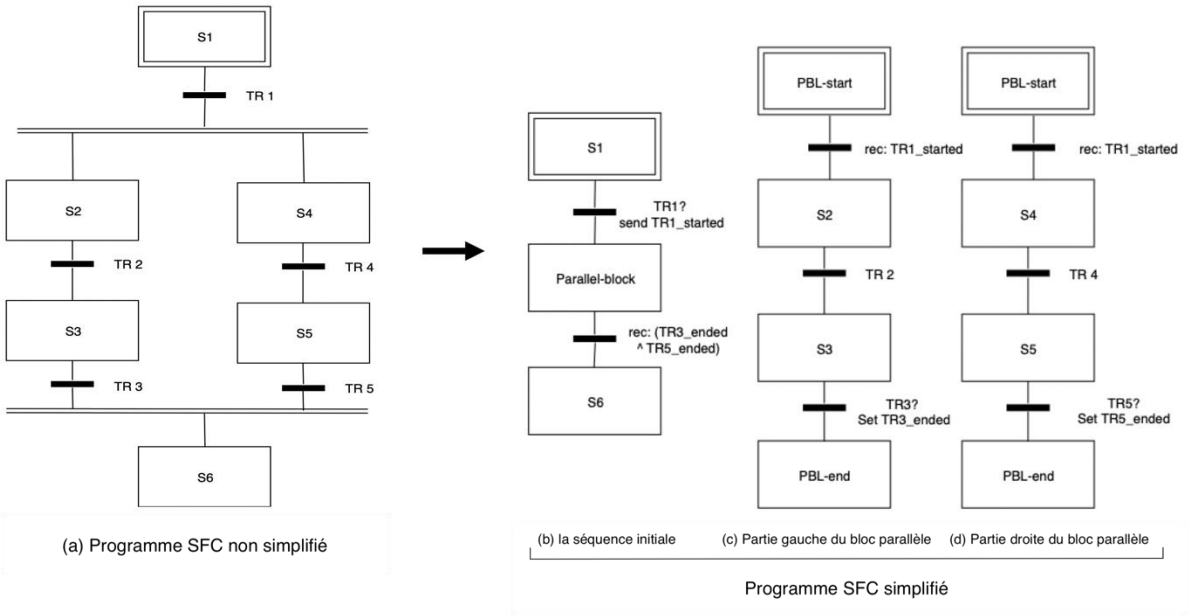



Figure 13: Explication de l'algorithme de simplification logique du programme SFC

**Réduction de la complexité de la séquence SFC** : l'idée est de remplacer le bloc contenant la séquence simultanée par une seule étape pour avoir une séquence simplifiée. Ensuite, créer d'autres séquences (représentant la partie simplifiée) qui synchronisent avec la séquence initiale à partir des évènement internes et des variables de type Booléenne.

La Figure 13 (b) représente la séquence initiale simplifiée du programme SFC de la Figure 13 (a). Elle consiste à remplacer la séquence simultanée par une étape « Parallel Block » avec :

- La condition de la transition d'entrée (TR 1) est la même de celle de la séquence simultanée (Parallel-block).
- Les transitions de sortie de la séquence simultanée (TR3 et TR5) sont remplacées par une transition ayant une condition (TR3_ended ^ TR5_ended) constituée de la réunion de toutes les conditions de sortie par l'opérateur logique « AND ».

**Les séquences synchronisées équivalentes à la séquence simultanée** : le contenu du bloc « Parallel block » peut être vue comme deux séquences (la séquence de la partie gauche et la séquence de la partie droite). Dans ce qui suit, nous référons à ces deux séquences par les termes « séquence gauche » et « séquence droite ». La Figure 13 (c) et la Figure 13 (d) représentent les séquences logiquement équivalentes aux « séquence gauche » et « séquence droite ». La Figure 13 (c) est créée en suivant les étapes suivantes :

- Ajouter une étape initiale à la séquence (l'étape « PBL-start »). Le rôle de cette étape est d'attendre la satisfaction de la condition d'entrée (TR1) à la séquence simultanée. À noter que la satisfaction de la condition (TR1) cause la génération d'un évènement de synchronisation interne (TR1_started).
- Les mêmes étapes et transitions de la « séquence gauche » sont placées ensuite.
- Ajouter une dernière étape à la séquence « PBL-end » pour marquer la fin de la séquence.

De même, la Figure 13 (d) est créée en suivant les mêmes étapes. Cependant, une telle représentation (Figure 13) est sujet à un fonctionnement non parfait causé par une satisfaction de la transition d'entrée vers la séquence simultanée sans que l'étape S1 soit non-active. Nous résoudrons ce problème lors de la traduction vers HA en utilisant des événements internes de HA au lieu de la condition de la transition.

**Création des automates équivalent**s : Les automates (a), (b) et (c) de la Figure 14 sont respectivement équivalent aux séquences SFC (b), (c) et (d) de la Figure 13. Les transitions entre « S1 » et « PB », « PB » et « S6 », « PBL-Idle » et « S2 », « S3 » et « PBL-ended », « PBR-Idle » et « S4 » et « S5 » et « PBR-ended » sont spéciales et permettent la synchronisation entre les automates. La transition entre « S1 » et « PB » de l'automate (a) est déclenchée suite à la satisfaction de la condition « TR1 » et génère un événement interne « TR1-started ». Cet événement « TR1-started » est utilisé pour déclencher les transitions entre « PBL-Idle » et « S2 » et entre « PBR-Idle » et « S4 ». Par conséquent, la seule manière d'arriver aux locations « S2 » et « S4 » seraient de passer la transition entre « S1 » et « PB ».

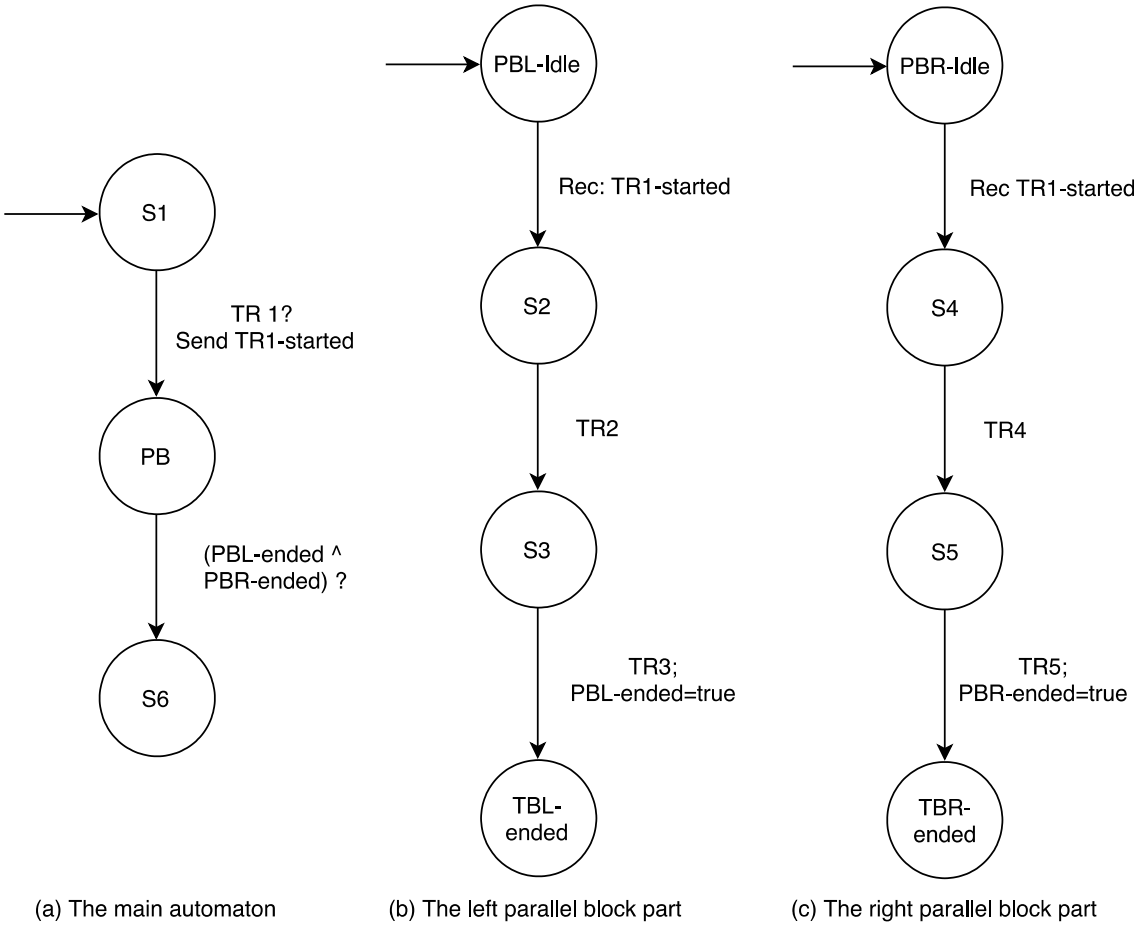



Figure 14: Automates représentant le programme SFC

### 3.3.5 Modélisation du système physique

#### 3.3.5.1 Les caractéristiques du procédé

Le système physique est le cœur de la production industrielle. Il est, en général, la cible des actions malveillantes qui cherche à créer des dommages sur l'un de ses biens, d'entraver ou saboter son processus de production. Et dans tous les cas, l'effet des actions malveillantes va être présentes sur le système physique. La connaissance du comportement attendu du procédé est donc primordiale pour vérifier l'existence d'indices sur des actions malveillantes dans un système de contrôle industriel.

Les actionneurs du système physique reçoivent les commandes et les actions depuis le système de contrôle pour gérer la production. L'application de ces entrées permet de conduire le système physique d'un état $x1$ vers $x2$ selon des lois déterminées par la dynamique du système.

La Figure 15 montre une modélisation d'un système physique qui est caractériser par :

- Les entrées représentées par les commandes et les actions reçues à partir du système de contrôle ($U_1$, …, $U_m$).
- Les sorties représentées par les mesures du système ($Y_1$, …, $Y_m$).
- La dynamique du système représentée par les lois qui décrivent le changement des variables du système ($x_1, \dots, x_n$) au cours du temps selon les entrées reçues.
- Les contraintes permettant de garantir le fonctionnement attendu du système.

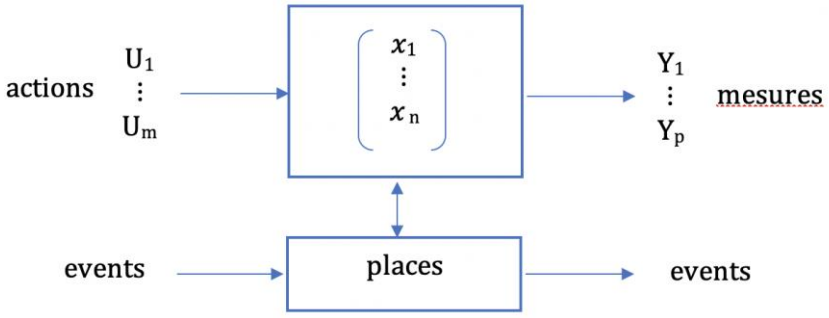



Figure 15: Modélisation d'un système physique

L'état du procédé peut être caractérisé par la valeur de ses mesures ($Y_1$, …, $Y_m$). Ces variables peuvent être de nature discrètes (par exemple un four peut être allumé ou éteint) ou continues (tel que le volume dans un réservoir). Le comportement des variables continues peut être modélisé par des équations algébriques ou des équations différentielles de la forme :

$$\frac{dx}{dt} = f(u, t)$$

Tandis que le comportement discret peut être exprimé à partir des assignations discrètes (par exemple mettre état_du_moteur=allumé).

Les entrées sont des variables qui peuvent être soit de nature discrète (par exemple commande d'ouverture d'une vanne), soit de nature continue (par exemple commande de régulation d'une vanne à une valeur d'a%, où a ∈ [0, 100]).

Une variable est caractérisée par :

- Son nom.
- L'adresse utilisée pour la communication de sa valeur (tel que le numéro de registre dans le protocole Modbus)
- La source générant le flux de données et sa destination. Par exemple, la valeur d'une variable d'entrée peut être récupérée en inspectant les paquets provenant du contrôleur vers un actuateur.

Les contraintes du système définissent les conditions dans lesquelles le système doit fonctionner afin de maximiser la performance de la production et de s'assurer qu'il n'y aurait pas des effets ou résultats inattendus. Ces contraintes sont généralement définies par un expert et comportent :

- Des conditions sur la séquence de commandes qui doivent être reçues par le système physique qui inclut l'ordre et le timing des commandes.
- Des conditions pour vérifier si une commande est acceptable (ou pas). Ces conditions incluent l'existence d'autres commandes qui sont en cours d'exécution ou bien l'état actuel du système physique. Par exemple, un système de contrôle des trains ne doit pas

envoyer des commandes de freinage tout en exécutant des fonctions de marche. De même, un système de remplissage d'un bac ne doit pas recevoir des commandes d'ouverture de vanne d'entrée lorsque la bac est plein.

- Des conditions sur les valeurs maximales ou minimales des variables du système, que ce soit un effet d'application d'une entrée ou bien causé par l'environnement. Ces conditions peuvent inclure la nécessité de réception d'une ou bien d'une seule commande ou bien une séquence de commandes suite à des états spécifiques du système.

#### 3.3.5.2 Modélisation du procédé

Le procédé est un système cyber-physique qui comprend deux aspects :

- L'aspect continu représenté par le changement des mesures du processus (par exemple, la variation du volume dans un réservoir)
- L'aspect discret à représenter par les commandes de contrôle (par exemple, vanne ouverte).

Comme cela a été expliqué précédemment, le modèle d'automate hybride est un modèle représentatif du comportement des systèmes hybrides. Pour cela, ce manuscrit propose de modéliser le procédé par un modèle HA. Ainsi, puisque le CPS peut être très complexe selon la taille ou bien le nombre d'équipements, sa modélisation par un seul automate hybride peut être très complexe ou non faisable (par exemple en provoquant une explosion d'état). Ce manuscrit propose donc de modéliser le procédé d'une façon modulaire, où chaque élément fonctionnel tel que les capteurs et les actionneurs est modélisé à part. Ensuite, tous les modèles seront synchronisés entre eux et leur combinaison représente le fonctionnement total du CPS.

La modélisation se fait donc en représentant les éléments suivants :

- La partie discrète est représentée par des places du HA.
- La partie continue est représentée par les états continus du HA.

- Les lois de la dynamique du système sont représentées sous forme d'équations différentielles par les activités des places du HA.
- Les entrées du CPS sont représentées par les événements et les conditions « Guard » sur les transitions du HA.
- Les contraintes du CPS sont représentées par les invariants (conditions sur les variables d'état) et les conditions de transitions (condition sur les commandes reçus).

Notre approche n'impose pas la manière de choisir les équipements fonctionnels du CPS. De même, ce manuscrit laisse pour l'utilisateur le choix du nombre de HA à utiliser pour modéliser son CPS.

### 3.3.6 Modèle générique de l'ICS

Les sections précédentes définissent la modélisation des éléments caractéristiques de l'ICS. La combinaison de ces modèles construit donc le modèle total de l'ICS. Par conséquent, le modèle générique de l'ICS comprend :

- Le modèle de l'architecture de l'ICS et les droits d'accès de chaque élément.
- Le modèle du PLC.
- Le modèle du CPS.

Cependant, les modèles du PLC et du CPS se basent dans leurs définitions sur les variables pour représenter les quantités d'un système physique, les commandes et les actions. Du point de vue de système de surveillance, ces connaissances restent des définitions théoriques du comportement du système tant que ces informations ne peuvent pas être reliées à l'implémentation du système. Il faut donc définir pour le système de surveillance comment peut-il récupérer les valeurs des variables spécifiées ; que ce soit depuis les communications en spécifiant les adresses utilisées pour transmettre les variables ou depuis d'autre sources. Ce manuscrit propose de récupérer les valeurs des variables spécifiées à partir des communications réseaux, ou en se connectant au PLC en spécifiant les adresses des registres Modbus relatives à ces variables.

En conclusion, le modèle générique illustré sur la Figure 16 présente un modèle qui englobe les caractéristiques de l'ICS. Ce modèle serait utilisé comme une base de connaissance pour créer les filtres de surveillance de l'IDS.

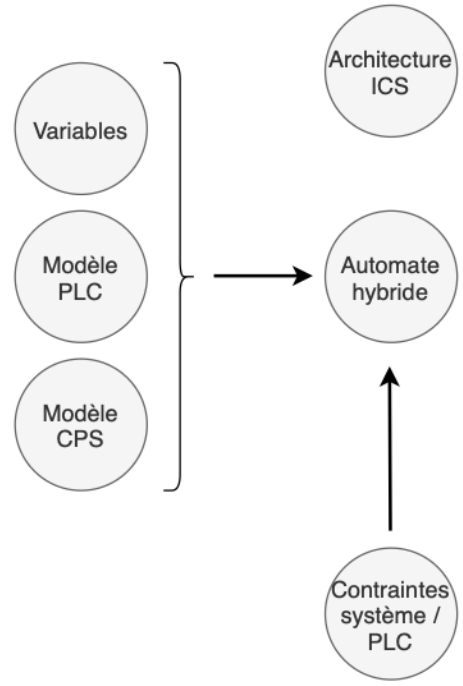



Figure 16: Modèle générique de l'ICS

## 3.4 Bilan du chapitre

Dans ce chapitre, les aspects modélisation des connaissances de l'IDSs ont été présentés. En premier lieu, les surfaces d'attaques possibles dans un ICS sont présentées. Puis, une modélisation des parties principales de l'ICS (l'architecture, le contrôleur et le procédé) a été expliqué. Ensuite, une méthode de conversion d'un programme PLC écrit dans le langage SFC en des modèles d'automate hybride est proposée. Enfin, un modèle générique de l'ICS est mis en place pour une description de l'ICS tout entier est présenté. En effet, la stratégie de détection mise en place dans notre approche se base sur ces modèles pour avoir les connaissances nécessaires de l'ICS.

# Chapitre 4 Conception d'IDS en se basant sur un modèle de l'ICS

## 4.1 Introduction

La littérature montre que la conception d'un IDS pour un ICS nécessite d'avoir une connaissance avancée du système à surveiller. Ensuite, on se réfère à cette connaissance pour vérifier la présence des indices d'action malveillante, ce qui permet de détecter des attaques connues, ou bien pour chercher des violations dans le comportement du système pour vérifier la présence d'une attaque peut être inconnue. Ces deux méthodes complémentaires permettent d'avoir une solution plus complète pour la détection des tentatives d'intrusions.

En fait, la connaissance du système peut être répartie sur plusieurs parties, que ce soit l'architecture, les variables du système, le modèle du PLC, le modèle du CPS et ses contraintes. Cette connaissance peut être soit spécifiée par un administrateur du système, soit générée par des algorithmes de génération automatique tels que celles qui apprennent les caractéristiques à partir des traces représentatives du système. Ensuite, cette connaissance doit être fournie à l'IDS pour surveiller le système d'une façon spécifique.

Ce chapitre est consacré à la présentation de notre approche de conception d'IDS basée sur un modèle pour gérer la complexité de la conception et la maintenance d'IDS pour les systèmes industriels.

Le but de cette approche est de :

- Créer un IDS avec état physique du système pour détecter les attaques possibles en se référant sur les connaissances approfondies du CPS.
  Un IDS avec état physique du système est un IDS qui analyse les événements observés et décide leur validité ou bien s'ils sont acceptables en se basant sur l'état physique du

système. Un événement est considéré acceptable s'il représente une évolution normale du système et respecte les contraintes du système. L'état physique du système peut être caractérisé par les valeurs des quantités (les variables) du système. Par conséquent, une analyse complète de l'IDS nécessite une vision complète sur l'état physique du système à surveiller. Cette vision peut être assurée pour l'IDS à partir d'un module spécifique capable de construire une image de l'état du système.

- Utiliser l'un des IDS standard existants tels que Snort, Suricata ou Zeek comme détecteur d'intrusion.
- Résoudre les difficultés de création des règles IDS pour un système complexe par une génération automatique de ces règles en se basant sur le modèle du système.

Dans ce chapitre, nous présentons notre approche de conception d'IDS à base modèle pour détecter les attaques possibles sur les ICSs. Cette approche sera expliquée sur un cas d'utilisation.

## 4.2 Le principe de l'approche de conception d'IDS proposée

L'approche proposée est de faire une génération automatique des règles pour un IDS avec état physique spécifique pour un ICS en se basant sur les connaissances approfondies du système. Cette approche, présentée dans la Figure 17, est composée des trois étapes suivantes :

1. **Construction du modèle** : il s'agit de créer un modèle du CPS qui définit son comportement normal en utilisant un langage de modélisation formel.
2. **Transformation du modèle** : un algorithme spécifique, présenté dans la suite, convertit le modèle d'entrée en règles de l'IDS.
3. **Implémentation de l'IDS en ligne** : le résultat est implémenté comme suit :
    a. Nous utilisons un IDS open source pour implémenter les règles générées IDS pour la détection d'intrusion.
    b. Comme certaines règles s'appuient sur l'état du CPS, un module spécifique réalise une prédiction de l'état du process et le fournit à l'IDS.

La première étape (construction du modèle) a déjà été expliquée dans le chapitre précédent (chapitre de modélisation). Dans ce qui suit, nous détaillons cette approche en présentant la méthode de synthèse (étape de transformation de modèle).

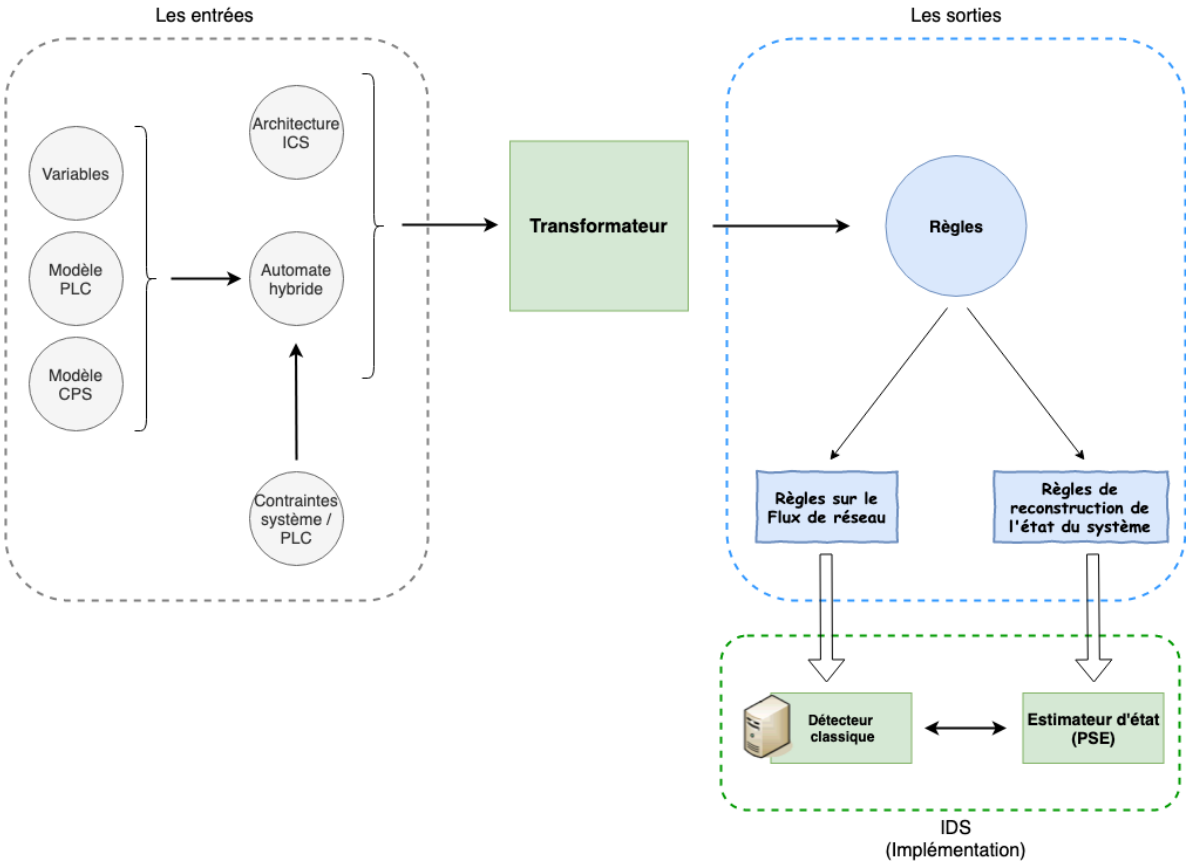



Figure 17: L'approche proposée dans ce manuscrit

### 4.2.1 Le principe de notre IDS

**L'IDS a pour objectif** de surveiller les invariants du système (les contraintes à respecter). Il génère une alerte si un invariant (représentant une contrainte) devient invalide ou si un événement reçu mène à une place interdite ou avec un invariant invalide.

**Pour la communication**, l'IDS a pour objectif de détecter des flux de communication non permis, qui peuvent être des communications créées par un nœud intrus, ayant une destination

intruse, ou bien communication entre deux nœuds légitimes qui ne sont pas prévu de communiquer entre eux.

**Pour un système physique**, l'IDS a pour objectif de détecter les commandes invalides (par exemple tentative de remplissage d'un bac plein), et les problèmes dû à des commandes passées valides (par exemple remplissage d'un bac à moitié plein) mais qui ne le sont plus, car la situation a évolué (le bac s'est rempli). Cette détection est réalisée via les invariants qui deviennent invalides.

**Pour un automate (un PLC)**, les entrées sont les commandes SCADA et les mesures du système. L'IDS générera une alerte si :

- les commandes SCADA ou les mesures mènent à une place interdite ou avec un invariant faux (commande non permise, mesure reçue mène à une situation anormale pour l'automate).
- les commandes ou bien les mesures violent un invariant (évolution anormale, le PLC attend trop longtemps la réalisation d'une action).

### 4.2.2 Le module de génération de règles

Le module de génération de règles ou transformateur de modèle est un logiciel permettant de transformer la connaissance de l'ICS, représentée par des modèles formels, en règles permettant la conception de l'IDS. Les données d'entrées du transformateur sont :

- L'architecture de l'ICS.
- Le modèle d'automate hybride du système, représentant :
    - Les variables (les quantités du système).
    - Le modèle du PLC.
    - Le modèle du CPS (les commandes d'entrées et les mesures créées, la loi de la dynamique du système physique modélisant le changement de valeurs des quantités du système physique).
    - Les contraintes du CPS.

Le transformateur transforme ces entrées en règles de l'IDS de TYPE-I (expliquées ci-dessous). Deux catégories de règles sont créées par le transformateur :

- Les règles du niveau de la technologie d'information (IT) vérifiant la communication réseau.
- Les règles du niveau de la technologie d'exploitation (OT) vérifiant les comportements du PLC et du CPS.

**Les règles du niveau IT** (TYPE-I.1) vérifiant la communication réseau sont considérés comme des règles de haut niveau, qui sont créées à partir du transformateur en se basant sur l'architecture de l'ICS.

**Pour la surveillance du système du niveau OT**, la génération des règles est plus compliquée. La vérification ici se base sur les valeurs des commandes et des mesures communiquées sur le réseau pour vérifier leur respect des contraintes du système. Suite à ces contraintes le transformateur génère :

- Règles de TYPE-I.2 : Les règles de détection des commandes non-légitimes (les commandes qui ne sont pas déjà définies dans la partie de la spécification).
- Règles de TYPE-I.3 : les règles de détection des mesures hors des intervalles spécifiées pour les variables du système. Ceci permet de détecter non seulement des injections des valeurs erronées, mais aussi des problèmes causés par des commandes légitimes (à état persistant) qui sont déjà appliquées mais qui deviennent non valides suite à l'évolution normale du système).
- Règles de TYPE-I.4 : les règles détectant les violations des invariants sur les quantités en se basant sur l'état actuel du système physique.
- Règles de TYPE-I.5 : les règles de détection du comportement anormal dans le changement des valeurs des quantités du système. Plus précisément, une quantité du système doit changer (suite à un évènement reçu) en suivant la loi de la dynamique du système. Le but de ces règles est de détecter un changement de valeur qui viole cette loi (par exemple, le volume dans un bac doit augmenter d'une façon continue d'une valeur égale aux flux des vanne d'entrée de la bac). Ces règles sont optionnelles, et

doivent avoir une spécification particulière (voir chapitre 3) en créant des nouvelles variables et spécifiant la méthode de calcul de la variable (comme activité dans le modèle de l'automate hybride) et les conditions d'acceptation de la valeur de cette variable (comme invariant dans le modèle de l'automate hybride).

- Règle TYPE-I.6 : Les règles détectant les commandes légitimes (les commandes qui sont définies dans les spécifications, mais qui ne sont pas acceptables suite à l'état actuel du système physique par exemple commande d'ouverture d'une vanne d'entrée sur un bac plein).

Les règles de TYPE-I seront implémentés dans le détecteur classique (un IDS open-source). Ces règles peuvent être regroupées comme suit :

- les règles TYPE-I.1 s'intéressent à la partie à l'en-tête des paquets, et plus spécifiquement, aux adresses sources et destinations des paquets.
- Les règles (TYPE-I.2, TYPE-I.3) nécessitent une simple vérification des valeurs des quantités du système extraites à partir de la communication.
- Les autres types de règles (par l'IDS) nécessitent une connaissance de l'état physique du système, ce qui nécessite d'avoir des règles de TYPE-II permettant de reconstruire l'état physique du système (du côté de l'IDS). Ces règles (règles de TYPE-II) seront implémentées par un module de reconstruction de l'état proposé par notre approche. Nous détaillons ce module dans la section suivante.

Enfin, le transformateur crée des règles de synchronisation et de stockage d'état de TYPE-III pour que l'IDS puisse récupérer les états reconstruits du système à partir du module de reconstruction de l'état.

## 4.3 Récapitulatif des règles générées par le transformateur

Le Tableau 3 présente un récapitulatif des règles générées par le transformateur pour la conception de l'IDS proposé.

Tableau 3: Récapitulatif des règles générées par le transformateur

| Niveau des règles | TYPE | Définition | Implémentation |
|---|---|---|---|
| **IT** | I.1 | Vérifier les nœuds communicants en recherchant ceux qui ne sont pas autorisés en fonction de l'architecture ICS. | Le détecteur classique open-source. |
| | I.2 | Vérifier le timing des paquets entre le contrôleur et le processus. | |
| **OT** | I.3 | Rechercher les paquets Modbus non légitimes en recherchant les violations des spécifications du protocole Modbus (par exemple, un code de fonction non spécifié). | |
| | I.4 | Détecter les mesures en dehors des intervalles spécifiés pour les quantités système. | |
| | I.5 | Détecter les violations d'invariants en fonction de l'état actuel du SCI. | |
| | I.6 | Détecter les changements anormaux des quantités du système physique. Ces règles détectent les changements qui ne sont pas conformes aux lois de la dynamique. Par exemple, le volume dans un réservoir doit augmenter de manière continue d'une valeur égale au débit des vannes d'entrée du réservoir. | |
| | I.7 | Détecter les commandes «légitimes» qui ne sont pas acceptables en fonction de l'état actuel du système physique. Par exemple, la réception d'une vanne d'entrée ouverte sur un réservoir plein. | |
| | II | Reconstruction des états du système physique | Le module d'estimation d'état (le PSE) |
| | III | Synchronisation des états reconstruits crées par le PSE avec l'IDS | Le détecteur classique open-source et le PSE. |

## 4.4 Algorithme de génération des règles

Comme cela a déjà été expliqué, deux catégories de règles sont créées à partir du transformateur : les règles de TYPE-I et les règles de TYPE-II.

### 4.4.1 Règle de TYPE-I

Les règles de TYPE-I sont générées à partir du transformateur pour surveiller la communication et le comportement de l'ICS. La littérature montre l'existence de plusieurs IDSs (tel que Snort, Suricata et Zeek) permettant d'implémenter des règles de détection d'intrusions et sont convenables pour qu'ils soient utilisées dans un système industriel. Notre approche consiste donc à utiliser un de ces IDSs et plus spécifiquement l'IDS Zeek pour implémenter les règles générées par le transformateur. Par conséquent, les règles de TYPE-I seront créées en respectant le format des règles de l'IDS Zeek.

**Génération des règles TYPE-I.1** (les règles sur la communication au niveau IT) **:** pour une communication à sens unique entre deux nœuds (communication du nœud A → B), une règle de type « permis » est créée, qui marque le flux sortant de A et à destination de B comme une communication autorisée.

Pour une communication à double sens (communication entre A ←→B), deux règles de type « permis » seront créées, la première autorisant la communication A → B, et la deuxième autorisant la communication B→A.

Toutes les communications qui ne sont pas marquées comme « permis » sont considérées comme des communications « non permis ». Et par conséquent, une alerte pour ces communications est créée.

**Génération des règles TYPE-I.2** (commandes légitimes)**:** pour générer ces règles, le transformateur extrait la liste des commandes légitimes à partir des modèles du PLC et du CPS. Ensuite, pour chaque commande, une règle de type « permis » est créée marquant le paquet comme acceptable. Enfin, une règle est créée pour générer une alerte pour les paquets non marqués comme « acceptable ».

**Génération des règles TYPE-I.3** (plage de valeur des quantités)**:** ces règles sont optionnelles et sont générées à partir du modèle des variables. En particulier, pour chaque variable ayant

une définition de la plage des valeurs acceptables, le transformateur crée des règles de type « non permis » pour les valeurs hors de l'intervalle spécifiée.

**Génération des règles TYPE-I.4** (vérification des invariants): ces règles sont créées à partir du modèle du CPS pour vérifier les invariants sur les variables. La vérification des mesures est effectuée sur:

1. Une valeur extraite depuis la communication entre le système physique et le contrôleur
2. Une valeur récupérée du côté de module de reconstruction d'état.

**Génération des règles TYPE-I.5** (changement de valeur des quantités) **:** ces règles sont générées à partir des modèles de CPS en se basant sur les états reconstruits du système et sont considérées comme étant cas particuliers des règles TYPE-I.4.

**Génération des règles TYPE-I.6** (acceptation de commande par rapport à l'état du système): ces règles sont créées à partir du modèle du PLC et se basent sur les états reconstruits du système physique. Pour chaque commande présente dans le modèle du PLC (ou bien dans les entrées du modèle de CPS), une règle détectant la violation de la condition de validité de transition est créée pour générer une alerte sur une commande légitime mais non-acceptable suite à l'état actuel de système.

### 4.4.2 Règle de TYPE-II

Les règles de TYPE-II sont générées par le transformateur pour reconstruire l'état du système, c'est-à-dire, pour créer une image de l'état physique de l'ICS de côté du système de surveillance. Plus précisément, le but est d'avoir dans la mémoire active de l'IDS toutes les valeurs actuelles des quantités du système physique. Pour un IDS à base réseau (NIDS), les valeurs des quantités sont généralement récupérées à partir des flux réseau. Cependant, il est possible de trouver des quantités qui ne passent pas dans les flux de communication, ce qui peut causer une vision non complète de l'état physique du système pour un IDS à base réseau. Par exemple, il est possible qu'aucun flux ne soit généré avec la valeur d'une quantité requise

pour l'analyse de l'IDS. Pour résoudre ce problème, nous proposons deux solutions, le choix entre les deux dépend de l'implémentation :

- La première solution consiste à **générer un trafic** en forçant la lecture de toutes les variables présentes dans les invariants depuis un module spécifique connecté au réseau. De cette façon, l'IDS aura l'opportunité de mettre à jour les variables.
- La seconde solution consiste à utiliser des **variables calculées** par un module spécifique qui utilise les activités (spécifiées dans le modèle de l'automate hybride). Ce module est capable de lire les variables nécessaires dans le PLC pour :
    - Initialiser la simulation (ou réinitialiser).
    - Corriger la simulation en fonction des mesures.
    - Lire les variables d'actions.

Par conséquent, le **transformateur** génère une ou deux règles suivant l'option choisie :

- Pour les variables mesurées (dans la mémoire du PLC) :
    - Règle 1 : le transformateur génère une règle pour récupérer les valeurs depuis la mémoire PLC.
    - Règle 2 (optionnel en relation avec la règle TYPE-I.6) : le transformateur génère une deuxième règle pour calculer la variable mesurée (par simulation en) en se servant de l'activité (spécifiée dans le modèle de l'automate hybride) et puis calculer la valeur du résiduelle entre la valeur mesurée et celle simulée.
- **Pour les variables non-mesurables** (par exemple un chronomètre sur une durée max d'application de commande) : le transformateur génère une règle pour calculer la valeur de la quantité (par une simulation qui exécute la règle spécifiée par une ou plusieurs activités dans le modèle d'automate hybride).

Cependant, aucun IDS générique ne permet de traiter ce type de règles (connexion à un PLC) et faire des simulations complexes. Et comme notre solution consiste à ne pas récréer l'IDS à partir du zéro, nous implémentons ces règles par un module extérieur que nous nommons le module de reconstruction d'état (ou bien en anglais, Process State Estimation, PSE).

Par conséquent, nous aurons besoin d'un nouveau type de règles (les règles de TYPE-III expliqué dans la section suivante) qui permettent à l'IDS de récupérer les états reconstruits à partir du PSE.

### 4.4.3 Règle de TYPE-III

Le but principal de ces règles est de permettre une synchronisation des états reconstruits du système physique entre l'IDS (le détecteur classique) et le PSE.

**Pour le PSE**, les règles de synchronisation doivent permettre de recevoir des requêtes des états reconstruit du système physique.

**Pour l'IDS**, les règles doivent permettre de récupérer et stocker l'état courant du système physique.

### 4.4.4 Algorithme d'estimation des variables du système

Le module de reconstruction d'état PSE se base sur les modèles en automates hybrides du PLC et du CPS pour estimer des variables non mesurées par le système. Mais encore, pour calculer les valeurs attendues des quantités suite à l'évolution naturelle du système. Cette estimation se fait en simulant les modèles d'automates hybrides (HA).

L'algorithme d'estimation des variables illustré dans la Figure 18 suit les étapes suivantes :

- Récupération des valeurs des mesures ($Y_i$) et des actions ($U_i$) depuis la mémoire active du PLC (Mémoire Modbus). Ces valeurs sont ensuite analysées et les dernières valeurs sont stockées.
- Application des transitions et mise à jour des places courantes dans les modèles HA.
- Application des activités des places courantes.
- Mise à jour des valeurs des quantités.

À noter que les variables non mesurées (manquantes) du système sont calculées à partir des autres variables existantes et en calculant leurs activités (définies par des fonctions dans les

modèles de l'ICS). Par ailleurs, pour les variables mesurées par le système, deux solutions peuvent être appliquées :

1. Utiliser les variables mesurées par une lecture à partir de la mémoire du PLC.
2. Calculer ces variables par simulation, en commençant la simulation par des valeurs initiales des variables.

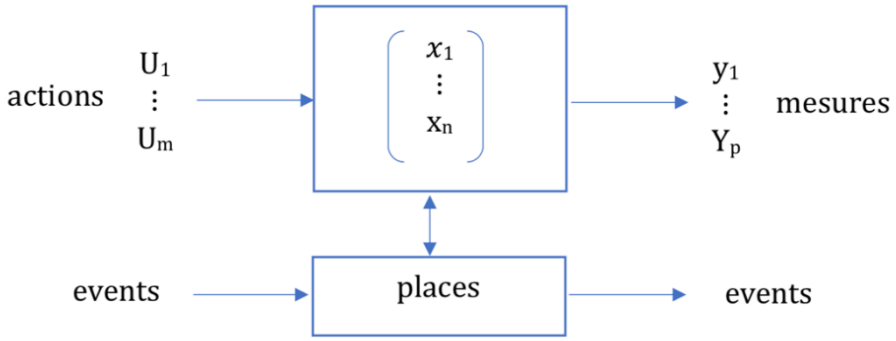



Figure 18: Simulation des modèles

Cependant, l'application de l'une de ces options n'est pas toujours efficace ; d'un part, un ICS est susceptible aux attaques et les variables mesurées peuvent être modifiées par des pirates. Et par conséquent, l'utilisation de ces variables aboutit à des faux résultats. D'autre part, la simulation à longue durée peut causer un écart important par rapport aux variables réelles.

Pour résoudre ce problème, nous utilisons les deux solutions à la fois en appliquant un filtre de correction illustré sur la Figure 19 entre les valeurs mesurées et celles créées par simulation pour avoir une estimation plus fiable. Ce filtre peut être configuré par un paramètre $\alpha$ pour choisir le rang de la valeur mesurée par rapport à la valeur simulée d'une variable $x$ en appliquant la règle suivante :

$$x' = \alpha * x_{sim} + (1 - \alpha) * x_{mes}$$

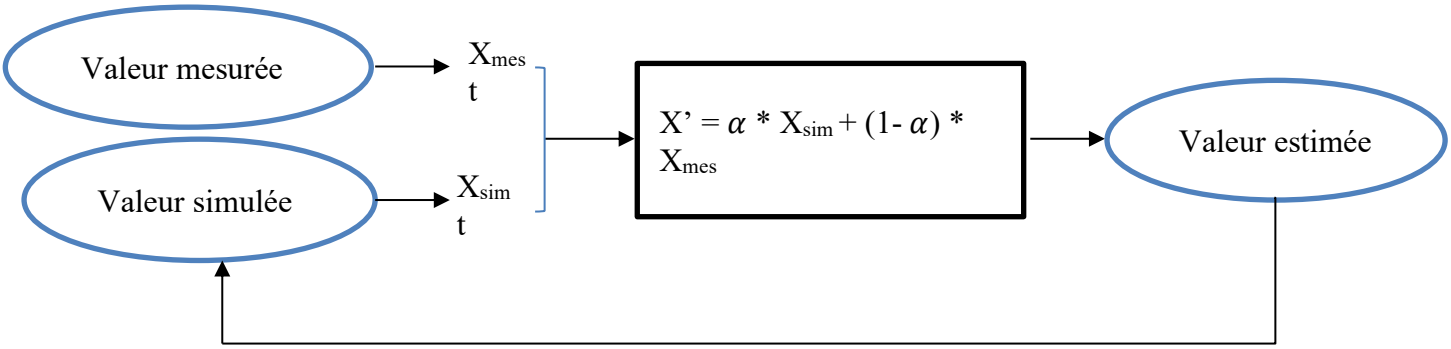



Figure 19: Estimation des valeurs

### 4.4.5 Algorithme de détection de l'IDS

La détection des intrusions possibles sur l'ICS se fait dans notre approche en utilisant un détecteur classique qui implémente les règles générées à partir du module transformateur. Le principe de détection de l'IDS est illustré dans la Figure 20, et consiste à répéter les étapes suivantes :

1. **Synchronisation de l'état IDS avec l'état du système** : dans cette étape, le système de détection d'intrusion extrait les commandes et les mesures de la communication réseau entre SCADA et l'automate. Il obtient également les autres mesures nécessaires du module PSE. Ensuite, le système de détection d'intrusion applique les transitions requises et met à jour l'état actuel du système. Cette étape consiste à
    a. Appliquer une **inspection de haut niveau** des paquets de communication (extraction des en-têtes).
    b. Appliquer une **inspection approfondie** du contenu des paquets de communication entre SCADA et PLC qui consiste à
        i. Identifier les quantités utiles (les mesures et les commandes).
        ii. Stocker les quantités dans la mémoire active de l'IDS.
    c. Reconstruction complète de l'état actuel du système qui consiste à :
        i. Récupérer le complément des valeurs des quantités (par exemple, les variables internes) depuis le module PSE en appliquant les règles de TYPE-III.

ii. Stocker les quantités complémentaires dans la mémoire actives de l'IDS.

2. **Surveillance et vérification** : l'IDS surveille les communications et l'état en continu et vérifie toutes les conditions de validité. L'état du système est considéré comme "non valide" s'il viole au moins l'un des invariants ou la condition de validité de la transition. Cette étape consiste à :
   a. Appliquer les règles de TYPE-I.1 suite à l'inspection de haut niveau
   b. Appliquer les règles de TYPE-I.2-3 pour chaque paquet observé après une inspection approfondie.
   c. Vérifier l'état actuel du système en continue en appliquer les règles de TYPE-I.4-6 suite à chaque mise à jour de l'état du système dans la mémoire du PLC.

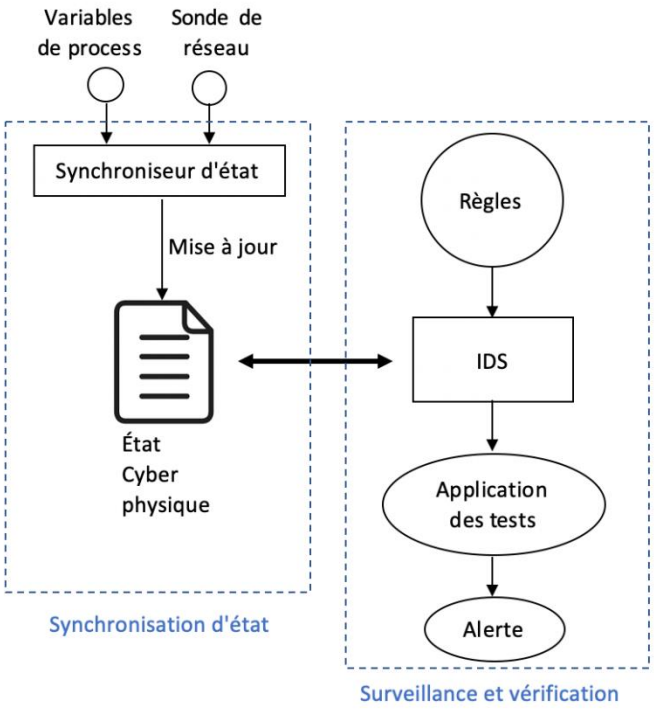



Figure 20: Le principe de détection de l'IDS proposé

## 4.5 Expérimentations et résultats

### 4.5.1 Étude de cas

Nous avons testé notre approche sur l'exemple de réacteur chimique illustré dans la Figure 21. Le réacteur chimique est composé d'un réservoir relié à deux vannes d'entrée et une vanne de sortie que nous appelons respectivement V1 et V2 et Vout. Les trois vannes sont contrôlées par un automate (PLC). Les vannes d'entrée peuvent être ouvertes ou fermées sans modulation. La vanne de sortie Vout est plus ou moins ouverte entre 0 et 100%. Le réservoir est équipé par des capteurs de volume et de température qui fournissent au contrôleur son volume et sa température actuels. Le processus est simulé dans une machine virtuelle connectée au réseau (testbed hybride).

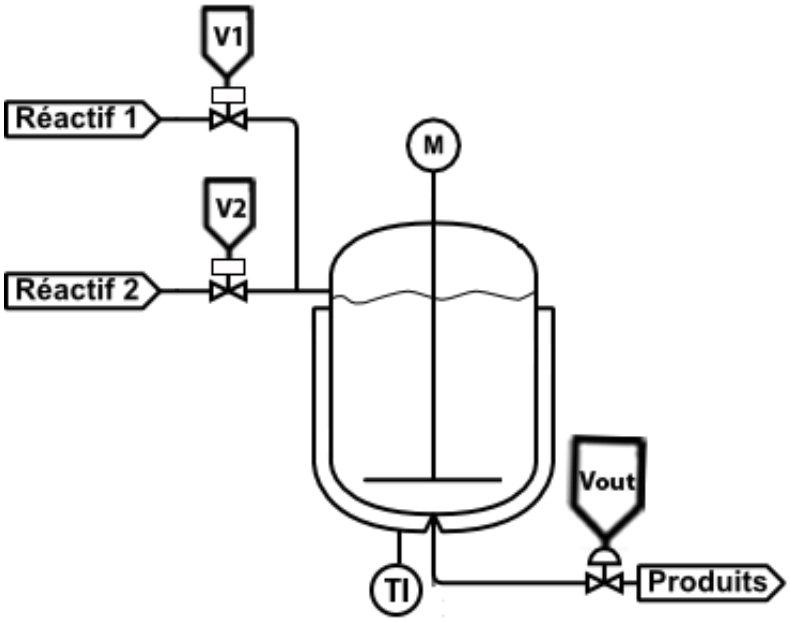



Figure 21: Réacteur chimique

Le PLC doit contrôler le processus pour créer le produit final tout en évitant le débordement du réservoir. La température est contrôlée par le PLC en contrôlant un système de refroidissement. En outre, le PLC fonctionne avec deux modes, automatique et manuel.

Lorsque l'automate est configuré en mode automatique, il contrôle automatiquement le volume et la température du système. Cependant, lorsque l'automate est configuré en mode manuel, l'utilisateur doit s'assurer manuellement de l'état du système. Le Tableau 4 montre les paramètres que l'opérateur peut modifier et vérifier sur l'interface Homme-machine (IHM).

Tableau 4: Les paramètres disponibles pour l'opérateur

| Type | Données | plage de valeurs |
|---|---|---|
| Variables modifiables (commandes de contrôle) | X1 | {0,1} |
| | X2 | {0,1} |
| | Xout | [0-100] |
| | Qc (liquide de refroidissement) | [0-300] |
| | Mode | {0,1} |
| variables en lecture seule (retours système) | Volume | [0-65000] |
| | Température | [0-100] |

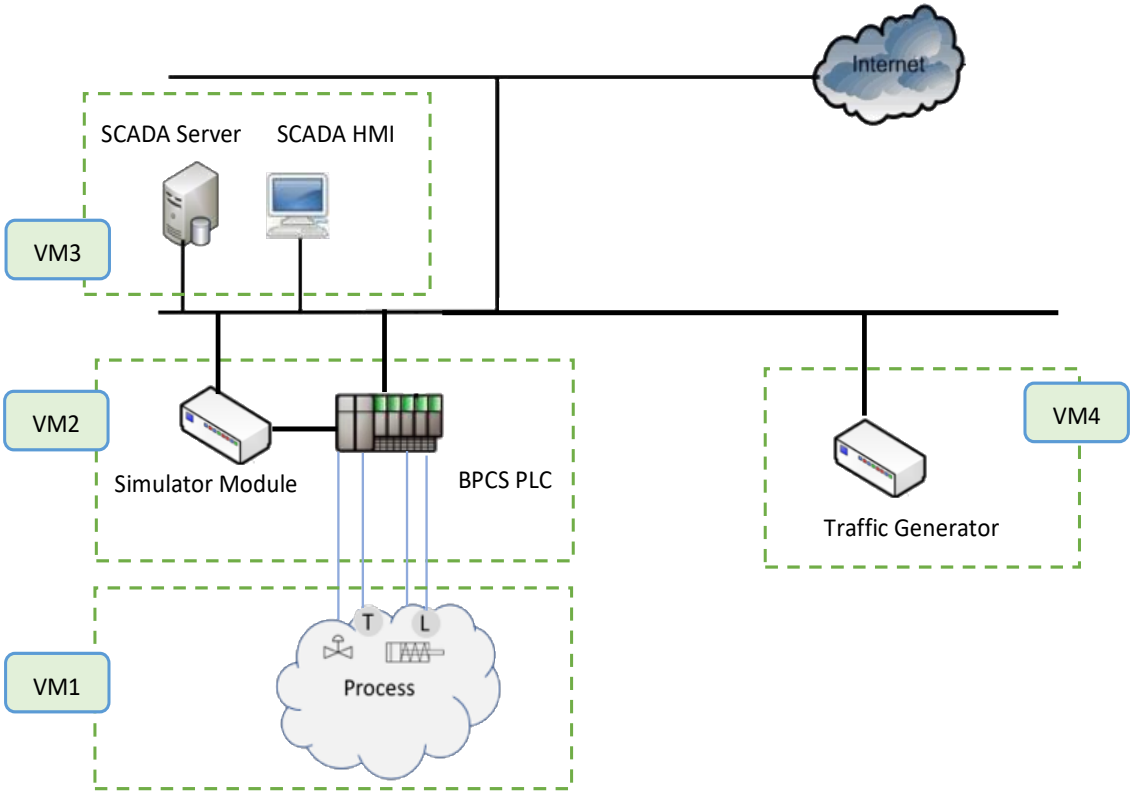



Figure 22: Architecture de l'implémentation

### 4.5.2 Mise en œuvre

Nous avons implémenté l'exemple d'expérience sous forme de simulation distribuée sur quatre machines virtuelles comme représenté dans la Figure 22 en utilisant Ubuntu comme système d'exploitation pour les machines virtuelles :

1. La première machine virtuelle VM1 implémente le logiciel de simulation du process.
2. La deuxième machine virtuelle VM2 implémente un logiciel de simulation du PLC et le logiciel de simulation proposé par notre approche de détection.
3. VM3 implémente le système de contrôle et d'acquisition de données (SCADA) et l'interface Homme-machine (HMI).

4. VM4 implémente un générateur de trafic qui interroge le simulateur pour créer un trafic avec les valeurs de simulation. Ceci permet à l'IDS d'avoir une source supplémentaire pour les variables non mesurées.

Un réseau NAT est utilisé pour la communication entre les machines virtuelles. Le Tableau 5 montre la configuration appliquée pour implémenter l'architecture. La communication entre les quatre VMs est enregistrée dans la VM2 à l'aide du logiciel « tcpdump ». Enfin, tout le traitement de l'IDS est effectué (hors ligne) sur un fichier d'enregistrement de communication de type pcapng créé par le logiciel tcpdump.

Tableau 5: les spécifications de l'implémentation

| Machine physique | Machine virtuelle | Logiciel | Spécification |
|---|---|---|---|
| **M1** | VM1 | Procès | Ubuntu 16, 2GB RAM, 2CPUs |
| | VM2 | PLC + Module de simulation | Ubuntu 16, 2GB RAM, 2CPUs |
| | VM3 | SCADA | Ubuntu 16, 2GB RAM, 2CPUs |
| | VM4 | Générateur de trafic | Ubuntu 16, 2GB RAM, 2CPUs |
| **M2** | - | L'IDS « Zeek » | Mac OS X, 16GB Ram, Core i7 |

### 4.5.3 Modélisation du système cyber-physique

Nous avons modélisé notre système à l'aide de cinq automates hybrides décrivant les comportements de "V1" (la vanne d'entrée 1), "V2" (la vanne d'entrée 2), "Vout" (la vanne de sortie), "Tank" (le capteur de volume du réservoir) et le "contrôleur".

Pour la simplicité de la lecture, nous avons décomposé les modèles sur :

1. La Figure 23 qui montre le modèle de la partie discrète. Le modèle de V2 n'apparaît pas sur la figure, mais est similaire à V1.
2. Le Tableau 6 qui montre les invariants des modèles.
3. Le Tableau 7 qui montre les transitions de modèles.
4. Le Tableau 8 qui montre les activités des variables à état continue.

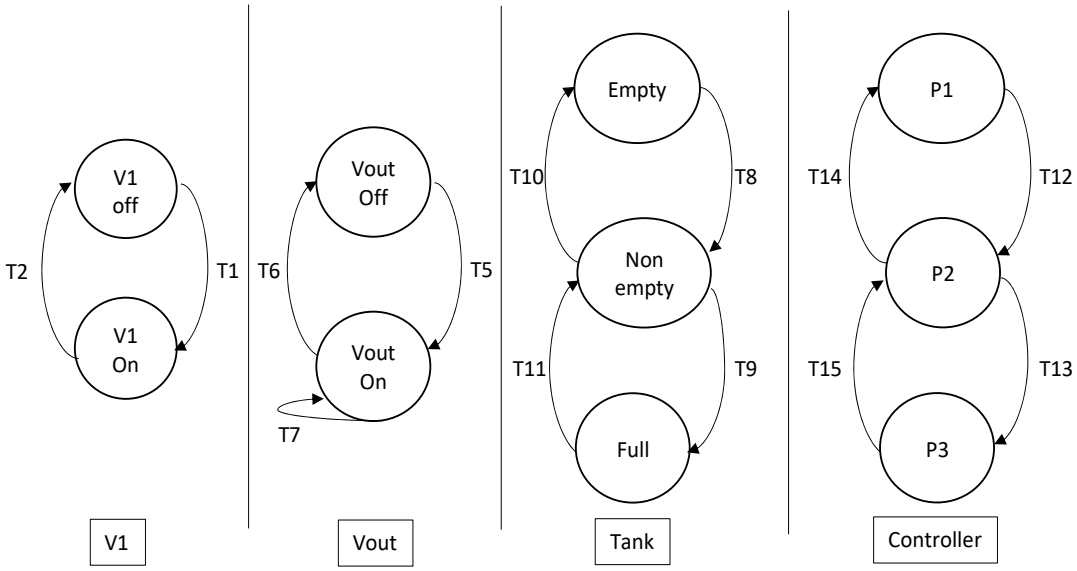



Figure 23: Modèle du CPS

Tableau 6: Les invariants

| # | Activité (L) | Invariant |
|---|---|---|
| **I1** | P1 | V<Vmax |
| **I2** | P1 | Xout==0 |
| **I3** | P1 | V_residual<=2 |
| **I4** | P2 | V<=Vmax |
| **I5** | P2 | H_sim>=0 |
| **I6** | P2 | H_sim<=Hmax |
| **I7** | P2 | V_residual<=2 |
| **I8** | P3 | V<=Vmax |
| **I9** | P3 | H_sim<=Hmax |
| **I10** | P3 | X1==0 |
| **I11** | P3 | X2==0 |
| **I12** | P3 | V_residual<=2 |

Tableau 7: Les transitions

| # | Activité (L) | Évènement reçues | Évènement généré | Guard | Jump | L' | Valid |
|---|---|---|---|---|---|---|---|
| **T1** | V1_off | Open_v1 | V1_open | | Timer_sim1=0 | V1_on | Timer_sim1>3 |
| **T2** | V1_on | Close_v1 | V1_closed | | Timer_sim1=0 | V1_off | Timer_sim1>3 |
| **T3** | V2_off | Open_v2 | V2_open | | Timer_sim2=0 | V2_on | Timer_sim2>3 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **T4** | V2_on | Close_v2 | V2_closed | | Timer_sim2=0 | V2_off | Timer_sim2>3 |
| **T5** | Vout_off | Open_vout | Vout_open | | Timer_sim3=0 | Vout_on | Timer_sim3>3 |
| **T6** | Vout_on | Close_vout | Vout_closed | | Timer_sim3=0 | Vout_off | Timer_sim3>3 |
| **T7** | Vout_on | Open_vout | Vout_open | | Timer_sim3=0 | Vout_on | Timer_sim3>3 |
| **T8** | Empty | V1_open, v2_open | | V_sim>0 | | Non_empty | V_sim>=0, V_sim<Vmax |
| **T9** | Non_empty | | | V_sim>=Vmax | | Full | V_sim==Vmax |
| **T10** | Non_empty | | | V_sim<=0 | | Empty | |
| **T11** | Full | | | V_sim<Vmax | | Non_empty | |
| **T12** | P1 | | | V>0 | | P2 | V>=, V<Vmax |
| **T13** | P2 | | | V>=Vmax | | P3 | V==Vmax |
| **T14** | P2 | | | V<=0 | | P1 | |
| **T15** | P3 | | | V<Vmax | | P2 | |

Tableau 8: Les activités

| # | Activité (L) | Vecteur d'activité |
|---|---|---|
| **F1** | V1_off | [f(timer_sim1) = timer_sim1 + 1] |
| **F2** | V1_on | [f(timer_sim1) = timer_sim1 + 1] |
| **F3** | V2_off | [f(timer_sim2) = timer_sim2 + 1] |
| **F4** | V2_on | [f(timer_sim2) = timer_sim2 + 1] |
| **F5** | Vout_off | [f(timer_sim3) = timer_sim3 + 1] |
| **F6** | Vout_on | [f(timer_sim3) = timer_sim3 + 1] |
| **F7** | Empty | $[f(v\_residual) = \lvert v\_sim - v\rvert,$ $f(v\_residual\_sum) = v\_residual\_sum + v\_residual]$ |
| **F8** | Non_empty | $[f(v\_residual) = \lvert v\_sim - v\rvert,$ $f(v\_residual\_sum) = v\_residual\_sum + v\_residual,$ $f(v_{sim}) = (v + aX1 + bX2 - cXout)\ if\ X1 + X2 + Xout > 0\ else\ V\_sim,$ $f(h\_sim) = alpha * v]$ |
| **F9** | Full | $[f(v\_residual) = \lvert v\_sim - v\rvert,$ $f(v\_residual\_sum) = v\_residual\_sum + v\_residual,$ $f(v_{sim}) = (v + aX1 + bX2 - cXout)\ if\ X1 + X2 + Xout > 0\ else\ V\_sim,$ $f(h\_sim) = alpha * v]$ |

### 4.5.4 Choix de l'IDS

Comme cela à été expliqué dans la partie 3.2, trois systèmes de détection d'intrusion ont été considérés comme base pour créer le détecteur d'anomalie : Snort, Suricata et Zeek. Cependant, le langage utilisé par les IDS Snort et Suricata (Snort-based rules) n'est pas adapté pour une détection nécessitant la vérification de plusieurs paquets ; et par conséquent, ne permettent pas de définir des algorithmes de détection avec état (stateful detection).

En revanche, pour une détection avec état (stateful detection), Zeek fournit un moyen puissant d'utiliser des variables globales dans son langage de script. Ces variables globales peuvent être de n'importe quel type supporté, ou même être une structure définie par l'utilisateur. Seules les fonctions, les hooks et les gestionnaires d'événements ne sont pas autorisés à utiliser le mot-clé global. Une fois la variable globale déclarée, elle peut être utilisée dans tous les scripts Zeek chargés qui sont chargés. De plus, le langage de script **Zeek** permet d'utiliser ses variables comme vous le feriez dans tout autre langage de script en utilisant un grand nombre de fonctionnalités de pré-construction. Cela permet de suivre facilement les états du système.

### 4.5.5 Résultat de la transformation du modèle

#### 4.5.5.1 Règles générées pour l'IDS

Pour cette expérience, nous avons choisi d'utiliser l'IDS opensource Zeek pour mettre en œuvre les règles générées par notre module générateur pour les modèles de spécification du CPS. Ces règles générées automatiquement respectent le langage de règles Zeek et contiennent les éléments suivants :

1. **Les règles définissant les variables globales** : deux variables globales "current_locations" et "state_variables" sont créées pour stocker les emplacements actuels des modèles CPS et les dernières valeurs des variables d'état.
2. **Règles d'extraction de commandes et de mesures à partir de requêtes de lecture et d'écriture Modbus** : étant donné que nous utilisons le protocole Modbus pour la communication entre SCADA et un automate, notre générateur a utilisé les événements

```
event modbus_write_single_register_response(c: connection, headers: ModbusHeaders, address: count, value: count){
    local t = network_time();
    local variable_name = address_state_variables[address];
    state_variables[variable_name]$value= value;
    state_variables[variable_name]$value_time= network_time();
}
```

Liste de code 1: Règles générées pour l'extraction de commandes SCADA

Zeek pour extraire des commandes et des mesures de la communication. Pour cette étude de cas, les commandes SCADA sont extraites des événements *modbus_write_single_register_response* et *modbus_write_multiple_registers_request*. Alors que les mesures sont extraites des événements *modbus_read_holding_registers_request* et *modbus_read_holding_registers_response*. Le code présenté dans la Liste de code 1 montre les règles générées pour extraire les commandes SCADA de la communication réseau.

```
if((state_variables["v"]$value>=Vmax) && location_in_locations("p2", locations) >=0 ){
      i = location_in_locations("p2", locations);
      locations[i] = "p3";
      if (!(state_variables["v"]$value==Vmax)) {
         generate_alarm(message);
      }
}
```

Liste de code 2: Règles générées pour estimer de nouveaux emplacements

3. **Règles pour l'estimation de l'état et la validation des transitions :** après la mise à jour des variables d'état, l'emplacement actuel du système est ré-estimé en appliquant les transitions. De plus, les transitions de validité sont vérifiées. L'exemple illustré sur la Liste de code 2 montre la règle générée pour la transition T13. Où *"location_in_locations("p2", locations)"* est une fonction permettant de vérifier si "p2" se trouve dans les emplacements actuels du CPS.

```
event modbus_write_multiple_registers_request(c: connection, headers: ModbusHeaders, start_address:
count, registers: ModbusRegisters){

   if(!is_scada(c$id$orig_h) !is_plc(c$id$resp_h){

      genAlert("unauthorized src or dest");

   }

}
```

Liste de code 3: Règles générées pour les autorisations de source et de destination

4. **Règles vérifiant l'autorisation de la source et de la destination** : pour chacun des événements de lecture et d'écriture Modbus, notre génération a généré une règle

```
if(location_in_locations("p2", locations) >= 0) {

   if(!(

      state_variables["v"]$value<=Vmax &&

      state_variables["h_sim"]$value>=0 &&

      state_variables["h_sim"]$value<=Hmax &&

      state_variables["v_residual"]$value<=2)

      ){

      generateAlarm(message);

   }

}
```

Liste de code 4: Règles générées pour la vérification des invariants

vérifiant que la source et la destination IP figurent dans la liste des nœuds autorisés à communiquer. La liste de codes la Liste de code 3 montre un exemple de vérification de l'autorisation d'écrire plusieurs registres.

5. **Règles vérifiant les invariants** : la vérification des invariants est appelée après le changement d'une variable d'état. En outre, elles sont appelées à plusieurs reprises en utilisant une horloge. Les règles de la Liste de code 4 affichent les règles générées pour vérifier les invariants I4, I5, I6 et I7.

### 4.5.6 Évaluation de l'IDS

Pour évaluer l'approche proposée, nous avons créé plusieurs scénarios d'attaque, comme c'est indiqué dans le Tableau 9.

La détection d'attaque pour le système par notre IDS est effectuée en analysant la source de données à l'aide du processus suivant : le système IDS recherche une source ou une destination IP non autorisée (niveau de vérification IP), puis effectue une analyse sémantique afin de rechercher les transitions ou invariants non-valides.

Tableau 9: Les attaques appliquées

| # | Attaque | Description |
|---|---|---|
| **1** | Variation brutale de Stuxnet | Composants épuisés suite à une répétition rapide d'actions acceptables |
| **2** | Compromettre les mesures de capteur | Compromettre un capteur pour donner des mesures erronées à l'automate |
| **3** | Compromettre les vannes | Compromettre une Vanne en envoyant des commandes externes directement. |
| **4** | Corruption de commande | Empêcher certaines commandes pour conduire le système à des états critiques. |
| **5** | Corruption de mesure | Tromper l'utilisateur pour conduire le système à un état inacceptable. |
| **6** | DoS sur le PLC | Empêcher l'envoi de commandes et la réception de mises à jour à partir du PLC. |

| | | |
|---|---|---|
| 7 | DoS sur le PLC basé sur l'inondation | Inonder l'automate avec de fausses commandes pour écraser les commandes légitimes. |

#### 4.5.6.1 Attaque N1: Variation brutale de Stuxnet:

Dans cette attaque, le pirate informatique utilise une machine SCADA pour saboter la vanne d'entrée v1 en répétant les commandes d'ouverture et de fermeture de V1 (chaque seconde) comme c'est illustré sur la Figure 24. Cette attaque est également appelée attaque de séquence et a été à l'origine de "l'effet de coup de bélier".

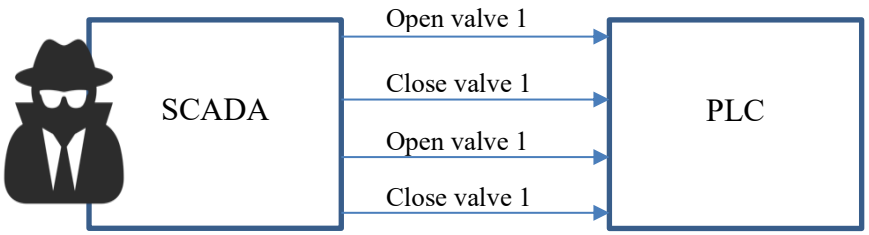



Figure 24: Variation brutale de commande

Lorsque cette attaque est appliquée, les événements open_v1 et close_v1 sont envoyés plusieurs fois par seconde, ce qui entraîne l'application des transitions T1 et T2 chaque seconde. Cependant, les conditions de validité de T1 et T2 sont violées, car Timer_sim durait moins de trois secondes et par suite notre système IDS considère ces événements comme des attaques. Notez que cette Timer_sim1 ne figure pas dans les informations transmises entre SCADA et le PLC et est donc créée par notre module de simulateur et envoyée au PLC.

#### 4.5.6.2 Attaque N2: Compromission des mesures de capteurs:

Lors de cette attaque, les mesures renvoyées pour le capteur de volume sont modifiées en ajoutant 100 aux valeurs réelles. Le scénario de l'attaque appliquée est le suivant : le pirate a attendu que l'utilisateur SCADA applique une commande "open_v1" et a regardé le volume

augmente jusqu'à atteindre la valeur 100, puis a appliqué l'attaque. En conséquence, le graphe du volume était tel qu'illustré dans la Figure 25. La valeur mesurée a changé de manière inattendue de 100 à 200, puis continue à augmenter comme spécifié dans l'activité F8. Ce changement inattendu dans la mesure du volume (le coup d'œil) a violé l'invariant I7 et provoqué la génération de l'alarme. Ici, nous notons que notre IDS a généré une seule alarme. L'alarme concernait uniquement la première valeur compromise, puis les valeurs du volume suivant étaient considérées comme acceptables, car elles confirmaient l'activité f (v_sim) de F8, qui repose sur la dernière valeur du volume mesuré " v "calculer la prochaine valeur attendue de v_sim. Toutefois, cette activité peut être modifiée pour calculer le v_sim attendu en utilisant sa dernière valeur au lieu de " v ". Ainsi, l'IDS génère des alarmes pour chaque mesure compromise au lieu de la première uniquement.

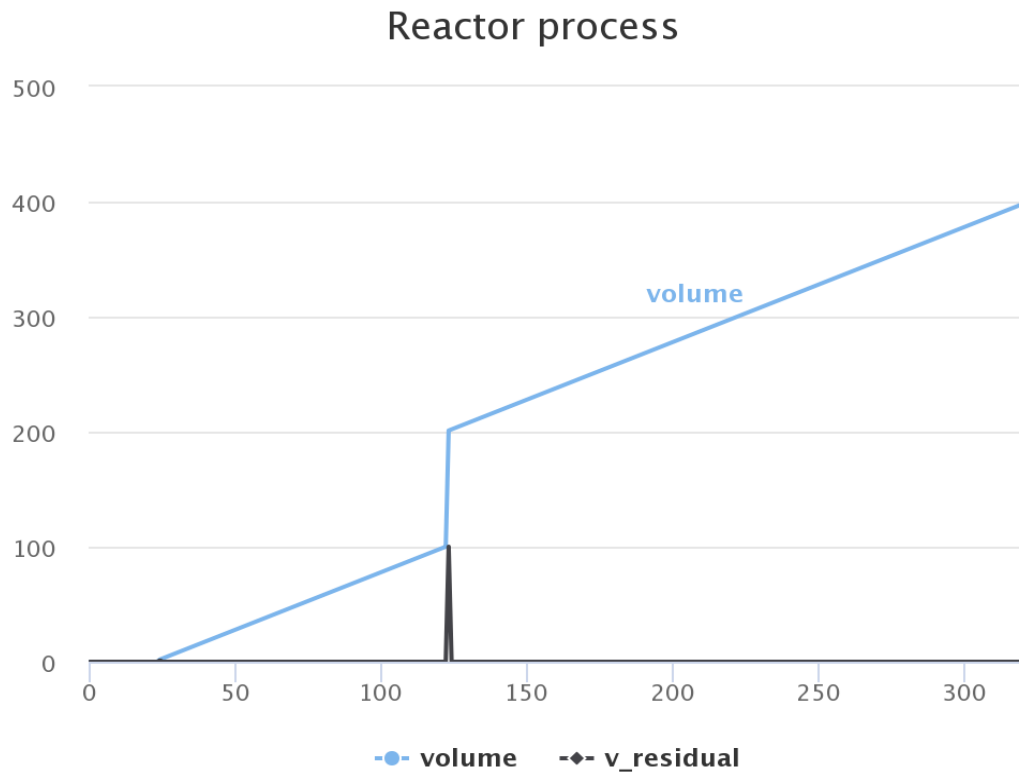



Figure 25: Capteur compromis

#### 4.5.6.3 Attaque N3: Compromission des commandes aux actionneurs

Cette attaque implique de compromettre l'une des vannes d'entrée (vanne V1) pour pouvoir être ouverte sans recevoir de commandes de l'automate. Par conséquent, lorsque nous avons appliqué cette attaque, l'HMI a montré que le volume augmentait alors que les vannes d'entrée étaient fermées. La Figure 26 montre le graphique du volume mesuré (volume) et du volume attendu (sim_volume). En analysant le résultat de l'IDS, nous avons constaté qu'à l'époque t < 200, les emplacements actuels étaient = ["v1_off", "v2_off", "vout_off", "vide", "p1"]. En conséquence, aucune activité n'est appliquée sur le v_sim, ce qui fait que la valeur attendue de v_sim reste égale à zéro. Cependant, entre 200 et 200 minutes, le volume commence à

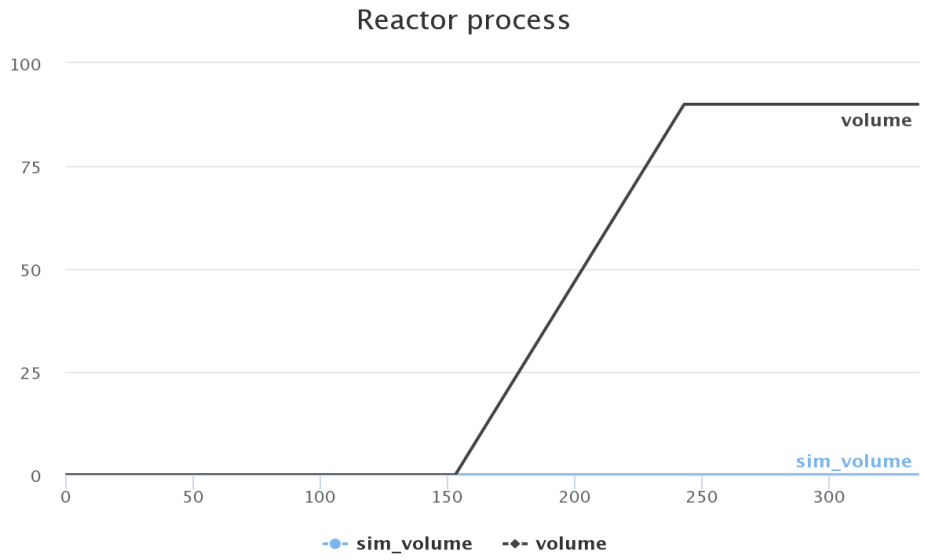



Figure 26: Vanne compromise

augmenter, mais aucun événement n'est reçu, et v_sim n'a pas changé (considérez l'activité F8) et les emplacements actuels n'ont pas changé. De plus, l'application de l'activité f (v_residual) de l'activité F8 a généré une valeur > 2, qui violait l'invariant I3 et provoquait la création d'alarmes.

#### 4.5.6.4 Attaque N4: Corruption de commande SCADA

Nous avons appliqué une MITM Attack pour empêcher l'utilisateur SCADA de fermer la vanne v1 ou de mettre le système en mode automatique, comme montre la Figure 27. L'IDS a détecté cette attaque après avoir violé les invariants I8, I9 et I10. La détection tardive est due aux raisons suivantes: cette attaque est effectuée sur les commandes avant leur arrivée à l'IDS, l'IDS n'a pas détecté cette corruption (l'IDS n'a pas reçu l'événement). Cependant, comme l'IDS vérifie toujours les invariants, trois invariants sont violés lorsque le réservoir approche du débordement. Par conséquent, l'IDS détecte toujours le problème avant qu'il ne provoque des dommages critiques au système.

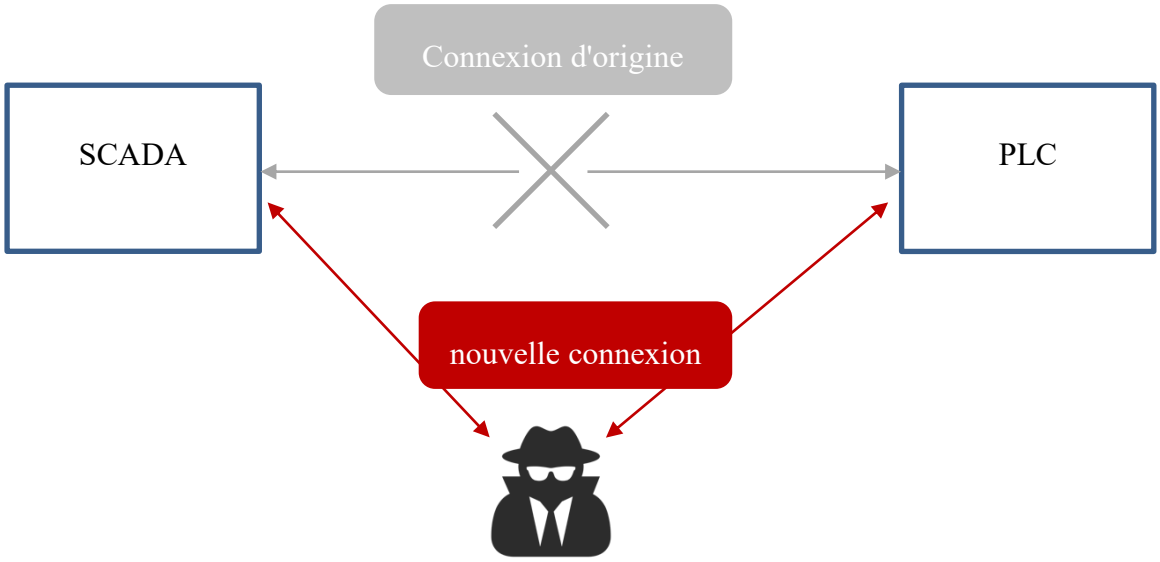



Figure 27: MitM entre SCADA et contrôleur

#### 4.5.6.5 Attaque N5: Corruption de mesure :

Une attaque MITM a été effectuée pour modifier les mesures envoyées de l'automate au SCADA comme c'est montré sur la Figure 27.

Si le volume dans le réservoir > 30000 et que la vanne V1 est ouverte:

modifier les mesures envoyées depuis le PLC au SCADA pour montrer que:

- la vanne V1 est fermée.
- le volume du réservoir = constant (30000).

Lorsque l'IDS a analysé la source de données, il a constaté que le volume ne changeait pas (dans les paquets réseau). Cependant, le PSE calculait la hauteur du volume dans le réservoir, en fonction de la valeur du volume lue dans la mémoire de l'automate. Par conséquent, la comparaison de la hauteur générée et de la hauteur maximale dans l'invariant a permis à l'IDS de détecter que le système ne se comportait pas comme prévu (la hauteur estimée est supérieure à la hauteur maximale autorisée). Ensuite, l'IDS a déclenché une alarme.

#### 4.5.6.6 Attaque N6: DoS sur PLC

Dans cette attaque, le pirate attend que l'utilisateur ouvre une vanne d'entrée, puis effectue une attaque MITM pour empêcher l'automate de recevoir un autre paquet SCADA comme c'est montré sur la Figure 28. Par conséquent, l'utilisateur en contrôle perd son contrôle et le système.

Cette attaque est créée par l'empoisonnement ARP (ARP poisoning) du serveur SCADA et du PLC pour envoyer le trafic via la machine pirate sans transférer de paquets entre le serveur SCADA et le PLC, afin que les paquets envoyés n'atteignent pas leur destination.

L'analyse IDS montre que l'attaquant a utilisé une adresse IP autorisée (l'adresse IP du SCADA HMI) et qu'il n'a pas violé les règles IP ou les spécifications du protocole.

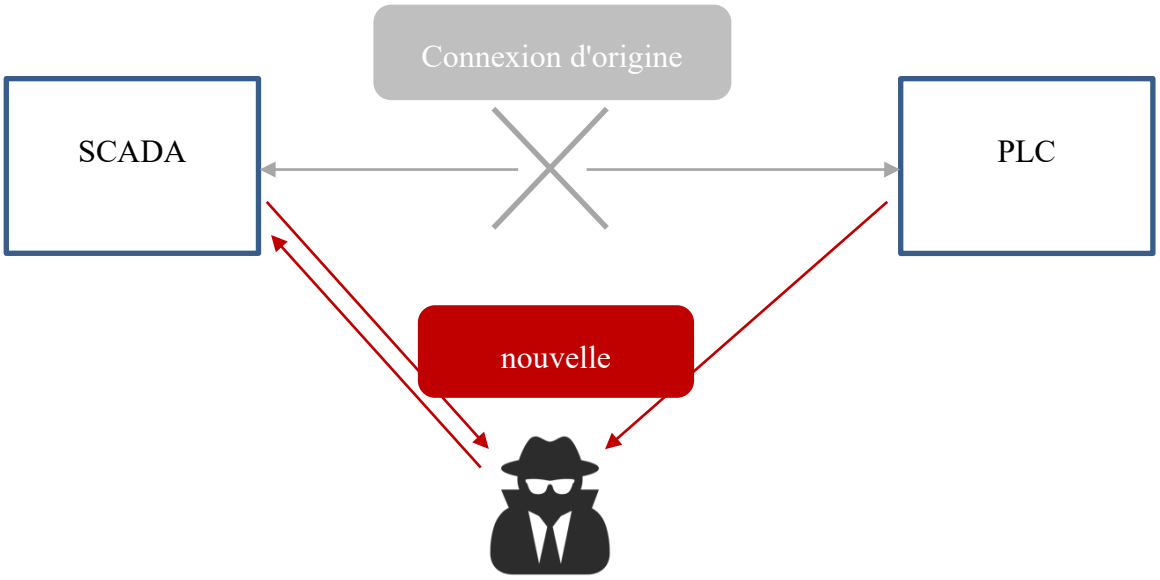



Figure 28: MitM Dos sur l'automate

Toutefois, même si le pirate informatique a arrêté le trafic légal entre le SCADA et l'API, l'IDS a poursuivi son analyse sur la base des données créées par le PVRM, lequel lit les données directement du PLC et les envoie à l'IDS (l'IDS reçoit toujours le v_sim, et h_sim). Et par conséquent, l'IDS a détecté une violation invariante (h_sim <= Hmax n'est pas vérifiée) et a déclenché une alarme.

#### 4.5.6.7 Attaque N7: DoS sur le PLC par surcharge de requêtes

Une attaque par flooding Modbus est une attaque par déni de service (DoS) basée sur une surcharge de requêtes, qui empêche le SCADA de contrôler l'automate comme c'est montré dans la Figure 29. En utilisant cette attaque, l'attaquant n'empêche pas la commande SCADA d'atteindre le contrôleur, mais il envoie un plus grand nombre de commandes pour couvrir les commandes légitimes à l'aide d'un programme illégal. Nous avons appliqué cette attaque à notre cas d'utilisation. Cependant, dans cet exemple, l'IDS ne considère pas que SCADA et l'automate comme sources autorisées. En conséquence, lorsque l'IDS a analysé le trafic, il a

détecté une source IP non autorisée envoyant des paquets à l'automate et a déclenché une alarme.

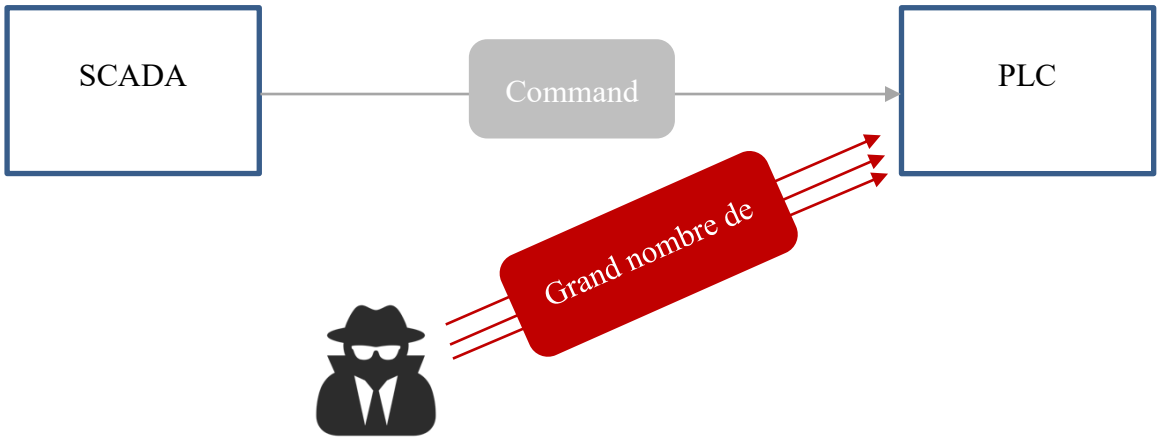



Figure 29: DoS sur le PLC basé sur l'inondation

## 4.6 Bilan du chapitre

Ce chapitre explique le principe de notre approche de conception d'IDS. Cette approche de conception se fait initialement par trois étapes : (i) la modélisation des connaissances de l'IDS, (ii) transformation des connaissances en des règles de l'IDS et du module de simulation de l'ICS et (iii) l'implémentation de l'IDS en ligne. La modélisation des connaissances de l'IDS est présentée dans le chapitre précédent. L'algorithme de transformation des modèles d'entrées en des règles de détections et des règles de simulations a été présenté dans ce chapitre. Une explication des règles résultantes par la transformation et ainsi expliqué. Enfin, ce chapitre est conclu par un exemple explicatif de l'application de l'approche sur un système de contrôle industriel simple.

# Chapitre 5 Application

## 5.1 Introduction

Afin de valider notre approche, nous commencerons par définir le périmètre de l'étude expérimentale. Les connaissances sur l'ICS sont représentées par un modèle générique de l'ICS composée d'un graphe de l'architecture et des automates hybrides décrivant le comportement du PLC et du CPS. Ensuite, le modèle de l'ICS est transformé en règles pour l' IDS « Zeek », ainsi qu'en un module d'estimation d'état du process physique (PSE).

Les sondes de l'IDS et le module d'estimation d'état du process (PSE) sont placés au plus proche du process physique (entre les niveaux 0 et 1 de l'architecture Purdue). Ainsi, le module PSE construit l'état du système en récupérons les valeurs du process depuis la mémoire du PLC.

Le but de cette approche est de pouvoir prendre en compte tout attaque qui cherche à impacter la partie opérative, quel que soit son type : Deny of Service (DoS), Man-in-the Middle (MitM), False Data Injection (FDI), changement de l'ordre de commande, etc. Nous retenons particulièrement les attaques aléatoires, directes, séquentielles, temporelles ou entre les injections de fausses commandes (depuis le PLC vers le process) et de fausses données (remontée depuis le process vers le PLC).

Enfin, l'ensemble de ces éléments forme le périmètre dans lequel est testé l'approche proposée.

## 5.2 Implémentation de l'ICS (étude de cas)

### 5.2.1 Description du process

Cette partie explique un exemple d'utilisation complet d'un système à cuves utilisé pour évaluer notre approche de détection d'intrusions.

Il s'agit d'un système à cinq cuves comme illustre la Figure 30. Le système est composé de cinq cuves, trois d'entre elles $C_A$, $C_B$ et $C_C$ sont considérés comme étant de capacité infinie (elles sont automatiquement remplies par un processus non décrit), et contiennent respectivement les produits A, B et C. Ces trois cuves déversent leurs produits dans deux réservoirs de mélange $C_1$ et $C_2$ afin de fabriquer le produit. Le processus de production est défini par les étapes suivantes :

1. Remplir les produits d'entrée dans une cuve de mélange.
2. Mélanger les produits.
3. Vider le mélange qui constitue le produit final.
4. Laver la cuve et se débarrasser de l'eau de lavage.

L'étape de remplissage est déclenchée en choisissant en premier lieu le(s) cuve(s) de mélange, puis en cliquant sur le bouton « Start ». Une cuve (C1 ou C2) ou bien les deux à la fois peuvent être choisit pour mélanger les produits A, B et C. Ceci se fait par l'ouverture d'une des vannes $V_{C1\_in}$ ou $V_{C2\_in}$, ou bien les deux à la fois. Ensuite, le remplissage du produit se fait par l'ouverture des vannes de remplissage $V_1$, $V_2$ et $V_3$ et dans le même ordre que celui mentionné. Le système de contrôle ouvre la vanne $V_1$ jusqu'à atteindre la valeur « Setpoint A » dans les cuves choisies pour le mélange, puis ferme $V_1$ et ouvre $V_2$ jusqu'à attendre la valeur « Setpoint A + Setpoint B ». Enfin, il ferme $V_2$ et ouvre $V_3$ jusqu'à atteindre la valeur « Setpoint A + Setpoint B + Setpoint C ».

L'étape de mélange est considérée déclenchée du moment où on commence à avoir deux produits (ou plus) dans la cuve de mélange. Ainsi, le mélange se termine après un certain temps d'attente (5 secondes) suite au remplissage des produits d'entrée.

L’étape de vidage est réalisée par l’ouverture d’une des vannes $V_{out1}$ ou $V_{out2}$ et selon la cuve de mélange utilisée. La sortie des vannes de vidage représente le produit final.

Une étape de lavage de la cuve est nécessaire après chaque production du produit final. Cette étape est déclenchée en ouvrant une vanne $V_{water1}$ ou $V_{water2}$ et selon la cuve du mélange choisie, jusqu’à que la vanne du mélange soit remplie. Puis, le système se débarrasse de l’eau de lavage en ouvrant une vanne de sortie de l’eau $V_{waste1}$ ou bien $V_{waste2}$. La cuve de mélange sera disponible pour un nouveau produit après avoir sorti l’eau de la cuve de mélange.

Des capteurs de niveau sont placés dans chacune des cuves du mélange et permettre de connaître le niveau du produit final. Deux capteurs de niveau $H_1$ et $H_2$ sont utilisés dans ce système pour mesurer respectivement le volume de produit dans les cuves $C_1$ et $C_2$.

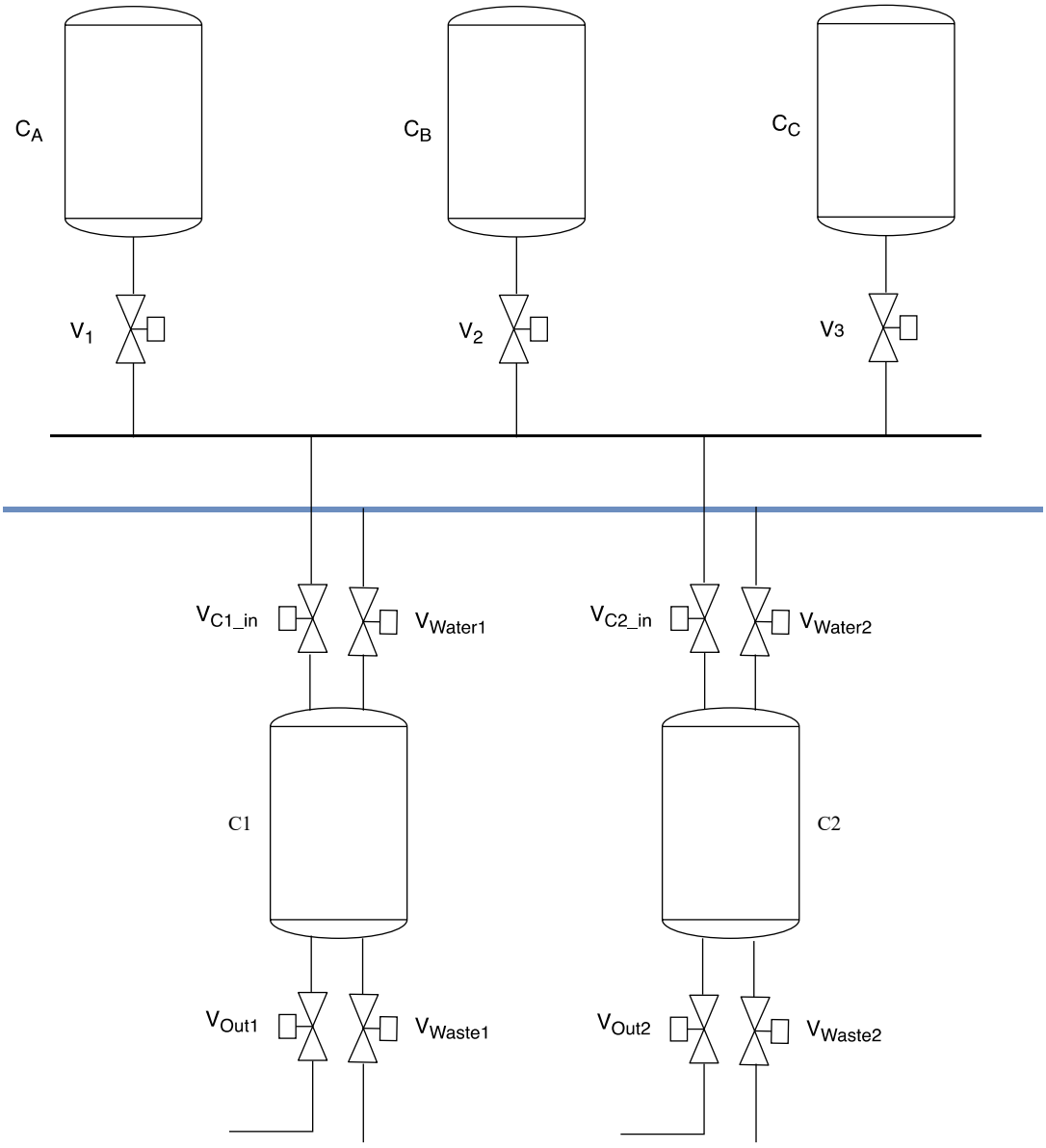



Figure 30: Le process physique

### 5.2.2 Architecture et implémentation de l'ICS

Le système industriel est implémenté par un testbed hybride ; l'ICS est simulé par un réseau de machines virtuelles qui utilisent le protocole Modbus/TCP pour avoir des communications réelles sur le réseau.

L'architecture de l'implémentation de l'ICS est illustrée sur la Figure 31. Elle comprend cinq machines virtuelles.

- Sur VM1 : le système de supervision est implémenté en utilisant l'outil mySCADA.
- Sur VM2 : le PLC est implémenté en utilisant l'outil OpenPLC et codé en utilisant le langage SFC.
- Sur VM3 : la simulation du process est implémentée.
- Sur VM4 : le module d'estimation du l'état (PSE) est implémenté.
- Sur VM5 : un module pour la génération du trafic (pour recevoir les états du PSE) est implémenté, ainsi que l'IDS Zeek. Le but du module de génération de trafic est d'interroger d'une façon continue le PSE, afin de créer un nouveau trafic contenant les états du système créés par le PSE, ce qui permet à l'IDS de récupérer ces informations.

Pour la communication entre les machines virtuelles, deux réseau « NAT Network » sont installés :

1. NAT-0 : pour la communication entre SCADA et PLC.
2. NAT-1 : pour la communication entre le PLC, le PSE, le process et le module de génération du trafic.

À noter que le PLC (VM2) est configuré avec deux NICs (Network interface card), la première pour se connecter sur NAT-0 et la deuxième pour se connecter sur NAT-1.

Ainsi, l'IDS est placé sur le réseau entre les niveaux 1 et 0 de l'architecture Purdue. Par conséquent, les sondes de l'IDS sont capables de récupérer la communication entre le système de contrôle et le process, ainsi que les états reconstruits à partir du module PSE.

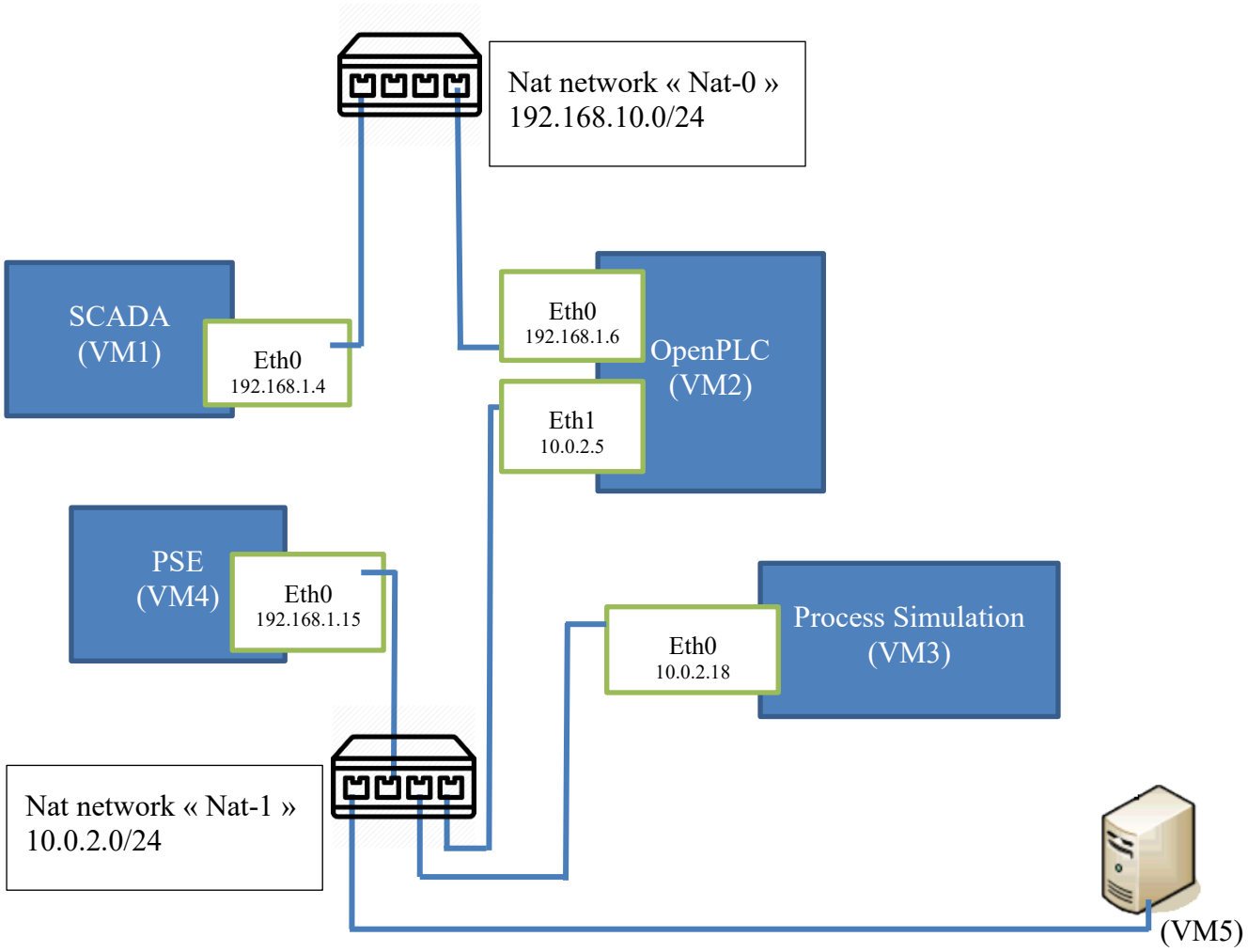


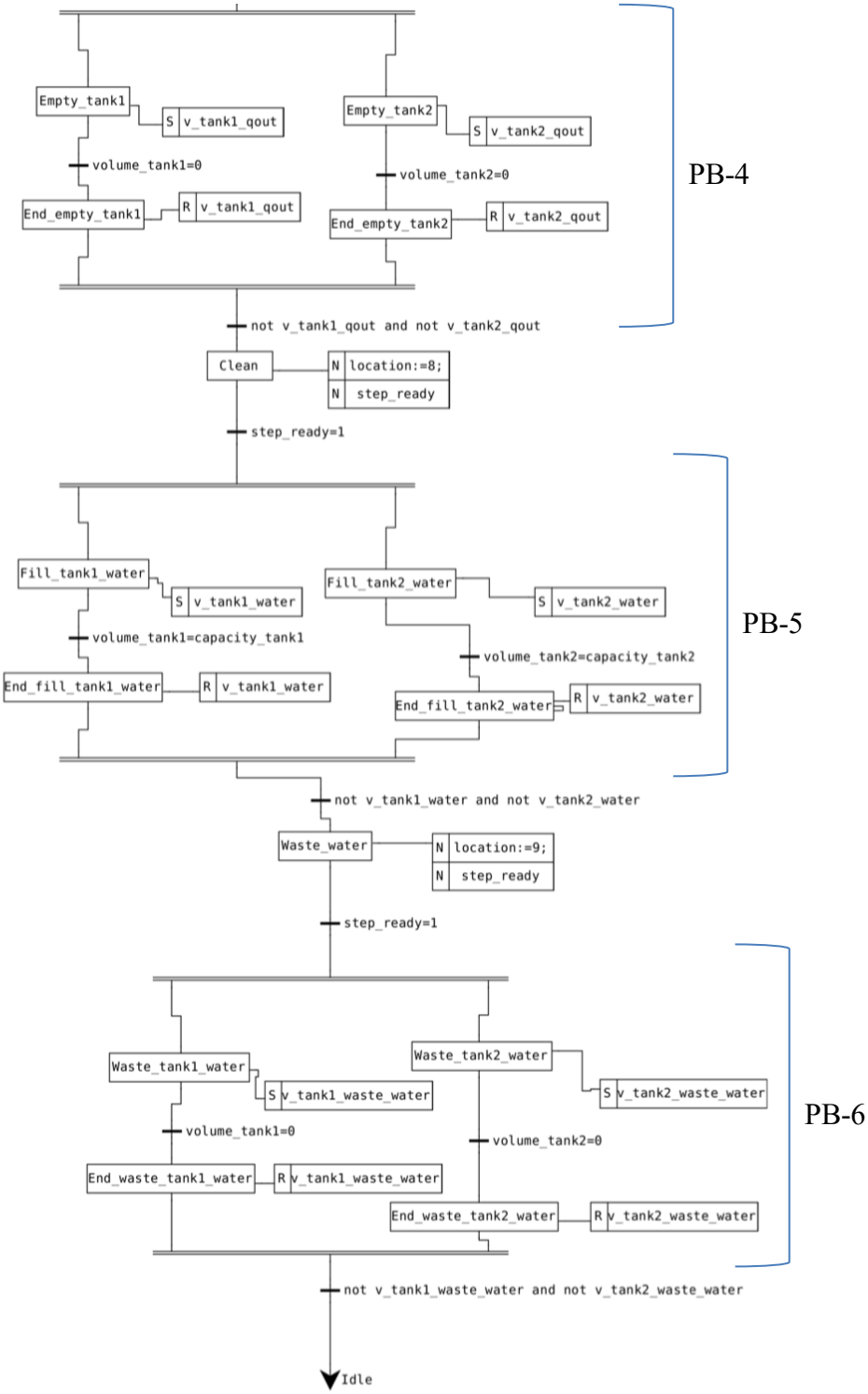

Figure 31: Architecture du process physique

### 5.2.3 Code et implémentation du PLC

Le contrôleur est codé en utilisant le langage SFC afin d'appliquer le contrôle du process comme cela est décrit dans la section précédente. Le code du PLC est composé de deux parties initiales :

1. La définition des variables illustrées sur la Figure 32.

2. Le programme séquentiel illustré sur les Figure 33, Figure 34 et




.

L'explication du programme SFC est la suivante:

- Le programme SFC commence par l'étape « Idle ». Dans cette étape, le PLC remet les valeurs des variables setpoint_A, setpoint_B, setpoint_C, is_tank1_chosen, is_tank2_chosen et start_sc à zéro. Ainsi, le PLC met step_ready à la valeur booléenne vrai (true). La mise de la valeur de la dernière variable (step_ready) à vrai permet d'achever la condition nécessaire pour franchir la première transition pour atteindre l'étape « choose_tank ».
- Dans l'étape « choose_tank », le PLC attend la réception du choix des cuves de mélange ainsi que la réception du bouton start pour atteindre l'étape de validation du choix « choice_validation ».
- L'étape « choice_validation » permet de vérifier le choix des cuves à remplir. Cependant, si aucune cuve n'est choisie, le PLC revient vers l'étape « choose_tank ». Si non, le programme franchit la transition suivante et atteint l'étape « Fill_A », où le PLC vérifie les cuves choisit pour les remplir.
- Le bloc parallèle commence et permet de remplir les cuves choisies, pour les remplir par le produit A, en ouvrant les vannes v1 et v_tank1_in et / ou v_tank2_in (selon le choix des cuves) jusqu'à atteindre les valeurs spécifiées par la valeur du « setpoint_A ». À la fin de ce bloc parallèle, le PLC ferme les vannes v_tank1_in et / ou v_tank2_in, ensuite ferme la vanne v1.
- De même, le PLC remplit les autres produit (B et C). Puis, il attend un certain temps (5 secondes) dans l'étape mixing.
- Ensuite, le PLC lave les cuves C1 et C2 en les remplissant par l'eau et se débarrassant des produits.
- Enfin, le PLC retourne vers son état initial.

| # | Name | Class | Type | Location | Initial Value | Option | Documentation |
|---|---|---|---|---|---|---|---|
| 1 | v1 | Local | BOOL | %QX0.0 | | | Modbus Register 0 (PLC output) to char |
| 2 | v2 | Local | BOOL | %QX0.1 | | | Modbus Register 1 |
| 3 | v3 | Local | BOOL | %QX0.2 | | | Modbus Register 2 |
| 4 | v_tank1_in | Local | BOOL | %QX0.3 | | | Modbus Register 3 |
| 5 | v_tank1_water | Local | BOOL | %QX0.4 | | | Modbus Register 4 |
| 6 | v_tank1_qout | Local | BOOL | %QX0.5 | | | Modbus Register 5 |
| 7 | v_tank1_waste_wa | Local | BOOL | %QX0.6 | | | Modbus Register 6 |
| 8 | v_tank2_in | Local | BOOL | %QX0.7 | | | Modbus Register 7 |
| 9 | v_tank2_water | Local | BOOL | %QX1.0 | | | Modbus Register 8 |
| 10 | v_tank2_qout | Local | BOOL | %QX1.1 | | | Modbus Register 9 |
| 11 | v_tank2_waste_wa | Local | BOOL | %QX1.2 | | | Modbus Register 10 |
| 12 | volume_tank1 | Local | INT | %QW13 | | | Modbus Register 13 |
| 13 | volume_tank2 | Local | INT | %QW14 | | | Modbus Register 14 |
| 14 | sc_v1 | Local | BOOL | %QX2.0 | | | Modbus Register 16 (PLC input) scada c |
| 15 | sc_v2 | Local | BOOL | %QX2.1 | | | Modbus Register 17 |
| 16 | sc_v3 | Local | BOOL | %QX2.2 | | | Modbus Register 18 |
| 17 | sc_v_tank1_in | Local | BOOL | %QX2.3 | | | Modbus Register 19 |
| 18 | sc_v_tank1_water | Local | BOOL | %QX2.4 | | | Modbus Register 20 |
| 19 | sc_v_tank1_qout | Local | BOOL | %QX2.5 | | | Modbus Register 21 |
| 20 | sc_v_tank1_waste_ | Local | BOOL | %QX2.6 | | | Modbus Register 22 |
| 21 | sc_v_tank2_in | Local | BOOL | %QX2.7 | | | Modbus Register 23 |
| 22 | sc_v_tank2_water | Local | BOOL | %QX3.0 | | | Modbus Register 24 |
| 23 | sc_v_tank2_qout | Local | BOOL | %QX3.1 | | | Modbus Register 25 |
| 24 | sc_v_tank2_waste_ | Local | BOOL | %QX3.2 | | | Modbus Register 26 |
| 25 | is_tank1_chosen | Local | BOOL | %QX3.3 | | | Modbus Register 27 |
| 26 | is_tank2_chosen | Local | BOOL | %QX3.4 | | | Modbus Register 28 |
| 27 | start_sc | Local | BOOL | %QX3.5 | | | Modbus Register 29 |
| 28 | setpoint_A | Local | INT | %QW36 | | | Modbus Register 36 |
| 29 | setpoint_B | Local | INT | %QW37 | | | Modbus Register 37 |
| 30 | setpoint_C | Local | INT | %QW38 | | | Modbus Register 38 |
| 31 | step_ready | Local | INT | %QW39 | | | Modbus Register 39 |
| 32 | location | Local | INT | %QW40 | | | Modbus Register 40 |
| 33 | capacity_tank1 | Local | INT | %QW41 | | | Modbus Register 41 |
| 34 | capacity_tank2 | Local | INT | %QW42 | | | Modbus Register 42 |

Figure 32: Variables du PLC

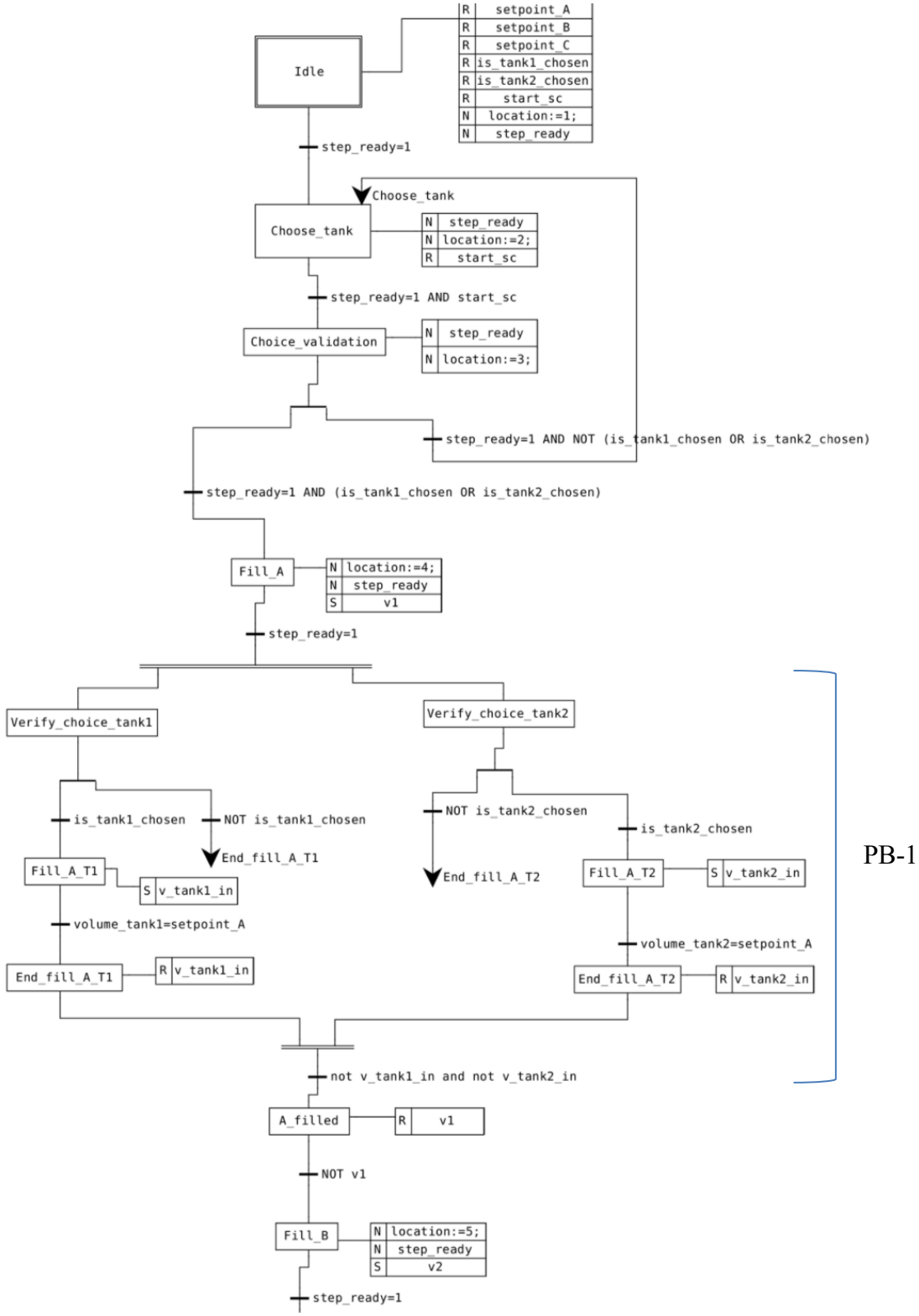



Figure 33: Programme SFC du PLC 1/3

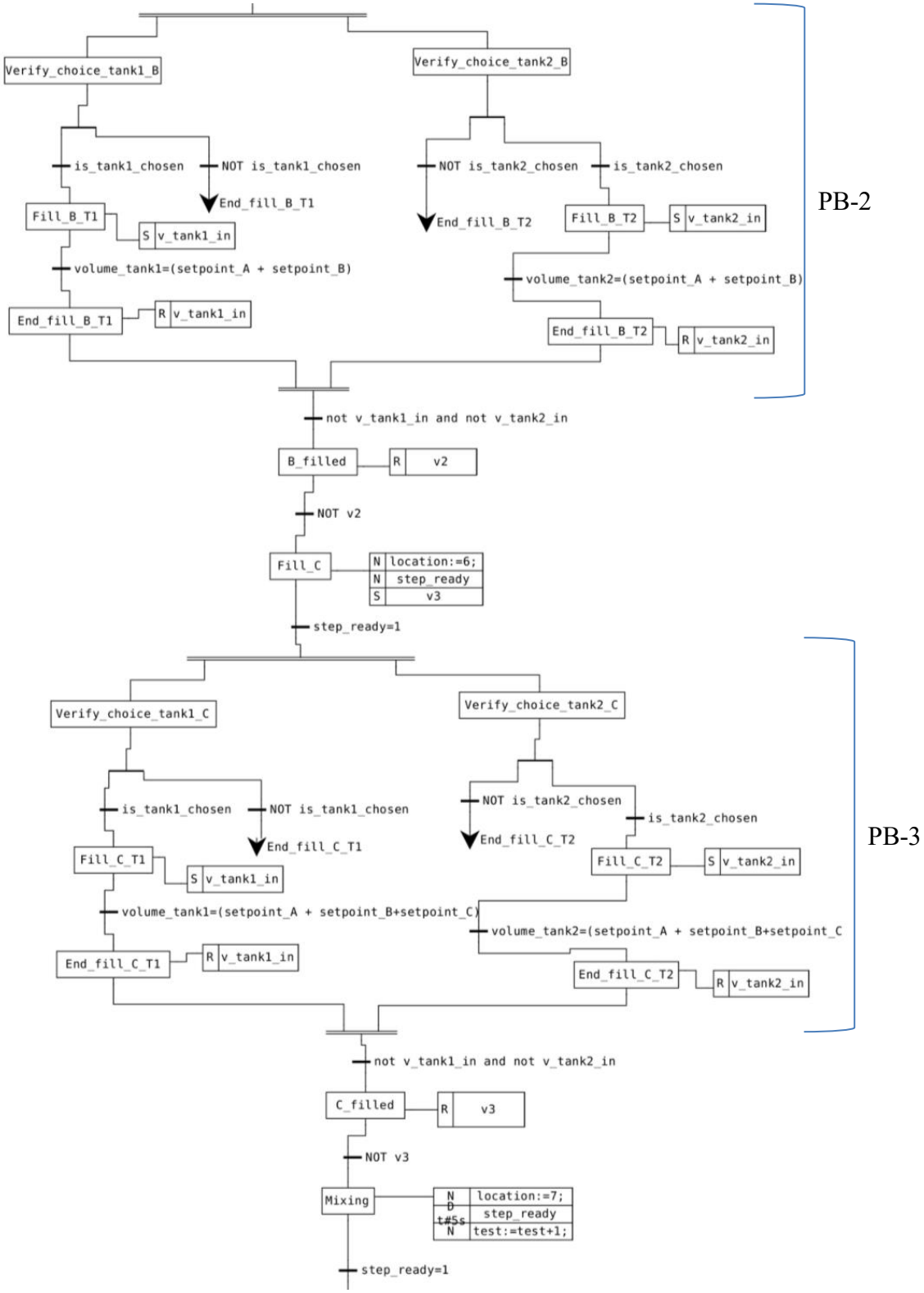



Figure 34: Programme SFC du PLC 2/3

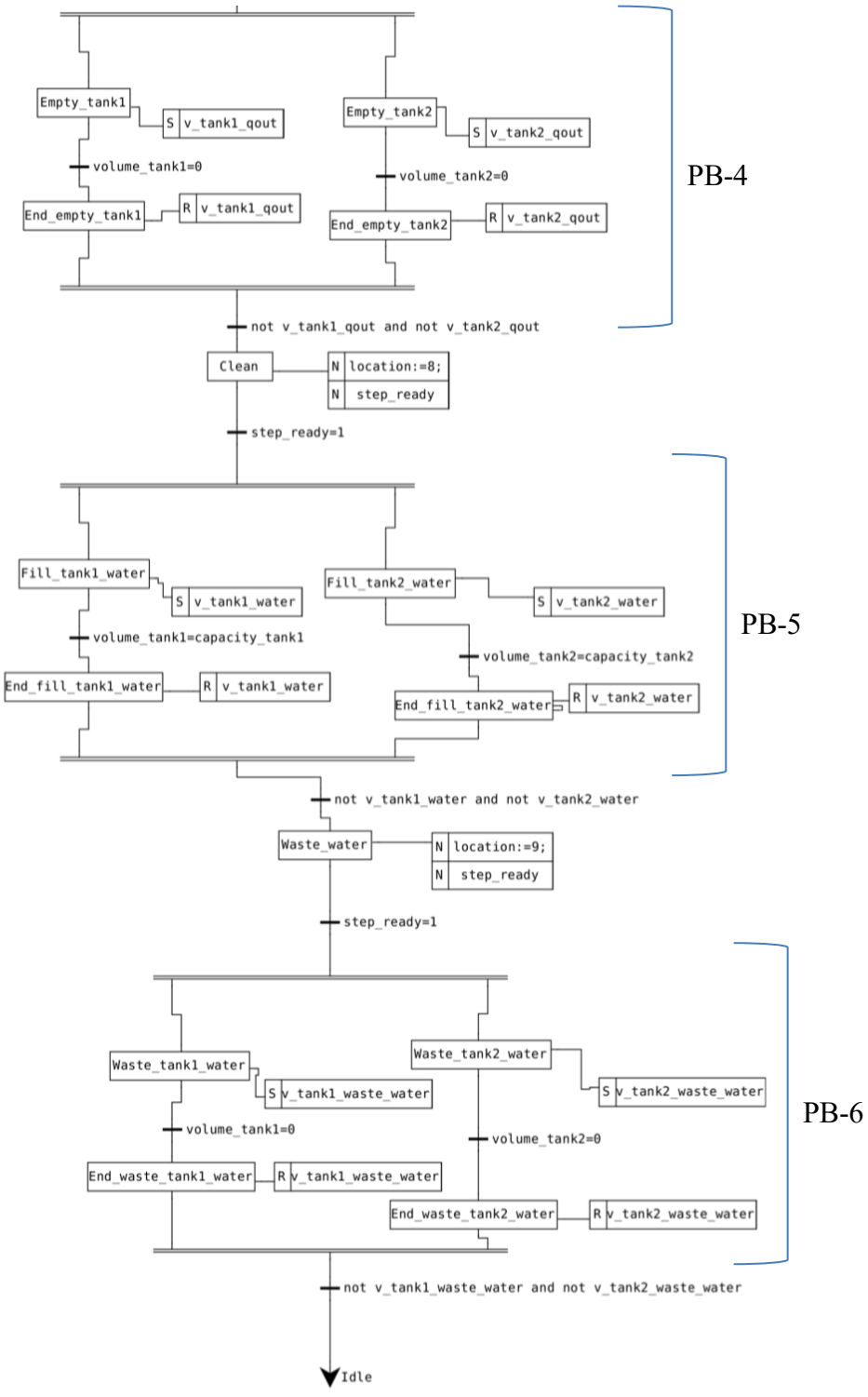



Figure 35: Programme SFC du PLC 3/3

## 5.3 Modélisation de l'ICS

La modélisation de l'ICS permet de fournir à l'IDS les connaissances nécessaires et de construire la première phase du processus de détection des attaques. Cette section détaille le modèle créé pour l'ICS et contient :

- Le modèle d'automate hybride (HA) modélisant le contrôleur (PLC).
- Le modèle HA du procédé.
- Les variables du système.
- L'architecture de l'ICS.

Dans ce qui suit, nous détaillons chacun de ces modèles ci-dessus.

### 5.3.1 Modèle du PLC

Le modèle du PLC est construit depuis le programme SFC du PLC et en suivant les étapes expliquées dans 4.2.4.2. Le modèle résultant est illustré sur les figures : Figure 36, Figure 37, Figure 38 et Figure 39.

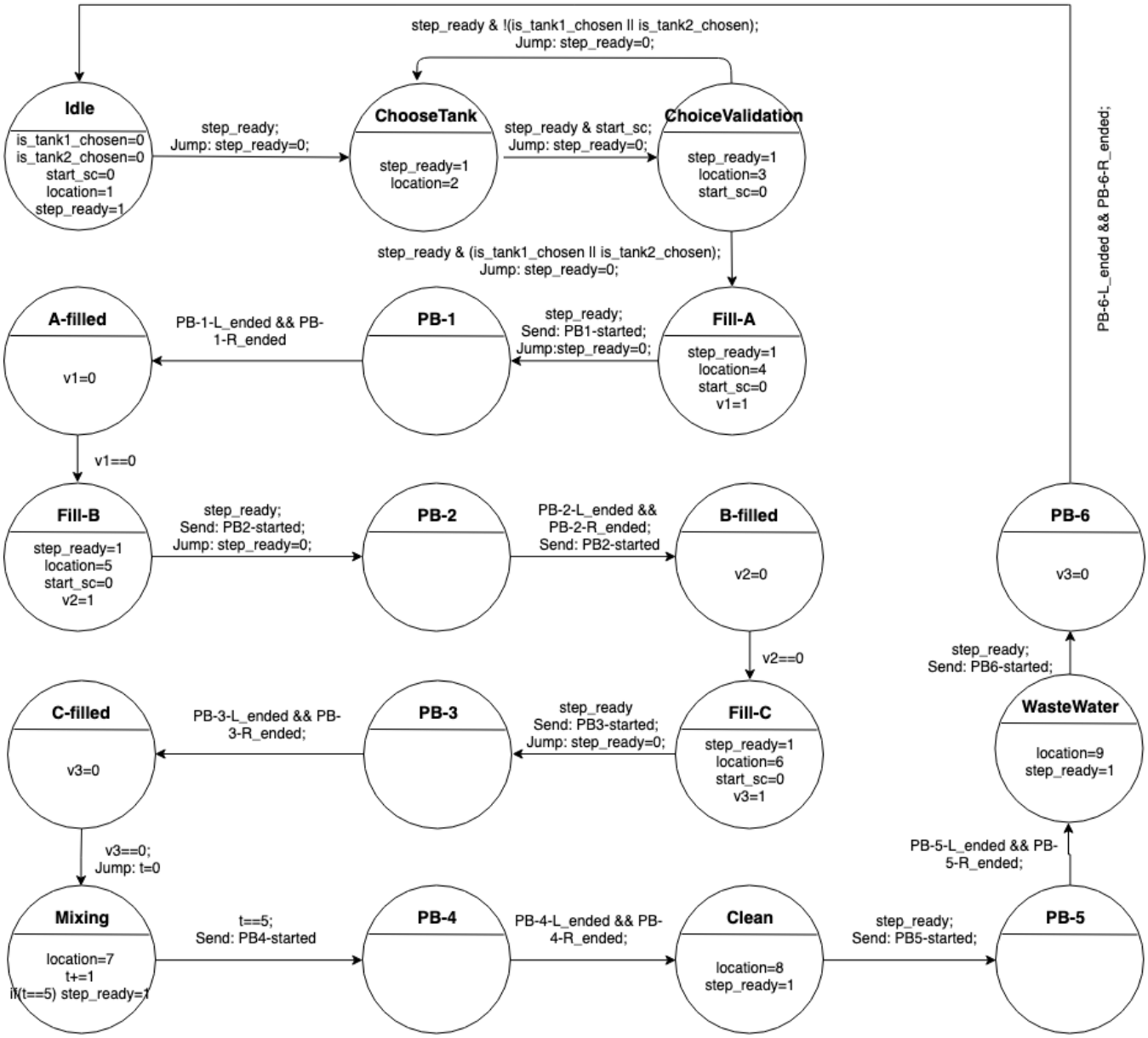



Figure 36: Le modèle HA du PLC 1/4

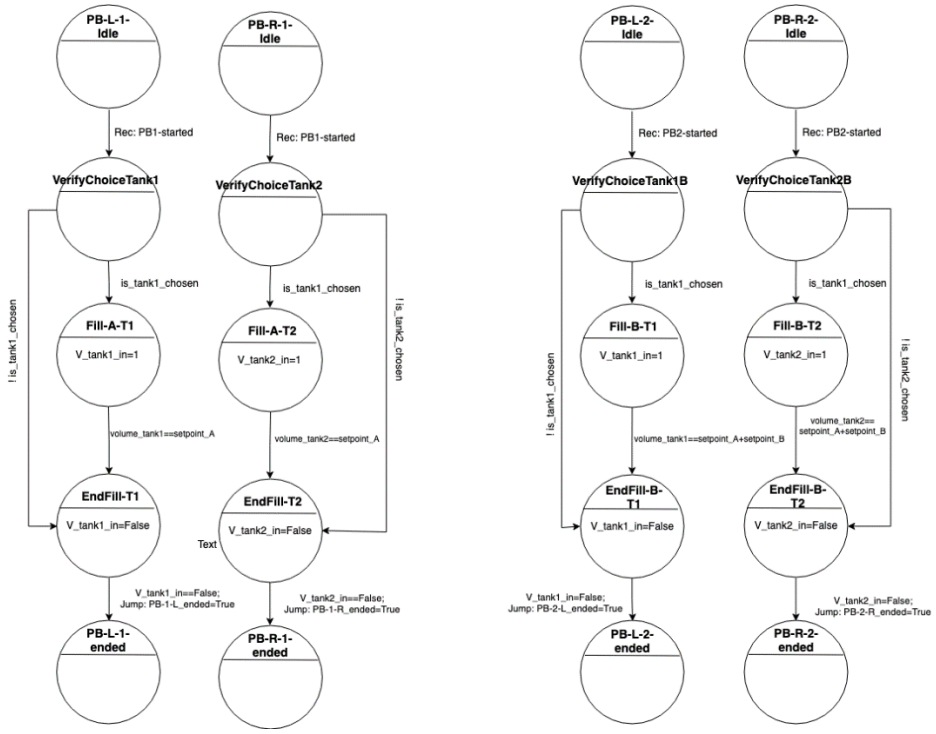



Figure 37: Le modèle HA du PLC 2/4


Figure 38: Le modèle HA du PLC 3/4

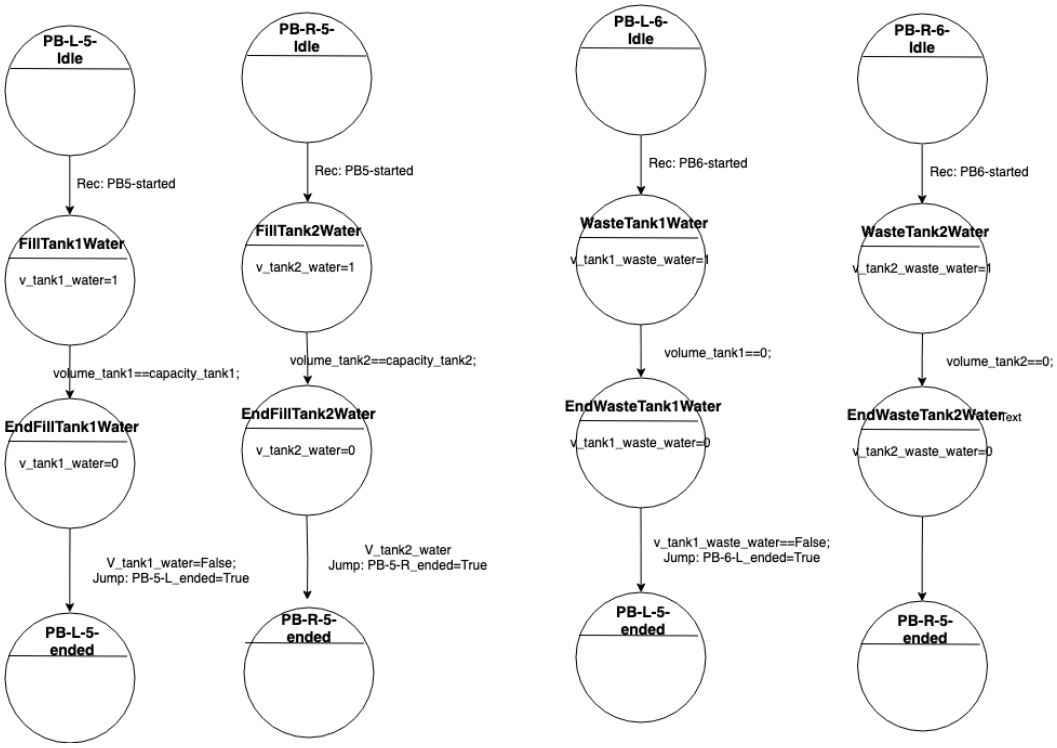



Figure 39: Le modèle HA du PLC 4/4

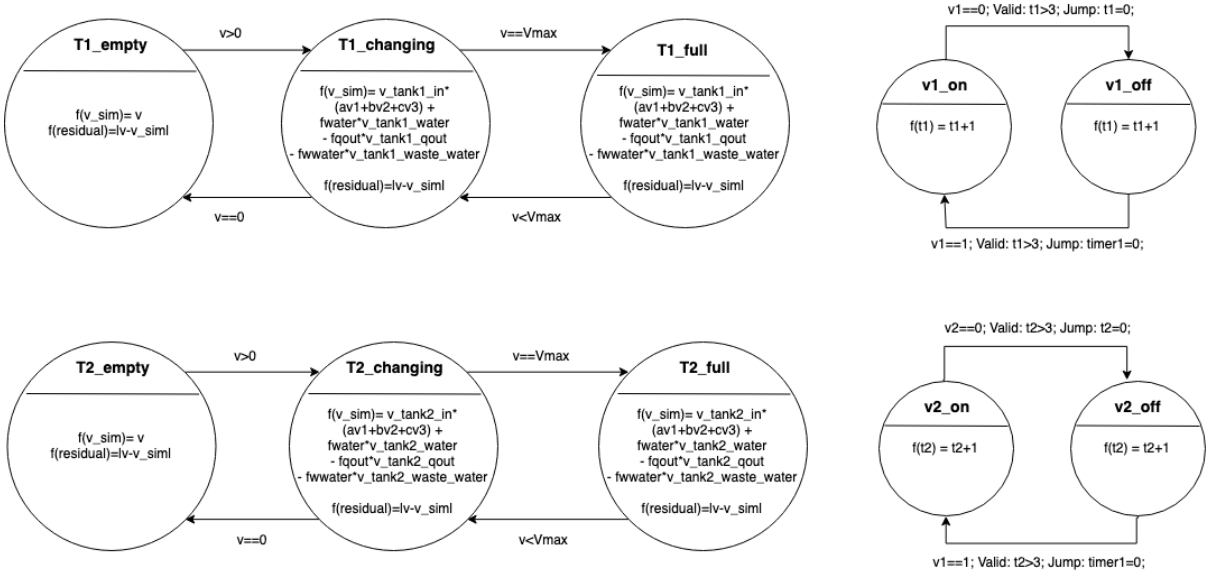



Figure 40: Le modèle HA du process physique

### 5.3.2 Modèle du process

Le modèle du process est créé manuellement à partir des spécifications du processus physique. Le modèle résultant est présenté dans la Figure 40.

### 5.3.3 Les contraintes de l'ICS

Comme c'est déjà spécifié dans le chapitre 3, les contraintes de l'ICS construisent les conditions qui doivent être vérifiées tout au long du processus de production. Ces conditions doivent être vérifiées par l'IDS afin de vérifier le fonctionnement légitime du processus de production. Ce manuscrit spécifie ces conditions en tant qu'invariants comme illustré dans le Tableau 10:

À noter que pour la lisibilité du tableau, ce manuscrit utilise le terme « Verify_open_valves_vs_simulation » pour la condition de vérification des vannes réelles ouvertes par rapport à la simulation. Ce terme résume les conditions suivantes :

"v1_residual_sum<5" et "v2_residual_sum<5" et "v3_residual_sum<5" et "v_tank1_in_residual_sum<5" et "v_tank1_water_residual_sum<5", "v_tank1_qout_residual_sum<5", "v_tank1_waste_water_residual_sum<5", "v_tank2_in_residual_sum<5", "v_tank2_water_residual_sum<5", "v_tank2_qout_residual_sum<5", "v_tank2_waste_water_residual_sum<5"

Où un résiduel est la différence entre la valeur réelle de la vanne et sa valeur de simulation. Un résiduel entre la valeur réelle d'une vanne v et sa valeur de simulation v_sim est égal à 1 si une de ces vannes est ouverte et l'autre et fermé. Et la somme de résiduels pour une vanne « v_residual_sum » représente la somme des résiduels de la vanne.

Tableau 10: Les invariants

| # | L | Invariant |
|---|---|---|
| **I1** | Idle | Verify_open_valves_vs_simulation |
| **I2** | ChooseTank | Verify_open_valves_vs_simulation |
| **I3** | ChoiceValidation | Verify_open_valves_vs_simulation |
| **I4** | Fill_A | Verify_open_valves_vs_simulation |
| **I5** | PB_1 | Verify_open_valves_vs_simulation |

| | | |
|---|---|---|
| **I6** | A_filled | Verify_open_valves_vs_simulation, v1_plc==0 |
| **I7** | Fill_B | Verify_open_valves_vs_simulation |
| **I8** | PB_2 | Verify_open_valves_vs_simulation |
| **I9** | B_filled | Verify_open_valves_vs_simulation, v2_plc==0 |
| **I10** | Fill_C | Verify_open_valves_vs_simulation |
| **I11** | PB_3 | Verify_open_valves |
| **I12** | C_filled | Verify_open_valves_vs_simulation, v3_plc==0 |
| **I13** | Mixing | Verify_open_valves_vs_simulation |
| **I14** | PB_4 | Verify_open_valves_vs_simulation |
| **I15** | Clean | Verify_open_valves_vs_simulation |
| **I16** | PB_5 | Verify_open_valves_vs_simulation |
| **I17** | WasteWater | Verify_open_valves_vs_simulation |
| **I18** | PB_6 | Verify_open_valves_vs_simulation |
| **I19** | Fill_A_T1 | volume_tank1 < (setpoint_A + threshold) |
| **I20** | Fill_A_T2 | volume_tank2 < (setpoint_A + threshold) |
| **I21** | Fill_B_T1 | volume_tank1 < (setpoint_A + setpoint_B + threshold) |
| **I22** | Fill_B_T2 | volume_tank2 < (setpoint_A + setpoint_B + threshold) |
| **I23** | Fill_C_T1 | volume_tank1 < (setpoint_A + setpoint_B + setpoint_C + threshold) |
| **I24** | Fill_C_T2 | volume_tank2 < (setpoint_A + setpoint_B + setpoint_C + threshold) |
| **I25** | EndEmptyTank1 | volume_tank1 == 0 |
| **I26** | EndEmptyTank2 | volume_tank2 == 0 |
| **I27** | EndFillTank1Water | volume_tank1 == capacity_tank1 |
| **I28** | EndFillTank2Water | volume_tank2 == capacity_tank2 |
| **I29** | EndWasteTank1Water | volume_tank1 == 0 |
| **I30** | EndWasteTank2Water | volume_tank2 == 0 |
| **I31** | T1_empty | volume_tank1_sim – volume_tank1 < 5, volume_tank1_sim – volume_tank1 > -5 |
| **I32** | T1_non_empty | volume_tank1_sim – volume_tank1 < 5, volume_tank1_sim – volume_tank1 > -5 |
| **I33** | T1_full | volume_tank1_sim – volume_tank1 < 5, volume_tank1_sim – volume_tank1 > -5 |
| **I34** | T2_empty | volume_tank2_sim – volume_tank2 < 5, volume_tank2_sim – volume_tank2 > -5 |
| **I35** | T2_non_empty | volume_tank2_sim – volume_tank2 < 5, volume_tank2_sim – volume_tank2 > -5 |

| | | |
|---|---|---|
| **I36** | T2_full | volume_tank2_sim – volume_tank2 < 5, volume_tank2_sim – volume_tank2 > -5 |

### 5.3.4 Les variables

Deux types de variables sont créés pour désigner les valeurs de l'ICS : les variables du système (les variables du process et du PLC) et les variables définies par l'utilisateur. La liste complète des variables est illustrée sur le Tableau 11 :

Tableau 11: Les variables du process

| Nom de variable | Type | Adresse | Envoyé vers l'IDS | Indice dans la liste des variables envoyés vers l'IDS |
|---|---|---|---|---|
| **v1_plc** | bool | 0 | False | |
| **v2_plc** | bool | 1 | False | |
| **v3_plc** | bool | 2 | False | |
| **v_tank1_in_plc** | bool | 3 | False | |
| **v_tank1_water_plc** | bool | 4 | False | |
| **v_tank1_qout_plc** | bool | 5 | False | |
| **v_tank1_waste_water_plc** | bool | 6 | False | |
| **v_tank2_in_plc** | bool | 7 | False | |
| **v_tank2_water_plc** | bool | 8 | False | |
| **v_tank2_qout_plc** | bool | 9 | False | |
| **v_tank2_waste_water_plc** | bool | 10 | False | |
| **volume_tank1** | bool | 13 | False | |
| **volume_tank2** | bool | 14 | False | |
| **is_tank1_chosen** | bool | 27 | True | 54 |
| **is_tank2_chosen** | bool | 28 | True | 55 |
| **start_sc** | bool | 29 | True | 53 |
| **start_sc_plc** | bool | 29 | True | 53 |
| **setpoint_A** | int | 36 | True | 58 |

| | | | | |
|---|---|---|---|---|
| **setpoint_B** | int | 37 | True | 59 |
| **setpoint_C** | int | 38 | True | 60 |
| **capacity_tank1** | int | 41 | True | 56 |
| **capacity_tank2** | int | 42 | True | 57 |
| **v1** | bool | | True | 1 |
| **v2** | bool | | True | 2 |
| **v3** | bool | | True | 3 |
| **v_tank1_in** | bool | | True | 4 |
| **v_tank1_water** | bool | | True | 5 |
| **v_tank1_qout** | bool | | True | 6 |
| **v_tank1_waste_water** | bool | | True | 7 |
| **v_tank2_in** | bool | | True | 8 |
| **v_tank2_water** | bool | | True | 9 |
| **v_tank2_qout** | bool | | True | 10 |
| **v_tank2_waste_water** | bool | | True | 11 |
| **v1_residual_sum** | int | | True | 12 |
| **v2_residual_sum** | int | | True | 13 |
| **v3_residual_sum** | int | | True | 14 |
| **v_tank1_in_residual_sum** | int | | True | 15 |
| **v_tank1_water_residual_sum** | int | | True | 16 |
| **v_tank1_qout_residual_sum** | int | | True | 17 |
| **v_tank1_waste_water_residual_sum** | int | | True | 18 |
| **v_tank2_in_residual_sum** | int | | True | 19 |
| **v_tank2_water_residual_sum** | bool | | True | 20 |
| **v_tank2_qout_residual_sum** | int | | True | 21 |
| **v_tank2_waste_water_residual_sum** | int | | True | 0 |
| **volume_tank1_sim** | int | | True | 22 |
| **volume_tank2_sim** | int | | True | 23 |
| **volume_tank1_residual** | int | | True | 24 |
| **volume_tank2_residual** | int | | True | 25 |
| **step_ready** | bool | | True | 26 |
| **location** | bool | | True | 27 |
| **t** | int | | True | 28 |
| **t1** | int | | True | 29 |
| **t2** | int | | True | 30 |
| **t3** | int | | True | 31 |
| **t4** | int | | True | 32 |

| | | | | |
|---|---|---|---|---|
| **t5** | int | | True | 33 |
| **t6** | int | | True | 34 |
| **t7** | int | | True | 35 |
| **t8** | int | | True | 36 |
| **t9** | int | | True | 37 |
| **t10** | int | | True | 38 |
| **t11** | int | | True | 39 |
| **PB_1_L_ended** | bool | | True | 40 |
| **PB_1_R_ended** | bool | | True | 41 |
| **PB_2_L_ended** | bool | | True | 42 |
| **PB_2_R_ended** | bool | | True | 43 |
| **PB_3_L_ended** | bool | | True | 44 |
| **PB_3_R_ended** | bool | | True | 45 |
| **PB_4_L_ended** | bool | | True | 46 |
| **PB_4_R_ended** | bool | | True | 47 |
| **PB_5_L_ended** | bool | | True | 48 |
| **PB_5_R_ended** | bool | | True | 49 |
| **PB_6_L_ended** | bool | | True | 50 |
| **PB_6_R_ended** | bool | | True | 51 |

### 5.3.5 Le modèle de l'architecture

Le modèle de l'architecture fournit à l'IDS les informations nécessaires sur les nœuds du réseau qui doivent communiquer entre eux. D'une autre part, il permet à l'IDS d'identifier certains nœuds tels que le PSE et le module de génération de trafique (afin de récupérer les informations complémentaires sur l'état du système). D'autre part, ce modèle permet de vérifier que seuls les nœuds autorisés sont autorisés à communiquer entre eux. Dans ce cas, l'IDS vérifie les adresses IP des sources et des destinations dans les paquets observés.

La Figure 41 illustre le modèle d'architecture du système de cuves. Cependant, il faut noter que le module SCADA, même s'il est membre de l'architecture de l'ICS, n'apparaît pas dans le modèle puisqu'il est implémenté sur un réseau (Nat-0) différent de celui de l'IDS (Nat-1). Autrement dit, les paquets communiqués depuis / vers le module SCADA ne doivent pas être détectées par l'IDS dans un comportement normal du système.

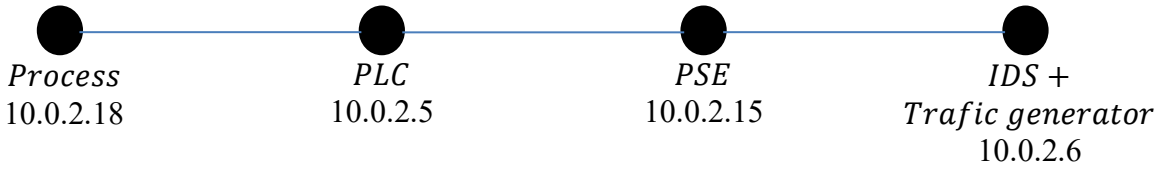



Figure 41: Modèle de l'architecture du process physique

## 5.4 Liste des attaques appliqués et les résultats des détections

Pour évaluer l'approche proposée, ce manuscrit applique plusieurs attaques comme c'est illustré sur le Tableau 12:

Tableau 12: Les attaques appliqués

| # | Attaque | Description |
|---|---|---|
| 1 | Compromission des mesures du capteur. | Compromission d'un capteur pour renvoyer des mesures erronées au PLC. |
| 2 | Compromission des actionneurs | Compromission d'une vanne en les envoyant directement des commandes externes. |
| 3 | Corruption des commandes. | Empêche les commandes d'atteindre le process pour conduire le système à des états critiques. |
| 4 | Corruption des mesures. | Trompe l'utilisateur pour conduire le système à des états inacceptables. |
| 5 | DoS sur le PLC | Empêche l'envoi des commandes et la réception des mises à jour à partir du PLC. |
| 6 | DoS par flooding sur le PLC. | Inonde le PLC avec de fausses commandes pour écraser les commandes légitimes. |

1. Attaque N1 (corruption des commandes sur v1 et v2) : cette attaque est de type MitM. Elle empêche la fermeture de v2 en remplaçant les commandes d'ouverture de v2 par des commandes d'ouverture de v1.
   Détection : violation de l'invariant I6.
2. Attaque N2 (corruption de mesure) : cette attaque est de type MitM. Elle change la mesure des volumes envoyé depuis le process vers le PLC en ajoutant 50 à la valeur

initiale.

Détection : l'application de cette attaque viole les invariants I32 et I35.

3. Attaque N3 (DoS) : dans cette attaque, l'attaquant utilise l'outil Ettercap pour appliquer un DoS sur le PLC. Ceci implique une perte de communication entre le PLC et le process.

   Détection : cette attaque est détectée par l'IDS en violant le modèle de timing entre les requêtes du process et le PLC. (il n'y a plus de paquet entre le process et le PLC pendant une durée assez longue). En plus, l'IDS détecte après un certain temps une violation de l'invariant I32. Ainsi, d'autre invariants sont violé (I23 et I24) lors de la réception des valeurs du volume réel après la fin de l'attaque.

4. Attaque N4 (inondations à base de DoS) : cette attaque est basée sur l'inondation du PLC par des valeurs erronées pour v1, v2, v3 et v1_in. En utilisant cette attaque, l'attaquant n'empêche pas l'envoie des bonnes commandes, mais envoie d'une façon continue des valeurs erronées au PLC pour écraser les valeurs initiales. En général, il existe deux méthodes pour appliquer ce type d'attaque :

   a. L'adversaire se connecte sur le réseau et envoie ses commandes d'inondation vers le PLC. Ceci serait détecté simplement par n'importe quel IDS en détectant une source non autorisée pour les commandes.
   b. L'adversaire se connecte directement au PLC et envoie ses commandes d'inondation sans laisser des traces sur le réseau. Cette méthode est plus sérieuse, et est appliquée dans ce manuscrit pour vérifier la possibilité de sa détection par l'approche de détection proposée.

   Cette attaque est appliquée dans deux scénarios différents où chacun implique des résultats différents :

   a. Dans le premier scénario, l'adversaire applique l'attaque avant le début du processus (le bouton « start » n'est pas encore actionné). D'après le programme SFC du PLC, ce bouton indique le début du processus de production. Du point de vue de PLC, tant que ce bouton n'est pas actionné, le processus ne doit pas commencer (l'étape équivalente est ChooseTank). Cependant, du point de vue

du process, il reçoit une commande d'ouverture des vannes d'entrée, donc le cuve commence à se remplir.
Détection : le modèle de simulation du contrôleur est construit à partir du programme SFC du PLC. Le PSE lit à partir de la mémoire du PLC que les vanne d'entrées sont ouvertes, mais le processus ne doit pas commencer (le bouton start n'est pas encore actionné). Par conséquent, le PSE n'ouvre pas les vanne de simulation (v1_sim, v2_sim, v3_sim, v1_in_sim) et reste dans l'attente du début de process et le volume de simulation reste zéro.
La détection dans ce cas se fait par la violation de l'invariant I32 et des invariants : I31, I32 et I2 qui consistent à calculer la somme des valeurs résiduelles entre l'ouverture des vannes réelles et les vannes de simulation.

b. Dans le deuxième scénario, l'adversaire attend qu'un l'utilisateur normal clique le bouton start, puis applique la même attaque.
Détection : après que le PSE reçoit depuis sa lecture du mémoire du PLC que le bouton start est cliqué, il ouvre la vanne v1 et v1_in comme (sans ouvrir les autres vannes) c'est spécifié dans son modèle (HA). Par conséquent, le volume dans le cuve1 de simulation augmente mais lentement par rapport au volume réel (en appliquant les activités du modèle HA).
Donc, l'IDS détecte une violation de l'invariant I32 et des invariants : I31, I32 et I2 en calculant la somme des valeur résiduelles entre l'ouverture des vannes réelles et les vannes de simulation.

5. Attaque N5 (compromission des vannes) : cette attaque compromet la vanne v1 et v1_in pour qu'elle soit ouverte sans recevoir une commande d'ouverture. Il peut simuler une attaque physique sur une vanne, ce qui ne peut pas être détecté en analysant le réseau de communication.
Détection : Selon le modèle de l'ICS, le PLC ne doit pas ouvrir la vanne d'entrée. Par conséquent, la vannes de simulation du module PSE restent fermées et le calcul des activités par le PSE résulte par des cuves vides.
Donc, l'IDS détecte une violation de l'invariant I32.

6. Attaque N6 (compromettre les capteurs): cette attaque compromet le capteur de volume dans la cuve1 et l'oblige à renvoyer des valeurs erronées des mesures du volume (en ajoutant 50 aux valeurs réelles lorsque le volume de la cuve1 est entre 150 et 450). Détection : Lorsque le PSE calculait le changement de volume, il considérait que le volume doit augmenter selon les lois de dynamique du système (selon les flux des vannes ouvertes).
   L'IDS détecte une violation de l'invariant I32. Ainsi, l'IDS détecte une violation des invariants I21 et I23.

## 5.5 Bilan du chapitre

Ce chapitre présente un exemple réaliste de l'application de notre approche sur un système industriel. En premier lieu, une description du système est présentée. Ensuite, les modèles créés en suivant notre approche constituant l'entrée du logiciel de transformations des règles sont détaillés. Enfin, ce chapitre est conclu par une application des attaques évaluatif de l'approche, en plus des résultats obtenus.

# Conclusion Générale

Les travaux présentés dans ce document traitent la surveillance d'attaques dans les systèmes de contrôle industriels. Ils ont été développés au sein du de l'équipe Gestion et Conduite des Systèmes de Production (GCSP) du laboratoire des Science pour la Conception, l'Optimisation et la Production de Grenoble (G-SCOP) et le laboratoire MECRL de l'université Libanaise - Faculté des Sciences I.

Cette thèse a apporté sa contribution au travers de la proposition d'un mécanisme de détection d'intrusion à base de modèles dans les systèmes industriels. Cette détection est basée sur des règles de surveillance et de détection générée par un module que nous proposons qui convertit automatiquement les modèles de l'ICS en des règles d'IDS.

L'état de l'art montre que les approches basées sur les connaissances du système fournissent des résultats intéressants en matière de détection d'intrusion dans le domaine des contrôleurs industriels. Cependant, nous avons compris que la mise en œuvre d'une telle approche est difficile en termes de conception et de maintenabilité d'IDS.

Au cours de cette thèse, nous nous sommes concentrés sur le développement d'une méthodologie pour faciliter la conception et la maintenabilité des IDS basés sur des modèles. Notre travail s'est concentré sur trois axes principaux :

- En premier lieu, nous avons utilisé une approche modulaire pour définir les modèles à utiliser pour représenter les connaissances du système en utilisant un formalisme standard (les automates hybrides).
- Ensuite, nous avons proposé un générateur de règles qui convertit le modèle d'entrée en des règles d'IDS. Les règles générées par ce générateur permettent la détection d'attaques au niveau IT et OT. En plus, ces règles permettent l'estimation de l'état du processus et le calcul des variables internes de l'ICS requises pour l'analyse sémantique de l'état du système. Enfin, les règles générées sont implémentées en utilisant l'open-

source IDS « Zeek ». L'utilisation d'un IDS opensource montre l'interopérabilité de notre travail avec la communauté open-source. Dans le même temps, elle nous permet de d'avoir un détecteur d'intrusion qui a déjà prouvé sa capacité sans avoir à réinventer la roue.

Afin de valider l'approche, nous avons développé un prototype informatique fonctionnel du mécanisme de détection proposé. Ce prototype nous a permis de mener une évaluation de l'approche sur la base de plusieurs cas d'étude, auxquels nous avons fait subir, en simulation, plusieurs scénarios d'attaques. Les résultats obtenus sont significatifs, confortant chacun des apports majeurs de l'approche proposée :

- La détection s'appuie sur des modèles comportementaux permettant l'analyse sémantique de l'information au sens métier.
- Le mécanisme de détection développé a permis une détection efficace des intrusions appliqués sur les systèmes sans avoir une connaissance préalable des éventuelles intrusions.
- L'utilisation des règles générées (et qui ne sont pas créés manuellement) a permis une vérification complexe des propriétés du système. Ces propriétés comprennent le respect de la dynamique du système représentée par les lois physique qui décrivent le changement des variables du système. Une vérification de telles propriété peut être très difficile à appliquer en utilisant des règles créer à la main.
- Avoir un IDS avec règles « générées » à partir d'un langage à haut niveau est très important dans le sens de maintenabilité des systèmes industriels qui peut être considérés comme un cauchemar dans le cas d'un système complexe.

Enfin, l'utilisation d'une solution open-source pour l'implémentation de l'IDS pour les systèmes industriels est un facteur de confiance pour détecter des actions malveillantes sur des systèmes critiques qui peuvent avoir des effets graves sur la vie des employés et des citoyens.

Aux termes de ces travaux, plusieurs axes de recherche se dégagent pour envisager, du point de vue des perspectives, de prolonger l'étude menée :

Du point de vue de l'implémentation, plusieurs idées peuvent être développées :

- L'implémentation que nous avons appliquée ne considère, en termes des langages de programmation des PLC, que la translation des programme PLC écrit en langage SFC en des règles d'IDS. Le développement des mécanismes de translation pour les autres langages, tels que LD, IL et ST est possible de point de vue théorique. Il serait bien d'avoir des algorithmes de translation prêts à être employé.
- Notre implémentation a seulement pris en considération le protocole Modbus, l'implémentation pour les autres protocoles de communication peut être encore utile.
- En plus, d'autres solutions open-source peuvent être considérés pour implémenter les règles de détection générées.

Du point de vue conceptuel, le problème d'incertitudes peut être pris en considération lors de l'application des approches de détection à base modèles, bien que nous ayons utilisé un système de filtrage pour réduire l'écart entre les valeurs reçues du système et les valeurs calculées. Cependant, le problème des incertitudes dans les processus de l'ICS n'est pas au centre de notre travail. L'approche proposée étant basée sur un modèle à boîte blanche, les incertitudes possibles et les erreurs dans les composants du processus peuvent être prises en compte dans la phase de modélisation. Par exemple, des plages de valeurs acceptables pourraient être utilisées au lieu d'utiliser uniquement des valeurs strictes. L'application d'une telle solution pourrait être utile pour tenir compte des incertitudes. Cependant, cela peut entraîner des retards dans le processus de détection. Plusieurs tests doivent être appliqués pour vérifier l'efficacité de ce type de solution.

# Références


3: Programming languages, 2003. . Int. Electrotech. Comm. IEC Std IEC 61131–3.

Albin, E., Rowe, N.C., 2012. A realistic experimental comparison of the Suricata and Snort intrusion-detection systems, in: Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference On. IEEE, pp. 122–127.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S., 1995. The algorithmic analysis of hybrid systems. Theor. Comput. Sci. 138, 3–34.

Automation, R., 2011. Converged Plantwide Ethernet (CPwE) Design and Implementation Guide.". Citeseer.

Axelsson, S., 2000. Intrusion detection systems: A survey and taxonomy. Technical report.

Banerjee, A., Kandula, S., Mukherjee, T., Gupta, S.K., 2012. BAND-AiDe: A tool for cyber-physical oriented analysis and design of body area networks and devices. ACM Trans. Embed. Comput. Syst. TECS 11, 49.

Barbosa, R.R.R., Sadre, R., Pras, A., 2013. Flow whitelisting in SCADA networks. Int. J. Crit. Infrastruct. Prot. 6, 150–158.

Basile, F., Chiacchio, P., Coppola, J., 2016. A Colored Timed Petri Net model for a cyber-physical view of automated warehouse systems, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, pp. 1–8.

Basnight, Z., Butts, J., Lopez Jr, J., Dube, T., 2013. Firmware modification attacks on programmable logic controllers. Int. J. Crit. Infrastruct. Prot. 6, 76–84.

Bauer, N., Engell, S., Huuck, R., Lohmann, S., Lukoschus, B., Remelhe, M., Stursberg, O., 2004. Verification of PLC programs given as sequential function charts, in: Integration of Software Specification Techniques for Applications in Engineering. Springer, pp. 517–540.

Carcano, A., Fovino, I.N., Masera, M., Trombetta, A., 2009. State-based network intrusion detection systems for SCADA protocols: a proof of concept, in: International Workshop on Critical Information Infrastructures Security. Springer, pp. 138–150.

Cárdenas, A.A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., Sastry, S., 2011. Attacks against process control systems: risk assessment, detection, and response, in: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. ACM, pp. 355–366.

Cardin, O., Ounnar, F., Thomas, A., Trentesaux, D., 2016. Future industrial systems: best practices of the intelligent manufacturing and services systems (IMS2) French Research Group. IEEE Trans. Ind. Inform. 13, 704–713.

Caselli, M., Zambon, E., Kargl, F., 2015. Sequence-aware intrusion detection in industrial control systems, in: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security. ACM, pp. 13–24.

Chatterjee, S., Thekdi, S., 2020. An iterative learning and inference approach to managing dynamic cyber vulnerabilities of complex systems. Reliab. Eng. Syst. Saf. 193, 106664. https://doi.org/10.1016/j.ress.2019.106664

Cheminod, M., Durante, L., Valenzano, A., 2012. Review of security issues in industrial networks. IEEE Trans. Ind. Inform. 9, 277–293.

Cheung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., Valdes, A., 2007. Using model-based intrusion detection for SCADA networks, in: Proceedings of the SCADA Security Scientific Symposium. Citeseer, pp. 1–12.

Control System Security DMZ | CISA [WWW Document], n.d. URL https://www.us-cert.gov/ics/Control_System_Security_DMZ-Definition.html (accessed 7.4.19).

Denning, D.E., 1987. An intrusion-detection model. IEEE Trans. Softw. Eng. 222–232.

Ding, Y.-X., Xiao, M., Ai-Wu Liu, 2009. Research and implementation on snort-based hybrid intrusion detection system, in: 2009 International Conference on Machine Learning and Cybernetics. Presented at the 2009 International Conference on Machine Learning and Cybernetics (ICMLC), IEEE, Baoding, China, pp. 1414–1418. https://doi.org/10.1109/ICMLC.2009.5212282

Evangelista, T., 2004. Les IDS: les systèmes de détection d'intrusions informatiques. Dunod.

Ferling, B., Chromik, J., Caselli, M., Remke, A., 2018. Intrusion Detection for sequence-based attacks with reduced traffic models, in: International Conference on Measurement, Modelling and Evaluation of Computing Systems. Springer, pp. 53–67.

Fiaidhi, J., Gelogo, Y.E., 2012. SCADA cyber attacks and security vulnerabilities. SERSC Res. Trend Comput. Sci. Relat. Areas ASTL 14, 202–208.

Flaus, J.-M., 2019. Cybersecurity of Industrial Systems (*Cybersé*curité des systèmes industriels). ISTE Ltd, London, and Wiley, New York.

Fovino, I.N., Carcano, A., Murel, T.D.L., Trombetta, A., Masera, M., 2010. Modbus/DNP3 state-based intrusion detection system, in: 2010 24th IEEE International Conference on Advanced Information Networking and Applications. IEEE, pp. 729–736.

Fovino, I.N., Coletta, A., Carcano, A., Masera, M., 2011. Critical state-based filtering system for securing SCADA network protocols. IEEE Trans. Ind. Electron. 59, 3943–3950.

Galloway, B., Hancke, G.P., 2012. Introduction to industrial control networks. IEEE Commun. Surv. Tutor. 15, 860–880.

Graja, I., Kallel, S., Guermouche, N., Cheikhrouhou, S., Hadj Kacem, A., 2018. A comprehensive survey on modeling of cyber-physical systems: Survey on modeling CPS. Concurr. Comput. Pract. Exp. e4850. https://doi.org/10.1002/cpe.4850

Hadeli, H., Schierholz, R., Braendle, M., Tuduce, C., 2009. Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration, in: 2009 IEEE Conference on Emerging Technologies & Factory Automation. IEEE, pp. 1–8.

Hadziosmanovic, D., Sommer, R., Zambon, E., Hartel, P.H., 2014. Through the eye of the PLC: semantic security monitoring for industrial processes, in: ACSAC.

Han, S., Xie, M., Chen, H.-H., Ling, Y., 2014. Intrusion detection in cyber-physical systems: Techniques and challenges. IEEE Syst. J. 8, 1052–1062.

Hassapis, G., Kotini, I., Doulgeri, Z., 1998. Validation of a SFC software specification by using hybrid automata. IFAC Proc. Vol. 31, 107–112.

Horkan, M., 2015. Challenges for IDS/IPS deployment in industrial control systems. Inst. Read. Room.

https://www.cisco.com/c/en/us/td/docs/solutions/Verticals/CPwE/CPwE_DIG/CPwE_chapter2.html [WWW Document], 2013. . Chapter Converg. Plantwide Ethernet Solut. URL https://www.cisco.com/c/en/us/td/docs/solutions/Verticals/CPwE/CPwE_DIG/CPwE_chapter2.html (accessed 6.12.19).

Hu, Y., Yang, A., Li, H., Sun, Y., Sun, L., 2018. A survey of intrusion detection on industrial control systems. Int. J. Distrib. Sens. Netw. 14, 155014771879461. https://doi.org/10.1177/1550147718794615

Hutchins, E.M., Cloppert, M.J., Amin, R.M., 2011. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Lead. Issues Inf. Warf. Secur. Res. 1, 80.

ICS-CERT Landing | CISA [WWW Document], n.d. URL https://www.us-cert.gov/ics (accessed 7.8.19).

Jeon, B.-S., Na, J.-C., 2016. A study of cyber security policy in industrial control system using data diodes, in: 2016 18th International Conference on Advanced Communication Technology (ICACT). IEEE, pp. 314–317.

Keliris, A., Konstantinou, C., Tsoutsos, N.G., Baiad, R., Maniatakos, M., 2016. Enabling multi-layer cyber-security assessment of Industrial Control Systems through Hardware-In-The-Loop testbeds, in: 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC). Presented at the 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, Macao, Macao, pp. 511–518. https://doi.org/10.1109/ASPDAC.2016.7428063

Khoury, J., Nassar, M., 2020. A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security, in: NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 1–9.

Kumar, S., Spafford, E.H., 1994. An application of pattern matching in intrusion detection.

Larsen, K.G., Pettersson, P., Yi, W., 1997. UPPAAL in a nutshell. Int. J. Softw. Tools Technol. Transf. STTT 1, 134–152.

Li, G., Song, D., Liao, L., Sun, F., Du, J., 2014. Learning automata-based adaptive web services composition, in: 2014 IEEE 5th International Conference on Software Engineering and Service Science. IEEE, pp. 792–795.

Lin, H., Slagell, A., Di Martino, C., Kalbarczyk, Z., Iyer, R.K., 2013. Adapting Bro into SCADA: building a specification-based intrusion detection system for the DNP3 protocol, in: Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop on - CSIIRW '13. Presented at the the Eighth Annual Cyber Security and Information Intelligence Research Workshop, ACM Press, Oak Ridge, Tennessee, p. 1. https://doi.org/10.1145/2459976.2459982

Linda, O., Manic, M., Alves-Foss, J., Vollmer, T., 2011a. Towards resilient critical infrastructures: Application of Type-2 Fuzzy Logic in embedded network security cyber sensor, in: 2011 4th International Symposium on Resilient Control Systems. Presented at the 2011 4th International Symposium on Resilient Control Systems (ISRCS), IEEE, Boise, ID, USA, pp. 26–32. https://doi.org/10.1109/ISRCS.2011.6016083

Linda, O., Manic, M., Vollmer, T., 2012. Improving cyber-security of smart grid systems via anomaly detection and linguistic domain knowledge, in: 2012 5th International Symposium on Resilient Control Systems. Presented at the 2012 5th International Symposium on Resilient Control Systems (ISRCS), IEEE, Salt Lake City, UT, USA, pp. 48–54. https://doi.org/10.1109/ISRCS.2012.6309292

Linda, O., Manic, M., Vollmer, T., Wright, J., 2011b. Fuzzy logic based anomaly detection for embedded network security cyber sensor, in: 2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS). Presented at the 2011 IEEE Symposium On Computational Intelligence In Cyber Security - Part Of 17273 - 2011 Ssci, IEEE, Paris, France, pp. 202–209. https://doi.org/10.1109/CICYBS.2011.5949392

Linda, O., Vollmer, T., Manic, M., 2009. Neural Network based Intrusion Detection System for critical infrastructures, in: 2009 International Joint Conference on Neural Networks. Presented at the 2009 International Joint Conference on Neural Networks (IJCNN 2009 - Atlanta), IEEE, Atlanta, Ga, USA, pp. 1827–1834. https://doi.org/10.1109/IJCNN.2009.5178592

Liping, C., Xiaoping, W., Xiong, G., Hongchang, Z., Fanli, Z., Bin, G., Lei, W., 2012. Modeling and simulating CAN-based cyber-physical systems in Modelica, in: 2012 IEEE Sixth International Conference on Software Security and Reliability Companion. IEEE, pp. 152–157.

Lu, T., Guo, X., Li, Y., Peng, Y., Zhang, X., Xie, F., Gao, Y., 2014. Cyberphysical security for industrial control systems based on wireless sensor networks. Int. J. Distrib. Sens. Netw. 10, 438350.

Luo, Y., 2013. Research and design on intrusion detection methods for industrial control system. PhD Hangzhou China Zhejiang Univ.

Maglaras, L.A., Jiang, J., 2014. Intrusion detection in SCADA systems using machine learning techniques, in: 2014 Science and Information Conference. Presented at the 2014 Science and Information Conference (SAI), IEEE, London, UK, pp. 626–631. https://doi.org/10.1109/SAI.2014.6918252

Maheshwari, N., Dagale, H., 2018. Secure communication and firewall architecture for IoT applications, in: 2018 10th International Conference on Communication Systems & Networks (COMSNETS). IEEE, pp. 328–335.

Manson, S., Anderson, D., 2019. Cybersecurity for Protection and Control Systems: An Overview of Proven Design Solutions. IEEE Ind. Appl. Mag. 25, 14–23. https://doi.org/10.1109/MIAS.2018.2875175

McLaughlin, S., Konstantinou, C., Wang, X., Davi, L., Sadeghi, A.-R., Maniatakos, M., Karri, R., 2016. The cybersecurity landscape in industrial control systems. Proc. IEEE 104, 1039–1057.

McLaughlin, S.E., Zonouz, S.A., Pohly, D.J., McDaniel, P.D., 2014. A Trusted Safety Verifier for Process Controller Code., in: NDSS.

Mitchell, R., Chen, R., 2014. Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems. IEEE Trans. Dependable Secure Comput. 12, 16–30.

Morris, T., Vaughn, R., Dandass, Y., 2012. A Retrofit Network Intrusion Detection System for MODBUS RTU and ASCII Industrial Control Systems, in: 2012 45th Hawaii International Conference on System Sciences. Presented at the 2012 45th Hawaii International Conference on System Sciences (HICSS), IEEE, Maui, HI, USA, pp. 2338–2345. https://doi.org/10.1109/HICSS.2012.78

Neitzel, L., Huba, B., 2014. Top ten differences between ICS and IT cybersecurity. InTech 61, 12–18.

NIST, S., 2008. 800-82. Guide Ind. Control Syst. ICS Secur. Final Public Draft Natl. Inst. Stand. Technol.

Nurika, O., Aminz, A.H.B.M., Rahman, A.S.B.A., Zakaria, M.N.B., 2012. Review of various firewall deployment models, in: 2012 International Conference on Computer & Information Science (ICCIS). IEEE, pp. 825–829.

Pajic, M., Mangharam, R., Sokolsky, O., Arney, D., Goldman, J., Lee, I., 2014. Model-driven safety analysis of closed-loop medical systems. IEEE Trans. Ind. Inform. 10, 3–16.

Papa, S., Casper, W., Nair, S., 2012. A transfer function based intrusion detection system for SCADA systems, in: 2012 IEEE Conference on Technologies for Homeland Security (HST). IEEE, pp. 93–98.

Paxson, V., 1999. Bro: a system for detecting network intruders in real-time. Comput. Netw. 31, 2435–2463.

Ren, X., Blanton, R.D., Tavares, V.G., 2016. A learning-based approach to secure JTAG against unseen scan-based attacks, in: 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, pp. 541–546.

Roesch, M., 1999. Snort – Lightweight Intrusion Detection for Networks 11.

SANS Institute: Reading Room - Industrial Control Systems / SCADA [WWW Document], 2015. URL https://www.sans.org/reading-room/whitepapers/ICS/paper/36297 (accessed 6.18.19).

Schamai, W., Fritzson, P., Paredis, C.J., 2013. Translation of UML state machines to Modelica: Handling semantic issues. Simulation 89, 498–512.

Schuett, C., Butts, J., Dunlap, S., 2014. An evaluation of modification attacks on programmable logic controllers. Int. J. Crit. Infrastruct. Prot. 7, 61–68.

Seiger, R., Huber, S., Schlegel, T., 2018. Toward an execution system for self-healing workflows in cyber-physical systems. Softw. Syst. Model. 1–22.

Shin, J., Son, H., Khalil ur, R., Heo, G., 2015. Development of a cyber security risk model using Bayesian networks. Reliab. Eng. Syst. Saf. 134, 208–217. https://doi.org/10.1016/j.ress.2014.10.006

Sicard, F., Zamai, É., Flaus, J.-M., 2019. An approach based on behavioral models and critical states distance notion for improving cybersecurity of industrial control systems. Reliab. Eng. Syst. Saf. 188, 584–603. https://doi.org/10.1016/j.ress.2019.03.020

Sicard, F., Zamai, E., Flaus, J.-M., 2018. Filters based Approach with Temporal and Combinational Constraints for Cybersecurity of Industrial Control Systems. IFAC-Pap. 51, 96–103. https://doi.org/10.1016/j.ifacol.2018.09.541

Sridhar, S., Govindarasu, M., 2014. Model-based attack detection and mitigation for automatic generation control. IEEE Trans. Smart Grid 5, 580–591.

Stevens, M.W., Pope, M., 1999. An implementation of an optical data diode. Citeseer.

Stouffer, K., Lightman, S., Pillitteri, V., Abrams, M., Hahn, A., 2015a. NIST special publication 800-82 guide to industrial control systems (ICS) security-revision 2 final public draft. Natl. Inst. Stand. Technol.

Stouffer, K., Lightman, S., Pillitteri, V., Abrams, M., Hahn, A., 2015b. NIST special publication 800-82 guide to industrial control systems (ICS) security-revision 2 final public draft. Natl. Inst. Stand. Technol.

Thongkanchorn, K., Ngamsuriyaroj, S., Visoottiviseth, V., 2013. Evaluation studies of three intrusion detection systems under various attacks and rule sets, in: TENCON 2013-2013 IEEE Region 10 Conference (31194). IEEE, pp. 1–4.

Threat landscape for industrial automation systems. H2 2018, 2019. . Kaspersky Lab ICS CERT. URL https://ics-cert.kaspersky.com/reports/2019/03/27/threat-landscape-for-industrial-automation-systems-h2-2018/ (accessed 7.8.19).

Vollmer, T., Manic, M., 2009. Computationally efficient Neural Network Intrusion Security Awareness, in: 2009 2nd International Symposium on Resillient Control Systems. Presented at the 2009 2nd International Symposium on Resillient Control Systems (ISRCS), IEEE, Idaho Falls, ID, USA, pp. 25–30. https://doi.org/10.1109/ISRCS.2009.5251357

Williams, T.J., 1994. The Purdue enterprise reference architecture. Comput. Ind. 24, 141–158.

Zamaï, É., Rigaud, F., Pétin, J.-F., Berruet, P., Toguyeni, A., 2007. Architectures de pilotage de procédés industriels. Editions Techniques de l'Ingénieur.

Zang, T., Gao, S., Liu, B., Huang, T., Wang, T., Wei, X., 2019. Integrated fault propagation model based vulnerability assessment of the electrical cyber-physical system under cyber attacks. Reliab. Eng. Syst. Saf. 189, 232–241. https://doi.org/10.1016/j.ress.2019.04.024

Zhu, B., Joseph, A., Sastry, S., 2011. A taxonomy of cyber attacks on SCADA systems, in: 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing. IEEE, pp. 380–388.

Zhu, B., Sastry, S., 2010. SCADA-specific intrusion detection/prevention systems: a survey and taxonomy, in: Proceedings of the 1st Workshop on Secure Control Systems (SCS). p. 7.

# Glossaire

- CPS : Cyber Physical System, système cyber physique.
- DCS : Système de contrôle distribué.
- DDoS : Destributed Deny of Service.
- DMZ : Demilitarized Zone, zone démilitarisée.
- DNP3 : Distributed Network Protocol.
- DoS : Deny of Service.
- FBD : Boîte Fonctionnelle.
- HA : Hybrid Automata, automate hybride.
- HIDS : Host-based Intrusion Detection System, système de détection d'intrusion hôte.
- HMI : Interface homme-machine.
- IACS : Système d'automatisation et de contrôle industriel.
- ICS : Système de Contrôle Industriel.
- IDS : Système de Détection d'Intrusion.
- IL : Instruction List.
- IT : Information Technology.
- LD : Ladder Diagram.
- MitM : Man-in-the-Middle.
- NIDS : Network-based Intrusion Detection System, système de détection d'intrusion réseau.
- NIST : National Insitute of Standards and Technology (USA).
- OT : Operational Technology.
- PLC : Automate Programmable Industriel.
- RTU : Remote Terminal Unit.
- SCADA : Système de contrôle et d'acquisition de données.
- SFC : Sequential Functional Chart.

- ST : Structured Text.

# Table de figures et des tableaux