MBT_Sec – Model-Based Testing for Security Components

Julien Botella[1], Frédéric Dadeau[2], Elizabeta Fourneret[2], Bruno Legeard[1,3], Julien Lorrain[2] and Romain Sibre[2]

[1]Smartesting Solutions & Services, Besançon, France
[2]FEMTO-ST/DISC - Université de Franche-Comté, Besançon, France

Résumé

Nous présentons dans ce papier le projet ANR ASTRID Maturation MBT_Sec (Model-Based Testing for Security Components). Ce projet vise à l'intégration et à l'industrialisation de travaux de recherche ayant pour objectif le test fonctionnel de sécurité, appliqué aux composants cryptographiques.

1 Contexte du projet

Les composants de sécurité présentent un caractère stratégique pour la sécurité des systèmes informatiques, tant dans le domaine Défense (pour le chiffrement et l'authentification par exemple) que dans le domaine Civil (au travers des cartes à puce et des systèmes de contrôle d'accès par exemple). Le test de sécurité de ces composants vise à garantir que les propriétés de sécurité définies, telles que la confidentialité ou l'intégrité, sont correctement implémentées au sein du composant. Il s'agit aussi de tester des vulnérabilités éventuelles du composant lorsque celui-ci est soumis à des attaques ou à des comportements malveillants.

Le projet MBT_Sec [1] vise le développement d'un environnement de génération automatique de tests de sécurité dédiés aux composants de sécurité. Il s'agit d'accroître la maturité technologique des résultats obtenus au sein du projet ASTRID OSeP [2] pour la faire passer d'un TRL 4 [3] actuellement à un TRL 5-6 et au delà.

Le projet MBT_Sec est fondé sur le partenariat entre Smartesting Solutions & Services, éditeur de logiciel, une PME innovante issue de la recherche, spécialisée sur les techniques de génération automatique de tests et l'Institut FEMTO-ST et notamment le Département Informatique des Systèmes Complexes. Dans le cadre des projets ASTRID (Accompagnement Spécifique des Travaux de Recherches et d'Innovation Défense), MBT_Sec s'effectue également en collaboration avec la DGA Maîtrise de l'Information / Sécurité des Systèmes d'Information, Département "Expertise de Logiciels Sécurisés" assure le suivi technique du projet, en étant partie prenante pour l'expérimentation et la validation des résultats.

Nous présentons dans ce résumé les résultats obtenus dans les projets précédents et faisant l'objet de ce présent besoin de maturation. Nous décrivons ensuite les premiers résultats obtenus suite à l'industrialisation de cette approche, au sein de l'outil de génération de tests Smartesting CertifyIt.

2 Combiner les approaches pour le test de sécurité

L'approche proposée dans le cadre du projet MBT_Sec s'articule autour du principe de test à partir de modèle [1] (MBT). Celui-ci consiste à utiliser un modèle fonctionnel de l'application

1. http ://projects.femto-st.fr/mbt_sec/
2. http ://osep.univ-fcomte.fr/
3. TRL = Technology Readiness Level

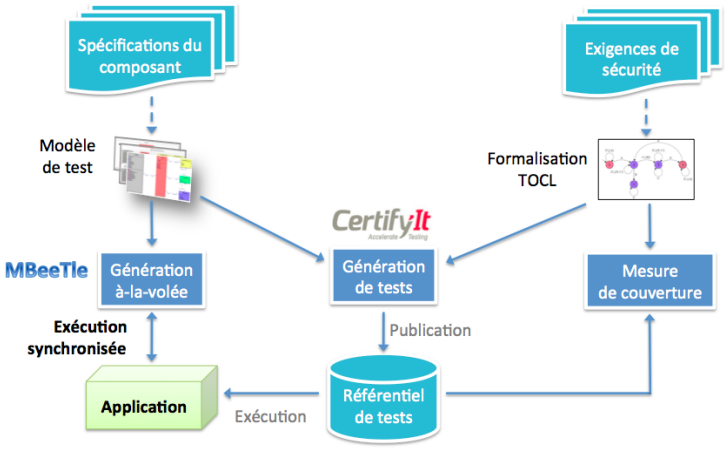

FIGURE 1 – Le processus de test de sécurité fonctionnelle proposé par MBT_Sec

pour décrire son comportement, et ainsi générer des tests, dits abstraits, visant à couvrir les comportements présents dans le modèle.

La figure 1 présente le processus MBT que nous souhaitons industrialiser. Le processus MBT classique démarre sur le côté gauche : un modèle de test est conçu par un ingénieur validation pour décrire le comportement du système qu'il veut valider [3]. Ici, nous utilisons des modèles décrits en UML/OCL. Un diagramme de classe permet de décrire le modèle du composant. Il est complété par un diagramme d'objets qui décrit l'état initial du système. La partie du dynamique du modèle, à savoir les comportements des opérations spécifiées, est décrite à l'aide de pré- et postconditions OCL, définissant les branches d'exécution du code. Suite à cette étape de modélisation, des critères de sélection de tests sont appliqués pour produire des cibles de tests, visant à exhiber chacun des comportements présents dans le code OCL. Un moteur de génération de tests calcule ensuite automatiquement une séquence d'opérations, qui, depuis l'état initial, permettra d'activer le comportement ciblé. Ces séquences constituent alors le référentiel de tests abstraits. Celui-ci pourra être concrétisés dans différents formats, dépendant du banc de test adressé, nécessitant le développement d'un adaptateur liant les données abstraites et les données concrètes. Ce processus MBT classique, dénomé également *off-line*, est implanté au sein de l'outil Smartesting CertifyIt [2].

Dans le cadre du projet MBT_Sec, les améliorations de ce processus sont de deux axes. D'une part, nous souhaitons améliorer la couverture de besoins de tests spécifiques, en utilisant des propriétés permettant de décrire les aspects de sécurité fonctionnelle du système. Ces propriétés peuvent être utilisées pour qualifier une base de tests, en évaluant si les tests existants couvrent la propriété, et/ou pour alimenter cette base de tests, en produisant des cibles pertinentes par rapport à la propriété considérée. D'autre part, nous nous sommes intéressés à la définition et à l'utilisation d'une technologie de génération de tests à la volée et en ligne, permettant de réaliser de larges explorations du modèle, couplant la génération et l'exécution des tests. Nous décrivons ces deux approches respectivement dans Section 2.1 et Section 2.2.

2.1 Test à partir de propriétés

Les propriétés utilisées dans notre approche sont basées sur les patrons de propriété de Dwyer [5]. Ceux-ci spécifiquent qu'une propriété est décrite par une combinaison de deux éléments : un motif (pattern) et une portée (scope) qui constitue une fenêtre d'observation de l'exécution du système, durant laquelle le motif, exprimant des occurrences ou des absences d'événements, doit être valide. Le langage TOCL [6], initialement proposé dans le cadre du projet ANR TASCCC, permet ainsi de décrire des propriétés temporelles à l'aide de ces constructions.

Par exemple, considerons la specification PKCS#11 qui définit l'API Cryptoki, une interface pour la gestion de la securité et l'interoperabilité des composants cryptographiques. La spécification définit differentes propriétés que nous sommes en mesure d'exprimer en TOCL. Ainsi, une des propriétés est définie comme suit : "Un utilisateur ne peut signer un message (opération C_SignInit) sans une authentification prélable (opération C_Login)". Le langage TOCL nous permet d'exprimer 2 propriétés qui formalisent cette exigence de sécurité fonctionnelle :

eventually isCalled(C_Login, @CKR :OK) **before** isCalled(C_SignInit, @CKR :OK)

qui définit qu'une opération de signature qui réussit (comportement CKR :OK) doit être précédée d'un opération de login ayant également été exécutée avec succès. On note ici la présence d'une portée (before - avant la première occurence de l'événement), et d'un motif (eventually - l'événement qui suit doit être présent). Les événements sont ici des appels aux opérations du système sous test, dans lesquelles on précise les comportements (ici, des cas de succès identifiés par @CKR :OK dans le code OCL de ces opérations).

Une seconde propriété vient compléter la précédente, pour exprimer qu'après une perte d'authentification, il y a nécessairement une étape de login pour pouvoir réaliser une signature.

eventually isCalled(C_Login,@CKR :OK)
between isCalled(C_Logout, @CKR :OK) **and** isCalled(C_SignInit, @CKR :OK)

Une fois la propriété décrite, celle-ci est automatiquement convertie en automate qui permet de donner une représentation visuelle des événements ainsi spécifiés, telle que réprésentée sur la Figure 2.

Cet automate sert de base à deux mécanismes complémentaires.

Mesure de couverture des tests. Il est tout d'abord possible de mesurer la couverture de l'automate (et, de ce fait, de la propriété associée) en détectant, pour chaque test, les transitions de l'automate qui sont couvertes. Ce processus, illustré par la Figure 2, est implanté au sein de l'outil CertifyIt. On distingue sur l'automate un état correspondant à un état d'erreur représentant une violation de la propriété. Si un test permet d'atteindre cet état, on se retrouve en présence d'une erreur, qui peut être dûe soit à la propriété (si celle-ci est trop stricte), soit au modèle (si celui-ci est trop laxiste).

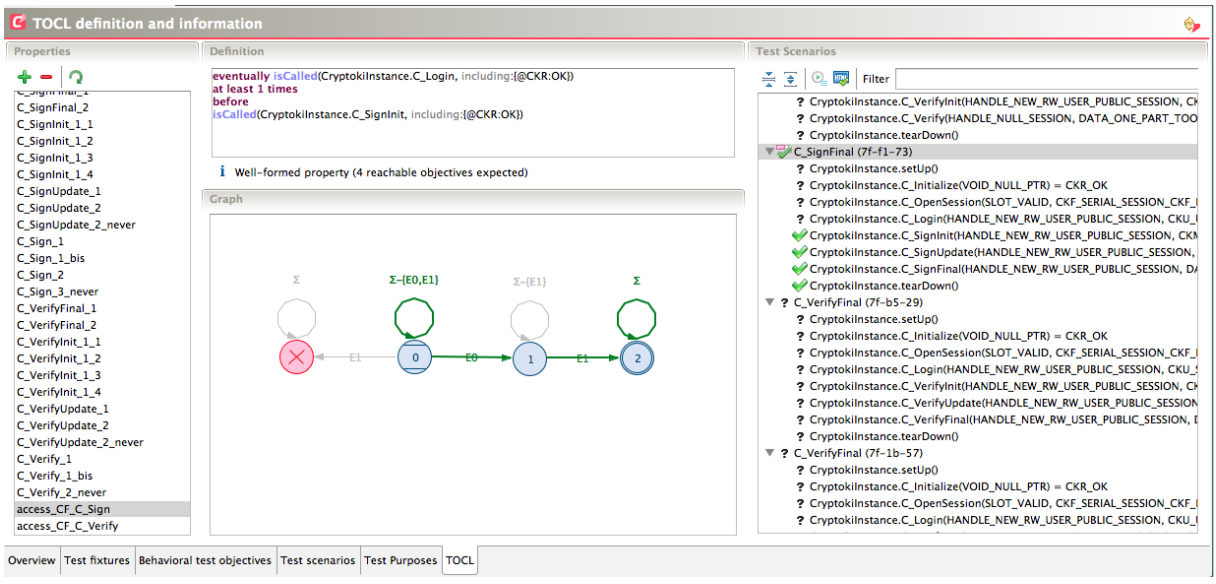

FIGURE 2 – L'interface TOCL de l'outil CertifyIt

Génération de tests complémentaires. Une fois les tests évalués, si la totalité des transitions de l'automate n'a pas été couverte, il est possible de générer des cas de tests supplémentaires dans CertifyIt. Ceci permet de venir compléter la base de tests pour exercer la propriété dans des configurations du système jusqu'alors ignorées.

2.2 Test on-line

MBeeTle est un outil de génération de tests *on-line*. La dénomination "on-line" signifie que l'outil génère et exécute les tests sur le système sous test en lui étant directement connecté. Cet outil se base sur la technologie d'animation de modèle extraite de l'outil CertifyIt. Il réutilise par ailleurs les éléments de la démarche off-line avec l'outil CertifyIt : il réutilise le modèle, la couche d'adaptation et les tests déjà généré par CertifyIt.

Le calcul des tests est similaire de l'idée du fuzzing, le moteur de MBeeTle va chercher à activer aléatoirement les opérations offertes par le système, ayant pour but de stimuler un comportement inhabituel et ainsi découvrir des vulnerabilités dans le système. Chaque pas calculé est ainsi envoyé au banc de test qui renvoie le résultat de l'exécution. La génération d'un pas de test consiste à sélectionner aléatoirement une opération, puis de valider le pas par un sélecteur. Le sélecteur valide le pas selon plusieurs stratégies : *all-steps* (chaque pas est valide), *tag-based* (en fonction de la couverture des comportements), *flower* (se base sur la reconnaissance des cas (non)-passants) ou en combinant ces stratégies unitaires dans une stratégie dite complexe.

3 Application aux composants de sécurité

Dans le contexte du partenariat avec la DGA, nous nous intéressons à des composants cryptographiques. A des fins de démonstration, nous avons expérimenté notre approche sur le standard PKCS#11 (Public Key Cryptographic Standards) [4]. Celui-ci est un standard visant la sécurité et interopérabilité des composants. PKCS#11 défini l'API Cryptoki, une interface de composants de sécurité.

Les travaux et les premières experimentations nous ont permis de tirer les conclusions suivantes.

Concernant l'utilisation des propriétés TOCL : (*i*) les propriétés TOCL permettent d'exprimer des exigences de sécurité présentes dans le document de spécification intial sans rajouter d'information superflue dans le modèle de test, (*ii*) les tests fonctionnels initialement calculés par CertifyIt ne couvrent que très partiellement ces propriétés, Les propriétés TOCL ne sont couvertes que partiellement par les tests fonctionnels, dont environ 35% de propriétés TOCL non couverts par ces tests (i.e., la situation décrite dans la propriété n'est pas exercée par les tests), (*iii*) en se basant sur les automates, CertifyIt permet de calculer en temps restreint les cibles de test, et ainsi de couvrir les propriétés TOCL à moindre effort.

Concernant l'utilisation de l'outil MBeeTle de génération de tests on-line : (*i*) la mise en oeuvre de cette approche présente un coût minimal, du fait de la réutilisation de composants de concrétisation de tests déjà existants, (*ii*) cet outil permet de couvrir de larges pans du système et d'activer des conditions limites liées à l'infrastructure qui n'étaient pas prises en compte par l'approche fonctionnelle, (*iii*) les traces produites par MBeeTle tendent à améliorer la couverture initiale des propriétés TOCL définies sur le modèle.

Finallement, nous avons constatés que les approches on-line et off-line, test fonctionnel classique, sont complémentaires. L'approache on-line peut être employée en tâche de fond pour éprouver le système de manière intensive, une fois que les premières campagnes de qualification du système sont effectuées.

4. http ://france.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-11-cryptographic-token-interface-standard.htm

4 Conclusion et travaux actuels

Le projet MBT_Sec propose d'industrialiser une technologie mature, initialement définie et expérimentée dans le cadre de projets ANR antérieurs : le projet ANR TASCCC (2009–2012) qui a permis la définition du langage de propriétés temporelles TOCL, et le projet OSeP (2012–2013) qui a montré l'intérêt de ces techniques pour le test de sécurité fonctionnelle des composants cryptographiques. Cette technique s'avère intéressante car elle répond à des besoins spécifiques de nos partenaires de la DGA dans le cadre de la validation de composants cryptographiques. Par ailleurs, l'approche proposée se veut plus générale et ne se limite pas à ce type d'applications. En effet, les expérimentations menées dans le cadre du projet TASCCC ont également montré l'intérêt de cette approche pour la validation de systèmes embarqués, tels que les cartes à puces. De plus, la couverture des propriétés permet de donner des garanties concernant les tests produits qui correspondent aux attentes des évaluateurs Critères Communs.

Nos efforts se focalisent actuellement sur l'implantation de critères de test de robustesse visant à exercer le système vis-à-vis en allant provoquer les événements indésirables [4]. Par ailleurs, même si les premiers résultats sont concluants en ce sens, nous souhaitons évaluer le pouvoir de détection d'erreur des deux approches proposées dans ce projet.

Références

[1] Boris Beizer. *Black-box testing - techniques for functional testing of software and systems.* Wiley, 1995.

[2] F. Bouquet, C. Grandpierre, B. Legeard, and F. Peureux. A test generation solution to automate software testing. In *Proceedings of the 3rd International Workshop on Automation of Software Test*, AST '08, pages 45–48, New York, NY, USA, 2008. ACM.

[3] F. Bouquet, C. Grandpierre, B. Legeard, F. Peureux, N. Vacelet, and M. Utting. A subset of precise UML for model-based testing. In *Proceedings of the 3rd International Workshop on Advances in Model Based Testing*, pages 95–104, 2007.

[4] F. Dadeau, K. Cabrera Castillos, and J. Julliand. Coverage criteria for model-based testing using property patterns. In *9th Workshop on Model-Based Testing, Satellite workshop of ETAPS 2014*, volume 141 of *EPTCS*, pages 29–43, Grenoble, France, April 2014. Open Publishing Association.

[5] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 411–420, New York, NY, USA, 1999. ACM.

[6] B. Kanso and S. Taha. Temporal constraint support for ocl. In Krzysztof Czarnecki and Gorel Hedin, editors, *Software Language Engineering*, volume 7745 of *LNCS*, pages 83–103. Springer Berlin Heidelberg, 2013.