Validation Formelle d'Architectures Logicielles Basée sur des Patrons de Sécurité

Fadi Obeid Philippe Dhaussy
Lab-STICC CNRS, UMR 6285 Ensta Bretagne
fadi.obeid@ensta-bretagne.org
philippe.dhaussy@ensta-bretagne.fr

May 4, 2018

Abstract

Les modèles de patrons de sécurité ont été proposés comme des solutions méthodologiques permettant de modéliser des mécanismes qui répondent à des problèmes de sécurité récurrents. Ceux-ci sont décrits dans la littérature et peuvent être exploités dans différents contextes de modélisation. Lors de leur intégration au sein d'un modèle d'architecture, ces modèles de patrons sont à adapter à ses spécificités. Une fois les modèles de patrons intégrés, il est nécessaire de valider formellement le résultat de cette intégration au regard des propriétés fonctionnelles de l'architecture initiale qui doivent être préservées, et au regard des propriétés formelles de sécurité associées aux patrons. Dans notre travail, nous exploitons une technique de model-checking pour la vérification des propriétés. Nous cherchons à exploiter notre approche dans le cadre de la modélisation des architectures SCADA.

Mots Clés: Patron de Sécurité, Model Checking, SCADA

1 Introduction

Un modèle de patron de sécurité est une solution méthodologique réutilisable pour un problème de sécurité récurrent. Divers patrons ont été décrits dans la littérature [1]. Associés à ces modèles, certains auteurs ont proposé des méthodologies pour leur mise en œuvre et leur intégration dans des modèles d'architectures logicielles. La description d'un patron inclut une description de ses propriétés qui doivent être respectées lors de leur intégration dans l'architecture à sécuriser. Il offre au développeur un guide sur la façon d'utiliser et de mettre en œuvre une solution de sécurité en fonction du contexte précis d'un problème de sécurité et d'un modèle d'architecture qu'il souhaite sécuriser. De ce fait, les patrons facilitent la communication entre experts et non-experts du domaine de la sécurité. L'intégration d'un modèle de patron dans un modèle d'architecture impose des adaptations lors de la conception de l'architecture à sécuriser. Dans ce travail, nous étudions une méthodologie et les concepts pour intégrer des modèles de patrons, conformes à des politiques et des exigences de sécurité. Les modèles d'architectures générés doivent pouvoir être validés formellement au regard des exigences attendues. Pour l'instant nous raisonnons sur des modèles d'architectures génériques dont nous décrivons les caractéristiques section 2. Sur la base de ces types de modèle, des

Étude financée par la DGA-Maîtrise de l'Information et Brest Métropôle Océanne(BMO).

bibliothèques de patrons de la littérature et d'une politique de sécurité, nous cherchons à générer des modèles sécurisés. Ceux-ci doivent pouvoir être exploités pour des vérifications formelles des propriétés fonctionnelles du modèle et des propriétés de sécurité. Bien que nous raisonnons aujourd'hui sur des modèles abstraits d'architectures, nous pensons que cette approche peut s'adapter à des cas de modèles de systèmes de supervision et d'acquisition de données (SCADA). Ces systèmes contrôlent de nombreuses infrastructures et procédés critiques, telles que les installations nucléaires et les usines chimiques. Dans un contexte de cyber sécurité, des efforts conséquents doivent être portés sur la conception des mécanismes de protection. Les approches de modélisation et de validation formelles ont tout à fait leur place dans ce contexte. Dans la suite de ce papier, nous donnons, section 2, quelques références de l'état de l'art et nous présentons les principes de formalisation des patrons étudiés. Nous illustrons en section 3, un patron particulier de type *Single Acces Point*. Nous précisons les types de modèles d'architectures sur lesquels portent nos travaux actuels. Ensuite, section 4, nous décrivons notre approche pour l'intégration des modèles de patrons et les principes de validation de propriétés. Nous concluons, section 5, par un bilan et les perspectives de ce travail.

2 Les patrons de sécurité

Un état de l'art sur les modèles de patrons de sécurité peut être trouvé entre autre dans [2, 3]. De nombreux efforts de recherche ont été faits sur la vérification des modèles de conception intégrant des combinaisons de patrons de sécurité. D'un point de vue modélisation, les travaux décrits dans [4] nous ont inspirés dans le sens où des schémas de modélisation UML sont décrits, intégrant des mécanismes de sécurité dans des modèles applicatifs simples. Nous souhaitons étendre ce travail pour des modèles d'architectures et des politiques de sécurité plus complexes et en prenant en compte non seulement les propriétés de sécurité associées aux patrons mais également les propriétés fonctionnelles initiales des architectures. De nombreuses études ont traité de la sécurité dans les SCADA. Ces études fournissent des voies de recherche pour différentes approches pour améliorer la sécurité dans ces systèmes.

La sécurisation d'une architecture implique en général l'intégration de plusieurs mécanismes de sécurité. La gestion de l'ensemble d'une politique de sécurité est donc partagée entre plusieurs dispositifs. Dans une approche orientée modèles, plusieurs modèles de patrons sont impliqués dans la construction d'un modèle d'architecture. Nous précisons ici que les types de modèle que nous considérons dans notre travail sont organisés comme un ensemble d'entités (ou composants), communiquant par des liens de communication ($fifo$ ou $ports\ synchrones$). Chaque entité a un comportement décrit par un automate. Sur les liens de communication, sont véhiculés des messages comportant des données (requêtes ou réponses) impliquant des accès à des ressources (mémoires) détenues par les entités. L'architecture interagit avec un environnement dans lequel des entités clientes sollicitent les services rendus par l'architecture. Le principe de la sécurisation

d'une telle architecture est de déployer les mécanismes de sécurité au sein de ce réseau d'entités.

Compte tenu de l'avancée de nos travaux, nous nous sommes focalisés sur quatre patrons que nous pourrons étendre par la suite sur les mêmes bases méthodologiques. Le patron $SingleAccessPoint(SAP)$ concentre les ports d'accès permettant de gérer les contrôles de sécurité au travers d'un seul point d'accès. Le patron $CheckPoint(CHP)$ est dédié à réaliser certaines vérifications relatives à la *politique* de sécurité qui est souhaitée être implantée. Le patron $Authorization$ complète les contrôles de sécurité consacrée à des droits d'accès. Le patron $FireWall$ implante des mesures de sécurité consacrées aux restrictions et filtrages des messages. Ces patrons intègrent les entités et concourent ensemble aux différents aspects liés aux exigences de sécurité devant être implantées. Dans la figure 1.a, nous illustrons la cohabitation entre deux patrons SAP et CHP au sein d'un composant.

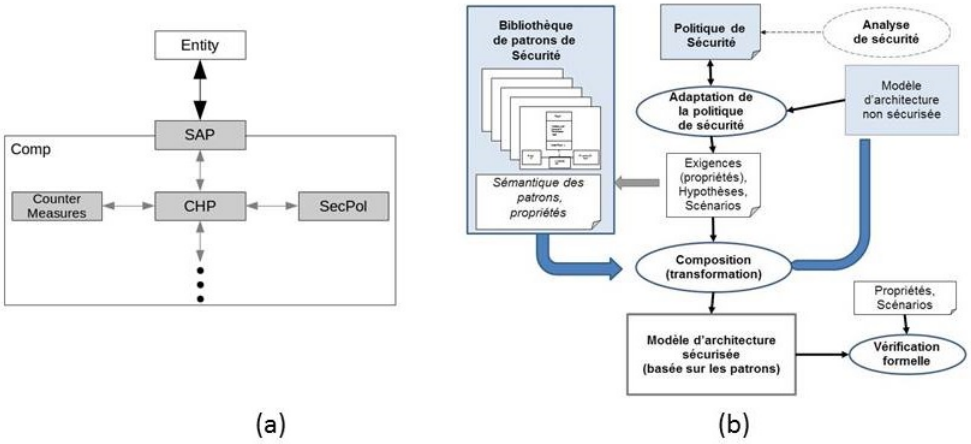

Figure 1: Processus d'intégration de patrons au sein d'une architecture.

3 Formalisation du patron Single Access Point

En guise d'illustration, nous présentons ici plus en détail le mécanisme de type SAP. Le patron SAP, introduit par [5], a pour objectif d'implanter un point d'accès unique afin d'optimiser le contrôle et la surveillance des messages d'entrées à un composant ou un ensemble de composants. Toute requête entrante est acheminée via un canal unique, où la surveillance peut être effectuée aisément. Le point d'accès unique est un endroit approprié pour éventuellement capturer un historique (Log) des accès. Ces données peuvent être utiles pour vérifier le séquencement des accès en fonction de leurs droits. SAP peut aussi avoir pour rôle de vérifier et d'apposer des signatures sur des requêtes. Il peut également avoir pour mission le contrôle d'accès à un ensemble de ressources localisées au sein d'un composant ou le contrôle d'accès à un ensemble de composants. Lors d'une demande d'accès à une ressource res détenue par un composant c, SAP implanté dans un composant

c vérifie si res est accessible au sein de c. Dans le cas d'une demande de transfert d'une requête à un autre composant c', le contrôle vérifie si c' est accessible depuis c. La mise en œuvre de SAP peut impliquer d'autres patrons, tel que CHP (figure 1.a), pour réaliser des contrôles sur les données transitant par ce point.

Une politique de sécurité est décrite par des objectifs de sécurité qui se déclinent en un ensemble de propriétés de sécurité qui peuvent être définies sous une forme littéraire ou formelle. Ces propriétés sont regroupées en trois grandes classes: confidentialité, intégrité et disponibilité. Chaque propriété représente les conditions que le système doit respecter. Une définition incorrecte ou l'application partielle d'une politique, peut entraîner le système dans un état non sécurisé. Parmi les exigences de SAP, voici deux exemples. La communication entre les composants internes n'est pas divulguée aux entités externes (*Confidentialité*). Les requêtes internes ne peuvent pas être modifiées par des entités externes (*Intégrité*). Dans notre travail, nous avons cherché à formaliser l'ensemble des exigences caractérisant chaque patron. Un exemple de propriété formelle est illustrée ci-dessous par l'invariant (1). Elle exprime que tout accès à une ressource (res) a été contrôlée auparavant ($checked$).

$$\begin{array}{l} \forall\, c \in \mathcal{S}ap,\, \forall\, e \in \mathcal{E}nt,\, \forall\, res \in \mathcal{R}es, \\ \Box\,(accessed\,(c,\, e,\, res) \Rightarrow checked_c\,(c,\, e,\, res)) \end{array} \quad (1)$$

Dans (1), $\mathcal{S}ap$ est l'ensemble des composants c détenant le mécanisme de type SAP, $\mathcal{E}nt$ est l'ensemble des entités demandant l'accès à la ressource res. $accessed\ (c,\ e,\ res)$ est le prédicat signifiant que e a accédé à la ressource res localisée sur c. $checked_c\ (c,\ e,\ res)$ est le prédicat signifiant que la requête de demande d'accès par e à la ressource res a été contrôlée. Lors de notre étude, nous avons ainsi formalisé l'ensemble des propriétés pour les quatre patrons étudiés, sous la forme de propriétés invariantes, comme (1), ou de propriétés de type vivacité. Nous indiquons dans la suite du papier quelques principes que nous proposons, d'une part, pour intégrer les patrons au sein d'un modèle d'architecture et, d'autre part, pour vérifier les propriétés sur le modèle suite à cette intégration.

4 Processus d'intégration des patrons

Dans notre approche, nous proposons un processus d'intégration des patrons dans un modèle d'architecture. Le processus invoque des éléments (en grisé sur la figure 1.b) pris en entrée de ce processus. Tout d'abord, nous considérons un modèle d'architecture non sécurisée du type décrit précédemment. Ensuite, une politique de sécurité est choisie. Elle est décrite sous la forme d'exigences de sécurité. La mise en œuvre de cette politique implique le choix de patrons de sécurité. Le choix des patrons est effectué pour l'instant par le concepteur au regard des exigences de sécurité qu'il souhaite implanter dans son modèle d'architecture. Les exigences de sécurité sont ensuite spécifiées sous la forme d'un ensemble de propriétés formelles adaptées au modèle de l'architecture à sécuriser. La réécriture des exigences constituant la politique de sécurité en propriétés formelles est basée

sur l'ensemble des propriétés formalisées pour chaque patron. En complément des propriétés, un ensemble de scénarios sont identifiés. Ils décrivent les scénarios nominaux d'interaction entre l'architecture et son environnement ainsi que les scénarios d'attaque contre lesquels l'architecture est supposée se protéger. Pour intégrer les mécanismes de sécurité sur chaque entité de l'architecture, nous avons proposé des principes de transformation des automates de l'architecture initiale. Les transformations sont basées sur l'ajout d'états et de transitions dans les automates et des appels de fonctions associées aux différents patrons.

Sur le nouveau modèle issu de la transformation, une validation formelle doit être réalisée. Nous choisissons, dans notre approche, de la réaliser à l'aide d'une technique de $model - checking$. Celle-ci permet de vérifier, lors d'une simulation exhaustive, toutes les propriétés formelles identifiées sur ce nouveau modèle. Les propriétés sont de deux ordres : les propriétés initiales de l'architecture non sécurisée et les propriétés ajoutées par les patrons qui correspondent aux contraintes de sécurité. Pour cette vérification, les scénarios d'emploi nominaux seront exploités pour valider le fonctionnement de l'architecture. Mais aussi, des scénarios simulant des attaques doivent être exploités pour pouvoir tester la robustesse de l'architecture. Actuellement ces derniers sont spécifiés pour décrire des interactions simples avec les fifos internes de l'architecture. Des travaux en cours seront exploités pour établir des politiques d'attaques plus sophistiquées.

Nous avons mené des expérimentations sur différents modèles de réseau d'automates. Pour mener les vérifications formelles de propriétés, nous générons les modèles au format FIACRE[1] qui permettent de spécifier les comportements de notre architecture ainsi que ses interactions avec son environnement. Le but est de prouver que le modèle d'architecture généré, et donc doté des mécanismes basés sur les patrons de sécurité, respecte les exigences décrites dans la politique de sécurité choisie au regard d'attaques. Dans notre travail, les vérifications formelles sont réalisées avec l'outillage OBP[2] qui a été développé dans l'équipe [6]. Nous formalisons les exigences de sécurité à vérifier avec le langage CDL associé à cet outil. Les propriétés formelles de sûreté sont des invariants ou des observateurs de rejet. Les propriétés de vivacité sont spécifiées par des formules logiques LTL. Celles-ci seront bientôt prises en charge par une extension d'OBP ($Plug$), en cours de développement. Des scénarios nominaux, ainsi que des scénarios d'attaques peuvent aussi être décrits en CDL, ce qui permet de profiter des travaux sur la réduction de la complexité lors des explorations des modèles [7], ce qui est un aspect problématique bien connu du $model - checking$.

5 Conclusion

Dans ce travail, nous étudions les principes de transformation de modèles d'architectures basés sur l'intégration de mécanismes ou patrons de sécurité. Notre objectif est de

[1] Langage défini dans le cadre du projet TopCased (http://www.topcased.org).

[2] OBP est accessible librement sur http://www.obpcdl.org ainsi que les codes (Fiacre et CDL) de quelques modèles d'expérimentation.

fournir des outils méthodologiques pour valider formellement un modèle d'architecture sécurisée devant respecter une politique de sécurité donnée. Nous avons expérimenté des choix d'implantation de patrons au sein des modèles qui peuvent être optimisés. Un travail complémentaire doit être conduit pour optimiser ces implantations et identifier les bons critères de comparaison entre les différentes stratégies.

Notre approche implique une formalisation complète des patrons de sécurité que nous souhaitons prendre en compte. Notre travail est à poursuivre en prenant en compte d'autres patrons de la littérature pour lesquels les formalisations se sont pas disponibles. Aussi, des combinaisons de patrons doivent être étudiées pour répondre à des exigences de sécurité plus sophistiquées. Nous constatons, sur ce point, que la modélisation des attaques est fondamentale car ce sont elles qui dirigent le processus de sécurisation. Dans notre travail, nous sommes donc confrontés à une problématique où beaucoup de paramètres sont à prendre en compte pour répondre au besoin de sécurisation des architectures. Les perspectives de ce travail concernent plusieurs axes à prendre en compte dans différentes directions. Ils concernent la prise en compte des politiques de sécurité qui peuvent être complexes, être définies de manière dynamique, porter sur des niveaux différents des architectures (accès aux ressources, aux données, aux canaux de communication). Ces politiques doivent reposer sur une formalisation indispensable à partir de laquelle, un processus de génération automatique de modèles d'architectures peut être envisagé.

References

[1] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. *Security Patterns: Integrating security and systems engineering*. John Wiley & Sons, 2013.

[2] Nobukazu Yoshioka, Hironori Washizaki, and Katsuhisa Maruyama. A survey on security patterns. *Progress in informatics*, 5(5):35–47, 2008.

[3] Eduardo B Fernandez and Rouyi Pan. A pattern language for security models. In *proceedings of PLOP*, volume 1, 2001.

[4] Ronald Wassermann and Betty HC Cheng. Security patterns. In *Michigan State University, PLoP Conf.* Citeseer, 2003.

[5] Joseph Yoder and Jeffrey Barcalow. Architectural patterns for enabling application security. *Urbana*, 51:61801, 1997.

[6] Philippe Dhaussy, Frédéric Boniol, Jean-Charles Roger, and Luka Leroux. Improving model checking with context modelling. *Advances in Software Engineering*, 2012.

[7] Luka Le Roux Ciprian Teodorov, Philippe Dhaussy. Environment-driven reachability for timed systems : Safety verification of an aircraft landing gear system. *Int. Software Tools for Technology Transfer (STTT)*, 2016.