Modélisation et Évaluation de la Sécurité des IHM

Youssou Ndiaye[1], Nicolas Aillery[1], Olivier Barais[2], Arnaud Blouin[2], and Ahmed Bouabdalah[3]

[1] Orange
[2] Inria/IRISA
[3] IMT-Atlantique
prénom.nom@[orange.com | irisa.fr | imt-atlantique.fr]

Résumé

Les Interfaces Homme-Machine (IHM) font parties intégrantes des systèmes d'information et permettent aux utilisateurs d'interagir avec le système sous-jacent. Ces IHM permettent de manipuler des informations sensibles, telles que des données personnelles de l'utilisateur. Des erreurs de fonctionnement de l'IHM ou un manque de sécurité peuvent conduire à des failles de sécurité. Les techniques de validation et de vérification logicielles peuvent aider à trouver des erreurs dans le fonctionnement des IHM. Ces techniques, cependant, font face à plusieurs limites. Dans cette thèse CIFRE, intitulée "modélisation et évaluation de la sécurité des IHM", nous visons à répondre à ces limites en dotant les ingénieurs logiciels de nouvelles méthodes de validation et de vérification visant à améliorer la sécurité des IHM.

Résumé

Human-Computer Interactions(HCI) are part of the Information System and allow user to interact with the system. HCI allow to manipulate sensitive data such as user's credentials. A lack of security or a dysfonctionnement of such HCI can lead to security flaws. Software testing technics can help to find problems on HCI, but they are facing several limits. This thesis entitled "Modeling and Assessing the HCI Security" aims at providing new methods and tools to the software engineers in order to improve the security of HCI.

Mots Clés : IHM, Sécurité, Validation et Vérification.

Keywords : HCI, Security, Validation and Verification.

1 Contexte

Pour interagir avec les systèmes d'information, les utilisateurs utilisent des Interfaces Homme-Machine (IHM). Comme tout composant logiciel, les IHM doivent être testées pour assurer leur bon fonctionnement. Les IHM sont d'autant plus importantes qu'elles permettent de manipuler des données sensibles telles que les données personnelles des utilisateurs. Des problèmes de sécurité ou la mauvaise conception d'une IHM peuvent faire apparaître des failles de sécurité. Les techniques de validation et de vérification logicielle peuvent aider à trouver des erreurs dans le fonctionnement des IHM. Ces techniques, cependant, font face à deux limites majeures. Premièrement, l'usage de modèles de fautes dédiés permet de trouver des erreurs de fonctionnement sur un système donné. Cependant, les modèles de fautes d'IHM actuels ne couvrent pas les erreurs de sécurité. Inversement, la majeure partie des modèles de fautes de sécurité ne considère pas l'IHM. Un premier verrou scientifique à lever est donc l'identification et la caractérisation de fautes d'IHM liées à la sécurité. Deuxièmement, le nombre d'actions qu'un utilisateur peut réaliser sur une IHM provoque une explosion combinatoire du nombre de tests possibles et mène à des problèmes de passage à l'échelle des outils de vérification logicielle. Cela concerne particulièrement la longueur des séquences d'actions, appelées parcours utilisateur. En 2012, une faille de sécurité a été identifiée dans le logiciel Skype permettant à un utilisateur de prendre le contrôle d'un autre compte [1]. Cette faille, rapidement corrigée, demandait la réalisation de 7 actions dans un ordre spécifique sur l'IHM de Skype pour exploiter un mauvais fonctionnement de cette IHM. De manière isolé, ces 7 actions fonctionnaient correctement. Leur assemblage, cependant, a permis d'exploiter une faille de sécurité de l'IHM. Un second verrou scientifique à lever est donc comment : vérifier ou valider en un temps raisonnable l'absence de failles de sécurité dans le code d'IHM malgré des longueurs de parcours utilisateur importantes ; prévenir l'apparition de telles failles dans le code pendant le développement des IHM.

Cette thèse, actuellement en première année, vise à proposer des réponses à ces deux verrous scientifiques pour doter les ingénieurs logiciels d'outils leur permettant de modéliser, d'évaluer et d'analyser la sécurité des parcours utilisateur d'une IHM.

Le reste de l'article se compose de la manière suivante. La section 2 introduit les concepts importants relatifs à cette thèse. La section 3 discute des travaux connexes. La section 4 détaille les axes de recherche visant à répondre aux deux verrous scientifiques identifiés.

2 Définitions

Nous rappelons dans un premier temps les définitions des notions abordées dans cet article.

Interface Homme-Machine : L'IHM est la discipline qui étudie le design, l'évaluation et l'implémentation des interfaces et interactions utilisateur. Cette définition considère les aspects d'utilisabilité et de design. Cette thèse ne traite pas de ces aspects pour se focaliser sur l'aspect génie logiciel des IHM, en l'occurrence la conception, l'implémentation, la vérification et validation logicielle de la sécurité des IHM.

Parcours utilisateur : Un parcours utilisateur est une suite ordonnée d'actions qu'un utilisateur peut réaliser sur une IHM. L'ensemble des parcours utilisateur forme le graphe de flot d'événements [16] (*event-flow graph*) utilisé en test d'IHM pour produire des tests. Les conditions sur l'accomplissement de chaque action dépendent de l'état actuel du système et de l'IHM.

1. https://www.reddit.com/r/netsec/comments/13664q/skype_vulnerability_allowing_hijacking_of_any/

Par exemple, la soumission d'un article en relecture sur un site dédié tel qu'Easychair constitue un parcours utilisateur dont certaines actions peuvent être soumises à des exigences de sécurité.

Exigence de sécurité : Selon Hope *et al.* [3] et adapté au parcours utilisateur, une exigence de sécurité est l'ensemble des conditions et capacités permettant de limiter ou de contrôler l'accomplissement des actions d'un parcours utilisateur. Dans l'exemple de la soumission d'un article sur Easychair introduit dans la définition précédente, le fait que l'utilisateur doit s'identifier afin de soumettre un article constitue une exigence de sécurité. Les moyens mis en œuvre pour garantir ces exigences de sécurité constituent les mécanismes de sécurité [2] (*e.g.,* Usage d'un login et d'un mot de passe pour identifier l'utilisateur sur Easychair).

3 Travaux connexes

Modélisation et sécurité des IHM. Des langages de modélisation ont été proposés pour modéliser la sécurité d'un système d'information [14, 8]. Des travaux de recherche ont également été conduits sur la validation des politiques de contrôle d'accès [17, 4, 6]. De nombreux outils et méthodes d'analyse statique ou dynamique de détection de failles, de fautes et de vulnérabilités ont été proposés pour améliorer la sécurité des systèmes [19, 7, 13]. Il n'existe cependant pas de langages permettant de modéliser à la fois une IHM, la sécurité de cette IHM et de valider la sécurité de celle-ci. Un tel langage est cependant nécessaire pour réaliser la validation et la vérification de la sécurité d'IHM.

Validation et vérification d'IHM. Le test d'IHM est une activité chronophage et sujette à des erreurs si elle est réalisée manuellement [9]. Des approches proposent d'extraire un modèle décrivant les différentes actions utilisateurs et leurs séquences possibles à partir de code existant [16] ou de spécifications [12]. Des tests d'IHM, *i.e.,* des parcours utilisateurs associés à des vérifications, peuvent ensuite être produits selon des critères de couvertures. Cependant, des études ont montré que ces approches pouvaient difficilement produire et exécuter de manière exhaustive des tests de longueurs importantes [15]. Différentes techniques ont été proposées pour réduire la quantité de test [1], ce qui peut constituer une piste de recherche pour les verrous de cette thèse. Concernant la vérification d'IHM, des approches ont été proposées pour vérifier le respect de certaines propriétés des IHM [18]. Ces approches ne considèrent pas le parcours utilisateur et les problèmes de sécurité.

Toutes ces approches se fondent sur des modèles de fautes d'IHM décrivant les erreurs que les tests et analyses doivent détecter. Ces modèles de fautes ne couvrent cependant pas la sécurité des IHM relative au parcours utilisateur [10, 11, 5].

4 Axes de recherche

Nous détaillons dans cette section les différents axes de recherche que nous envisageons de suivre pour lever les deux verrous scientifiques détaillés dans la section 1. Les deux axes de cette thèse sont : 1/ L'identification et la caractérisation de fautes d'IHM liées à la sécurité et au parcours utilisateur ; 2/ sur la base de ces fautes, la modélisation et l'évaluation de la sécurité des parcours utilisateur.

4.1 Caractériser les failles IHM

Nous considérons dans cette thèse une erreur d'IHM comme pouvant être un comportement inattendu de l'IHM ou une faille de sécurité pouvant être exploitée pour provoquer un comportement inattendu.

Les techniques de test logiciel reposent sur des modèles de fautes décrivant les erreurs à trouver. Un modèle de fautes est une abstraction décrivant des erreurs pouvant affecter un système. Il existe deux catégories de fautes d'IHM que nous allons considérer dans nos travaux : 1/ les fautes liées à des erreurs classiques de développement logiciel (par ex. erreurs liées à l'implémentation du code de l'IHM) ; 2/ les mauvaises pratiques de développement logiciel consistant à déléguer certaines validations d'exigences de sécurité sur le côté client d'une application. Le cas des applications Web avec l'usage de JavaScript côté client constitue un exemple illustratif de cette dernière catégorie (*e.g.,* vérification du mot de passe de l'utilisateur avec du code JavaScript coté client d'une application web).
Les moyens d'action d'un attaquant sont diverses, compliquant ainsi la définition du modèle de faute. Un attaquant peut par exemple exploiter un parcours utilisateur défaillant, comme dans l'exemple de Skype, sans modifier le code de l'IHM. Il peut également altérer le code de l'IHM, par exemple le code JavaScript côté client d'une application Web, pour modifier un parcours utilisateur. Les travaux de cette thèse visent à proposer une caractérisation précise des ces deux catégories de fautes d'IHM.

Pour réaliser cette étape, nous envisageons d'étudier les rapports de bug et de sécurité de projets open-source ou d'Orange pour identifier empiriquement les problèmes de sécurité liés aux IHM reportées.

4.2 Modéliser et évaluer la sécurité d'un parcours utilisateur

Une des limites actuelles des techniques de test d'IHM est l'incapacité à produire et exécuter des tests de grandes longueurs. De plus, les techniques de vérification d'IHM se focalisent sur d'autres fautes que celles liées à la sécurité des IHM et des parcours utilisateur. Pour lever ce verrou, nous envisageons différentes pistes. D'abord, en utilisant le modèle de faute que nous aurons défini, nous allons chercher à prouver formellement ou a valider l'existence ou non de failles sur les parcours utilisateur capturés dans un modèle (*i.e.,* accomplissement d'une action sans respect des exigences de sécurité). Cette piste permettrait d'évaluer du code ou des modèles d'IHM existants. Pour réduire le nombre de parcours utilisateur à analyser, nous allons privilégier dans notre approche les sous-parcours dont les actions sont soumises à des exigences de sécurité. Dans le cas de tests d'IHM existantes, des analyses de code statiques ou dynamiques peuvent également permettre de réduire le nombre de parcours utilisateur.

Une autre piste consisterait à fournir aux développeurs des informations au moment du développement ou de la conception d'une IHM concernant le respect des exigences de sécurité. Cela permettrait de lever des alertes sur les étapes d'un parcours susceptibles de rendre ce dernier vulnérable. A la manière des LoA (Level of Assurance) de l'ISO 29115 dédié à l'authentification, nous allons définir des critères d'évaluation de la sécurité d'un parcours. Ces critères nous permettrons par exemple d'évaluer le niveau de sécurité d'un parcours de récupération d'un mot de passe.

Ces deux pistes requièrent un langage permettant de modéliser à la fois les parcours utilisateur d'une IHM ainsi que ses exigences de sécurité. L'utilisation ou la modification de langages existants (*e.g.,* Malai [12]) sera privilégié.

Références

[1] S. Arlt, A. Podelski, C. Bertolini, M. Schäf, I. Banerjee, and A. M. Memon. Lightweight static analysis for GUI testing. In *2012 IEEE 23rd International Symposium on Software Reliability Engineering*, pages 301–310, 2012.

[2] Matt Bishop. What is computer security ? *IEEE S&P*, 99(1) :67–69, 2003.

[3] Paco Hope and Peter White. Software security requirement the foundation for security. *Cigital Inc., Available : http ://sqgne. org/presentations/2007-08/Hope-Sep-2007. pdf.*

[4] Vincent C Hu and D Richard Kuhn. General methods for access control policy verification (application paper). In *Information Reuse and Integration (IRI), 2016 IEEE 17th International Conference on*, pages 315–323. IEEE, 2016.

[5] Vinay M Igure and Ronald D Williams. Taxonomies of attacks and vulnerabilities in computer systems. *IEEE Communications Surveys & Tutorials*, 10(1), 2008.

[6] Somesh Jha, Ninghui Li, Mahesh Tripunitara, Qihua Wang, and William Winsborough. Towards formal verification of role-based access control policies. *IEEE Transactions on Dependable and Secure Computing*, 5(4) :242–255, 2008.

[7] Nenad Jovanovic, Christopher Kruegel, and Engin Kirda. Pixy : A static analysis tool for detecting web application vulnerabilities. In *Security and Privacy, 2006 IEEE Symposium on*, pages 6–pp. IEEE, 2006.

[8] Jan Jürjens. Umlsec : Extending uml for secure systems development. In *International Conference on The Unified Modeling Language*, pages 412–425. Springer, 2002.

[9] David J. Kasik and Harry G. George. Toward automatic generation of novice user test scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI'96, pages 244–251, 1996.

[10] Carl E Landwehr, Alan R Bull, John P McDermott, and William S Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys (CSUR)*, 26(3) :211–254, 1994.

[11] Valéria Lelli, Arnaud Blouin, and Benoit Baudry. Classifying and qualifying gui defects. In *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on*, pages 1–10. IEEE, 2015.

[12] Valéria Lelli, Arnaud Blouin, Benoit Baudry, and Fabien Coulon. On model-based testing advanced guis. In *Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on*, pages 1–10. IEEE, 2015.

[13] Xiaowei Li and Yuan Xue. Block : a black-box approach for detection of state violation attacks towards web applications. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 247–256. ACM, 2011.

[14] Torsten Lodderstedt, David Basin, and Jürgen Doser. Secureuml : A uml-based modeling language for model-driven security. In *International Conference on the Unified Modeling Language*, pages 426–441. Springer, 2002.

[15] A. M. Memon and Qing Xie. Empirical evaluation of the fault-detection effectiveness of smoke regression test cases for gui-based software. In *20th IEEE International Conference on Software Maintenance, 2004. Proceedings.*, pages 8–17, 2004.

[16] Atif M Memon. An event-flow model of GUI-based applications for testing. *Software Testing, Verification and Reliability*, 17(3) :137–157, sep 2007.

[17] Tejeddine Mouelhi. *Testing and Modeling Security Mechanisms in Web Applications.* PhD thesis, Institut National des Télécommunications, 2010.

[18] Raquel Araújo de Oliveira. *Formal specification and verification of interactive systems with plasticity : applications to nuclear-plant supervision.* PhD thesis, Grenoble Alpes, 2015.

[19] Filippo Ricca and Paolo Tonella. Analysis and testing of web applications. In *Proceedings of the 23rd international conference on Software engineering*, pages 25–34. IEEE Computer Society, 2001.