Exploration de Scénarios de Systèmes Cyber-Physiques pour l'Analyse de la Menace

Tithnara Nicolas Sun, Luka Le Roux, Ciprian Teodorov, Philippe Dhaussy

Résumé

La cybersécurité des systèmes est un besoin vital depuis que l'industrie se dirige vers l'automatisation, que ce soit dans les systèmes cyber-physiques ou dans "l'industrie du futur". Il est désormais nécessaire d'envisager la sécurité comme une problématique continue et accompagnant le système tout au long de son cycle de vie. Les méthodes de modélisation actuelles se focalisent généralement sur le système à un instant précis, que ce soit en conception ou en pentest. Cependant, ces analyses ne tiennent pas compte de l'évolution du système et ne garantissent donc pas de résultats pérennes. Dans cet article, nous proposons un environnement de modélisation et de simulation de scénarios d'attaque prenant en compte l'aspect opérationnel du système. Pour cela nous introduisons un langage de scénarios exécutable. Ce langage permet à la fois d'enrichir une modélisation statique du système d'une part et de modéliser le comportement de l'attaquant d'autre part. Nous illustrons notre approche sur une station de pompage. Nous montrons comment un expert peut capturer d'une manière abstraite sa compréhension du système afin de la confronter à une modélisation des capacités de l'attaquant. Ceci permet d'exhiber des analyses formelles de sécurité sur le système sous attaque.

1 Introduction

Les systèmes industriels se complexifient de plus en plus, du fait de l'automatisation croissante des procédés. Cela implique une dépendance grandissante en ces processus automatiques. C'est pourquoi il est crucial de garantir la sécurité de fonctionnement des systèmes face à la menace cyber [3]. L'ingénierie dirigée par les modèles préconise un ensemble de techniques permettant de répondre à ce besoin. Les systèmes sont représentés par des modèles sur lesquels peuvent s'appliquer des analyses de vérification et de validation formelles [8].

Toutefois, les méthodes de modélisation actuelles connaissent certaines limitations en sécurité. Premièrement dans un contexte *ad hoc*, c'est-à-dire sur un système existant , du point d'un attaquant opportuniste, la connaissance du système est partielle. Du point de vue de l'attaquant, le modèle du système ne peut pas être détaillé à un niveau d'abstraction trop bas. L'utilisation de modèles de spécification du système est donc à remettre en question. D'autant plus que le système possède un comportement dynamique lié à son fonctionnement. Cet aspect dynamique implique des changements de phases qui peuvent induire des faiblesses exploitables

par un attaquant pour s'introduire dans le système et/ou causer des dommages. Deuxièmement, les méthodes d'analyse ne sont généralement pas pérennes. Les résultats d'analyses ne sont valides que sur la configuration courante du système. Le système, au cours de son cycle de vie, peut changer de configuration lors de changements volontaires (maintenance) ou involontaires (panne). Ces variations mettent en doute la validité des analyses sur la durée.

Afin de résoudre ces problèmes, nous proposons un environnement de modélisation et de simulation de scénarios d'attaque. Pour ce faire, nous nous basons sur trois processus distincts sur un environnement de modélisation et de simulation : la modélisation de l'architecture du système, la modélisation du comportement nominal du système et la modélisation de l'attaquant. Cette approche modulaire permet de spécialiser séparément chaque aspect du problème et donc de concevoir des outils adéquats. Notre approche est mise en oeuvre avec le langage de modélisation statique Pimca pour l'architecture du système [10] associé à un langage de scénarios exécutable dédié Target System Modeling (TSM). Nous présenterons les langages ainsi que leur utilisation avec le model-checker OBP2 (Observer-Based Prover 2) pour les analyses formelles de sécurité. Le langage de scénarios exécutable que nous introduisons permet de capturer le comportement nominal du système et le comportement de l'attaquant. À partir de cet environnement, nous montrons comment un expert peut représenter sa compréhension du système de manière abstraite et comment produire des analyses de sécurité.

Dans cet article, nous introduirons les langages utilisés en détaillant le langage de comportement. En parallèle, nous illustrons notre approche sur le cas d'étude d'une station de pompage. Puis nous présentons nos résultats d'analyse. Enfin nous concluons en mettant en perspective notre approche avec la littérature.

2 Capture du fonctionnement du système

Dans le cadre de l'analyse de la menace de systèmes industriels, nous modélisons la structure du système étudié avec le langage Pimca [10], puis nous modélisons le comportement nominal du système à l'aide de notre langage de scénarios TSM. Enfin, nous modélisons également le comportement de l'attaquant avec le langage TSM. Notre approche est outillée avec l'environnement Openflexo, le cadre logiciel est disponible en ligne. [1] La section 2.1 introduit le cas d'étude qui illustrera l'approche. Nous présentons Pimca en 2.2 et TSM en 2.3.

2.1 Station de pompage

La station de pompage, fig.1, est un système cyber-physique contrôlant automatiquement le niveau d'un réservoir d'eau grâce à un PLC (Programmable Logic Controller). Une vanne électrique d'entrée alimente le réservoir en eau, tandis qu'une vanne manuelle de sortie, connectée à une pompe électrique, vide le réservoir. La vanne manuelle est actionnable par un technicien. La vanne électrique et la pompe sont directement contrôlées par le PLC qui relève le niveau d'eau du

1. https ://research.openflexo.org/CTA.html

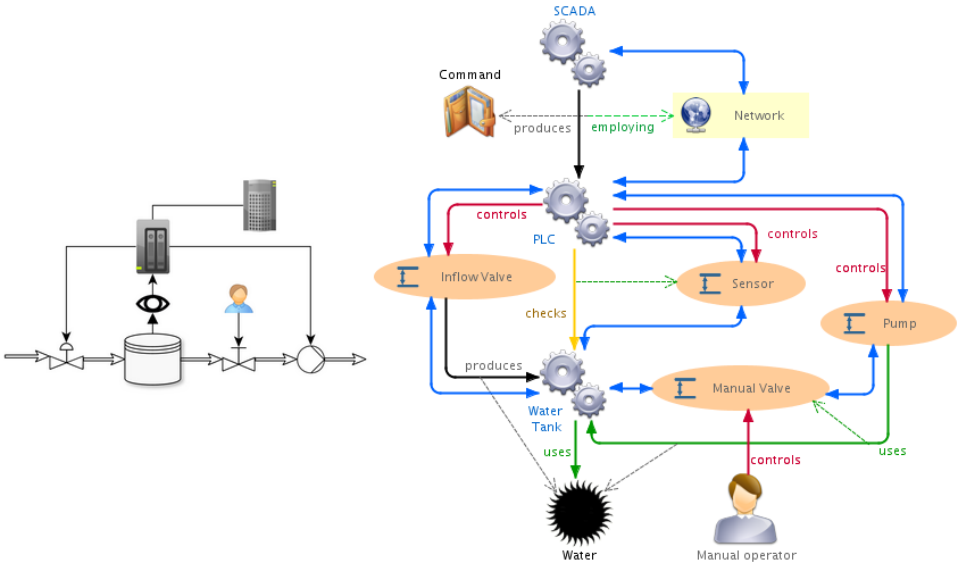

FIGURE 1 – Station de pompage : structure (gauche), modèle Pimca (droite)

réservoir à travers un capteur. De plus le PLC relaie régulièrement les mesures du niveau d'eau à une centrale SCADA (Supervisory Control And Data Acquisition) sur le réseau du site industriel. Le comportement du PLC suit les commandes suivantes : (i) ouvrir la vanne d'entrée et éteindre la pompe si le niveau d'eau atteint un seuil bas, (ii) fermer la vanne d'entrée et allumer la pompe si le niveau d'eau atteint un seuil haut, (iii) chaque mesure de niveau est envoyée au SCADA.

2.2 Pimca

Afin de conduire des analyses de sécurité, une modélisation adaptée du système est nécessaire. Pimca [10] est un langage graphique de modélisation système pour la sécurité qui met en lumière la surface d'attaque [5], l'ensemble des points d'interactions avec le système pour l'attaquant. C'est un langage à haut niveau d'abstraction qui satisfait le contexte de modélisation *ad hoc*. Pimca modélise le système en différents composants pourvus de comportements (*machinerie*). Les éléments du système susceptibles d'être cibles d'un attaquant sont représentés par des *ressources*. Ces composants sont liés entre eux via des *relations* expressives.

À partir de la description du système, nous identifions les différents composants qui se traduisent en éléments de modèle Pimca, fig.1. Le réservoir, les vannes, la pompe, le capteur, le technicien, le PLC et le SCADA présents dans le schéma, fig.1, sont capturés par des machineries car ils possèdent un comportement. De plus, nous ajoutons au modèle le réseau du site industriel à travers lequel communiquent le SCADA et le PLC, car c'est un point d'échange actif du système et donc une machinerie. Enfin, nous modélisons deux cibles potentielles pour un attaquant, à savoir les instructions suivies par le PLC et l'eau qui circule dans le système.

Ensuite nous identifions les différentes relations du modèle Pimca. Les connexions entre les composants représentés dans le schéma initial sont modélisées

par la relation bidirectionnelle *échange*. Pour les relations de plus haut-niveau, nous pouvons distinguer différentes classes : (i) le *contrôle* que le PLC exerce sur la vanne d'entrée, la pompe et le capteur, (ii) le *contrôle* exercé par le technicien sur la vanne manuelle, (iii) la *production* d'eau au regard du système de la vanne d'entrée pour le réservoir, (iv) l'*utilisation* de l'eau du réservoir par la pompe, (v) la *production* de commandes du SCADA au PLC, (vi) l'*utilisation* de l'eau par le réservoir et (vii) la *vérification* du niveau du réservoir par le PLC.

Ce modèle statique capture la structure du système et permet des analyses de sécurité telles que la déduction de surface d'attaque présentée dans [10]. En exploitant les relations expressives, il permet un raisonnement itératif sur les cibles intermédiaires et les points d'entrées potentiels . Toutefois pour conduire des analyses de sécurité tenant compte des aspects dynamiques du système, il est nécessaire de capturer le comportement dynamique.

2.3 Target System Modeling

Afin de nous focaliser sur cette problématique, nous introduisons un nouveau langage exécutable dédié, Target System Modeling (TSM). À travers TSM, nous spécifions ces comportements pour exécuter, simuler et produire des analyses. ***Comportement nominal :*** Dans Pimca, les *machineries* sont définies comme les éléments pourvus de comportement. Le comportement du système est modélisé par un ensemble de machines à états synchronisées. L'ensemble des comportements d'un composant constitue une machine à états qu'on appelle une unité d'exécution. Les unités d'exécutions sont reliées entre elles (une-à-une), à la manière des machineries de Pimca, à travers des canaux de synchronisation. Ensuite, nous capturons la configuration du système global dans un ensemble de variables dont certaines sont globales et d'autres sont internes à une unité d'exécution donnée.

TSM utilise un formalisme d'actions similaire au formalisme introduit par Dijkstra [2] pour capturer le comportement au sein d'une unité d'exécution. Une action TSM est définie comme un couple de garde/commande. La garde est une expression booléenne qui détermine si l'action est déclenchable. La commande décrit les effets de l'action. Ceci permet de modéliser séparément chaque composant du système. De plus, pour permettre la communication synchrone entre les différentes unités d'exécutions, une action peut être assignée à un canal de synchronisation. Cela signifie que l'action doit être déclenchée en synchronisation avec une action de l'autre côté du canal en un seul pas atomique. Cependant, ce formalisme ne permet pas la synchronisation de plus de deux unités d'exécution à la fois, c'est pourquoi nous introduisons la notion d'action urgente. Les actions urgentes doivent être déclenchées en priorité par rapport aux autres actions. Ceci permet de gérer les synchronisations entre de nombreuses unités d'exécution en les considérant comme des synchronisations une-à-une urgentes qui se résolvent toutes avant que le système ne puisse évoluer, via des actions non urgentes. Le prototype d'une action est donc : $ID : urgent?\ chanID(?\ |!)([garde])?/(commande;)*$.

Dans le cas de la station de pompage, certaines instances de machineries

(réseau, capteur, vanne manuelle) sont simplement des relais, leur comportement consiste à relayer un message d'un élément émetteur du système à un autre élément récepteur. Par exemple, le niveau d'eau du réservoir est transmis au PLC à travers le capteur. Ceci est capturé par une action synchronisée avec l'émetteur pour réceptionner le message et par une action urgente synchronisée avec le récepteur pour envoyer le message. Concrètement le capteur est une unité d'exécution possédant 2 variables (*waterLevel, isTriggered*) et contenant 2 actions :
-$reception : mesure? / waterLevel := value; isTriggered := true;$
-$emission : urgent\ updatePLC! [isTriggered] / isTriggered := false;$

Le réservoir d'eau possède un niveau qui fluctue en fonction de l'arrivée d'eau depuis la vanne électrique et la sortie depuis la vanne manuelle. De plus, le niveau d'eau est relevé par le capteur lorsqu'il évolue. Ce comportement est capturé par des actions synchronisées pour augmenter le niveau lorsque la vanne électrique envoie de l'eau et diminuer le niveau lorsque la vanne manuelle pompe de l'eau. Le niveau d'eau est relayé au capteur à chaque changement par une action urgente synchronisée avec le capteur. La pompe et la vanne électrique ont une action qui leur permet de déclencher le changement de niveau d'eau dans le réservoir quand ils sont en marche. Leur état de marche est contrôlé par le PLC à travers une action synchronisée. La vanne manuelle peut être fermée ou ouverte par un technicien.

Le PLC a un comportement plus complexe. Sa commande stipule que le niveau d'eau du réservoir doit être maintenu entre un seuil bas et un seuil haut donnés. Pour ce faire, à chaque relevé de niveau d'eau dans l'action synchronisée avec le capteur, le PLC détermine si le système est à un niveau convenable ou s'il approche les valeurs à risques. Ensuite une action urgente synchronisée avec le SCADA, à travers le réseau, permet de relayer cette information. Enfin, si le PLC a déterminé que le niveau atteignait prochainement le seuil haut, deux actions urgentes synchronisées ordonnent à la pompe et à la vanne électrique de se mettre en marche et de s'éteindre respectivement. Dans le cas du seuil bas, le contraire est modélisé. Le SCADA réceptionne les messages du PLC. Il possède une action déclenchée lorsque le niveau est anormal qui met le SCADA en alerte.

À partir de ce modèle de système, nous simulons pas-à-pas le comportement nominal pour vérifier et valider le comportement du système. Pour conduire des analyses de sécurité, il reste toutefois à modéliser le comportement de l'attaquant

Comportement de l'attaquant : En utilisant le langage TSM, nous étendons le modèle pour permettre l'intervention d'un attaquant. Pour cela, nous spécifions des possibilités d'interactions avec les composants du systèmes, c'est-à-dire les unités d'exécutions. Nous définissons ces interactions comme des altérations du comportement nominal. Elles sont modélisées par l'ajout d'actions d'attaque au sein des unités d'exécution. L'objectif de l'attaquant est de causer un dysfonctionnement du système. Nous recherchons les scénarios possibles qui peuvent mener à cet objectif à travers des actions d'attaque. Notre modélisation d'attaquant revient donc à capturer des capacités d'interactions que l'attaquant a avec le système.

Dans le cas de la station de pompage, l'attaquant a deux objectifs : (i) causer un débordement du réservoir et (ii) ne pas déclencher l'alerte du SCADA en cas de

Forcer vanne d'entrée		•					•	•	•		•	•
Fermer vanne manuelle			•						•	•		•
Bloquer pompe				•				•		•	•	
Brouiller réseau					•		•			•	•	•
Couper capteur						•						
Objectif 1	X	X	X	X	X	**O**	X	O	O	X	**O**	**O**
Objectif 2	-	-	-	-	-	**O**	-	X	X	-	**O**	**O**

TABLE 1 – Model-checking de la station de pompage (O : succès, X : échec)

succès du débordement. Voici la liste des actions d'attaque que nous modélisons : (i) forcer l'ouverture de la vanne électrique, (ii) fermer la vanne manuelle, (iii) bloquer la pompe, (iv) brouiller le réseau, (v) couper le capteur.

3 Analyse de sécurité

Le modèle exécutable nous permet de rechercher des scénarios d'attaque en fonction des capacités de l'attaquant grâce au model-checking. En effet, les objectifs de l'attaquant peuvent être formulés en propriétés LTL qui peuvent être vérifiées. Si ces propriétés sont violés alors l'attaquant peut atteindre ses objectifs. Dans notre cas, les propriétés sont les suivantes :

1. "`[] !|waterOverflow|`"
2. "`[] (|waterOverflow| -> <>|scadaAlert|)`"

Les propositions `|waterOverflow|` et `|scadaAlert|` rédigées en TSM expriment respectivement que le réservoir déborde et le SCADA est en alerte.

Notre implémentation exécutable de TSM utilise Java et peut être connectée au model-checker OBP2 pour la simulation et l'analyse du comportement. OBP2 manipule la configuration d'une instance du modèle TSM. Il permet d'évaluer des prédicats et de calculer l'ensemble des transitions tirables vers de nouvelles configurations à partir de la configuration courante. L'interface d'OBP2 permet de simuler pas à pas le comportement du système. OBP2 garde en mémoire la configuration courante ainsi que la trace d'exécution ce qui permet de recharger une configuration antérieure. L'interface permet également de déclencher les transitions tirables depuis la configuration courante pour une exploration manuelle. Par ailleurs OBP2 permet de vérifier des propriétés LTL avec GPSL [1]. Lors de l'exploration automatique impliquant la composition d'automates de Büchi, le model checker explore l'espace d'états du système et indique si une propriété est valide ou non.

Dans un premier temps, nous avons simulé le comportement nominal du système, c'est-à-dire sans qu'aucune action d'attaque ne soit possible pour vérifier que le système fonctionnait sans action malveillante. Ensuite nous avons vérifié les propriétés 1 et 2 sur l'ensemble des combinaisons de capacités d'attaquant possibles pour déterminer les capacités qui lui permettent d'atteindre ses objectifs.

La table 1 montre les résultats de notre analyse avec l'outil OBP2 et l'exploration des configurations du système. Chaque colonne représente une combinaison de capacités de l'attaquant ainsi que le fait que l'attaquant puisse atteindre son

objectif 1 (faire déborder le réservoir) et son objectif 2 (ne pas se faire répérer par le SCADA en cas de débordement). On peut notamment voir que l'attaquant peut atteindre ses deux objectifs avec une seule capacité : "couper le capteur". On peut également noter que les capacités "fermer la vanne manuelle" et "bloquer la pompe" sont équivalentes au regard des objectifs, il est donc inutile pour l'attaquant d'utiliser ces deux capacités. Enfin on peut voir que si l'attaquant ne peut pas couper le capteur, il doit alors brouiller le réseau, forcer la vanne d'entrée et arrêter le flux de sortie d'une manière ou d'une autre pour arriver à ses fins.

Cette analyse du modèle TSM de la station de pompage exhibe un point critique dans le fonctionnement du système. Le capteur est un élément essentiel puisque sa mise hors service permet à l'attaquant d'atteindre ses deux objectifs. Le modèle et les analyses sont disponibles en ligne. [2]

4 État de l'art

L'exploration de scénarios de systèmes cyber-physiques pour l'analyse de la menace est un problème qui alimente de nombreux travaux de recherche.

Certaines approches produisent des analyses d'impacts [7, 9]. Celles-ci considèrent la dynamicité du modèle et permettent de capturer le comportement du système pendant une attaque. Néanmoins, la modélisation de l'attaquant est insuffisante dans le premier cas [7] pour capturer des scénarios d'attaques complexes. Dans le second cas [9], le formalisme d'automates permet de représenter des impacts d'attaques plus complexes au prix d'une modélisation intrusive aux composants (ajouts d'états et transitions) ce qui demande une compréhension poussée du fonctionnement du système et pose des problèmes de pérennité. Il est donc difficile d'appliquer ces approches dans un contexte *ad hoc*.

D'autres approches sont basées sur le formalisme d'arbres d'attaque [4, 8]. Les arbres d'attaques permettent de modéliser des scénarios d'attaque dans un formalisme expressif. Contrairement à notre approche, ces outils mettent en jeu de multiples niveaux d'abstraction, ce qui requiert une compréhension poussée du fonctionnement système et de ses vulnérabilités. Ceci ne convient pas à un contexte de modélisation *ad hoc* et limite l'extensibilité de la modélisation, donc sa pérennité.

5 Conclusion

L'ingénierie dirigée par les modèles peut apporter des réponses pertinentes à la sécurité des systèmes cyber-physiques. Nous proposons un environnement de modélisation et de simulation de systèmes utilisant deux langages : Pimca pour la modélisation structurelle et TSM pour la modélisation comportementale. Notre approche exécutable est implémentée avec OpenFlexo, connectée au model-checker OBP2 et validée sur le cas d'étude de la station de pompage.

L'approche repose sur la complémentarité Pimca-TSM pour répondre au besoin de modélisation *ad hoc* des systèmes. D'une part, elle capture d'une manière abstraite la dynamique du système pour modéliser ses changements de phases.

2. https ://github.com/Lawyne/pimca-tsm-afadl20

D'autre part, elle permet de capturer d'une manière peu intrusive le comportement de l'attaquant. En effet, les actions sont aisément extensibles et se prêtent donc mieux à l'extension de modèles que les formalismes d'automates ou d'arbres.

Il reste néanmoins à étudier la possibilité d'étendre le formalisme TSM avec une couche réflexive[6] nous permettant la modélisation non-intrusive et réutilisable de l'attaquant sur des systèmes plus complexes.

Remerciements : Nous tenons à remercier la Direction Générale de l'Armement (DGA) qui finance en partie ce projet.

Références

[1] M. Brumbulli, E. Gaudin, and C. Teodorov. Automatic Verification of BPMN Models. In *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, Toulouse, France, 2020.

[2] E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8) :453–457, 1975.

[3] A. Ginter. The Top 20 Cyberattacks on Industrial Control Systems. Technical report, Waterfall Security Solutions, 2017.

[4] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer. DAG-based attack and defense modeling : Don't miss the forest for the attack trees. *Computer science review*, 13 :1–38, 2014.

[5] P. K. Manadhata and J. M. Wing. An attack surface metric. *IEEE Transactions on Software Engineering*, 37(3) :371–386, 2011.

[6] N. Papoulias, M. Denker, S. Ducasse, and L. Fabresse. End-user abstractions for meta-control : Reifying the reflectogram. *Science of Computer Programming*, 140 :2 – 16, 2017. Object-Oriented Programming and Systems (OOPS 2015).

[7] E. Peterson. Dagger : Modeling and visualization for mission impact situation awareness. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 25–30. IEEE, 2016.

[8] S. Pinchinat, M. Acher, and D. Vojtisek. ATSyRa : an integrated environment for synthesizing attack trees. In *International Workshop on Graphical Models for Security*, pages 97–101. Springer, 2015.

[9] B. Sultan, F. Dagnat, and C. Fontaine. A methodology to assess vulnerabilities and countermeasures impact on the missions of a naval system. In *Computer Security*, pages 63–76. Springer, 2017.

[10] T. N. Sun, B. Drouot, J. Champeau, F. R. Golra, S. Guérin, L. Le Roux, R. Mazo, C. Teodorov, L. Van Aertryck, and B. L'Hostis. A Domain-Specific Modeling Framework for Attack Surface Modeling. In *Proceedings of the 5th International Conference on Information Systems Security and Privacy*, pages 341–348. SCITEPRESS, 2020.