Tests basés sur des modèles et méthodes formelles pour la sécurité des services Web

Moez Krichen

Tests basés sur des modèles et méthodes formelles pour la sécurité des services Web

Moez Krichen

Laboratoire ReDCAD, Université de Sfax, Tunisie
moez.krichen@redcad.org

Résumé. Les applications Web (AW) sont en constante évolution et déployées à grande échelle. Cependant, ils sont exposés à diverses attaques. Le plus grand défi auquel sont confrontées les organisations est de savoir comment développer une AW qui répond à leurs exigences en matière d'échange de données sensibles, de commerce électronique et de flux de travail sécurisés. Dans cet article, nous donnons un aperçu sur les principales techniques de vérification dédiées aux AW, à savoir les tests basés sur des modèles et les méthodes formelles.

1 Introduction

Au cours de la dernière période, les organisations ont utilisé le Web non seulement comme un outil pour annoncer leurs images, produits et services, mais également pour effectuer leurs tâches quotidiennes, y compris les données sensibles et les flux de travail complexes. De plus, en raison de la popularité et de la diffusion des appareils portables sophistiqués, plusieurs applications passent des versions de bureau classiques aux versions Web pour cibler davantage d'appareils à faible coût de portabilité (25). D'autre part, le nombre d'attaquants ne cesse de croître, et leurs techniques d'attaque deviennent de plus en plus sophistiquées et dangereuses, ce qui impose de réels défis de sécurité aux organisations pour sécuriser leurs applications web (AW). Ainsi, la sécurité des AW est devenue un domaine de recherche important et plusieurs solutions ont été proposées pour protéger les AW. Dans cet article, nous donnons un aperçu les principales techniques de vérification dédiées aux AW, à savoir les tests basés sur des modèles (38; 40; 32; 39) et les méthodes formelles (51; 23; 59).

2 Préliminaires

Test basé sur un modèle (TBM) Le test basé sur un modèle (27; 33) est une méthodologie où le comportement du système sous test (SST) est codé au moyen d'un modèle abstrait. Cette méthodologie permet d'extraire automatiquement des scénarios de test abstraits du modèle considéré. Ces séquences de stimuli sont exécutées sur le SST. Le verdict correspondant est donné en comparant les sorties obtenues du SST avec les sorties produites par le modèle (43; 56). En ce qui concerne le test des aspects

de sécurité, les auteurs de (29) ont proposé une approche basée sur un modèle pour tester les aspects de sécurité de l'Internet des objets (IoT) dans les villes intelligentes. L'approche proposée est basée sur Price Timed Automata et Attack Trees et tire parti de l'adoption du langage de test standard TTCN- 3 (35) et d'une architecture orientée cloud (36) qui est en charge de exécution de cas de test et collecte des verdicts obtenus. De même, les auteurs de (30; 31) ont proposé une méthodologie basée sur un modèle pour tester les propriétés de sécurité de l'IoT. Le formalisme adopté était basé sur le modèle des " Automates temporisés étendus avec entrées et sorties ". Dans (28), un ensemble de techniques d'optimisation a été adopté afin de diminuer la complexité des procédures TBM.

Méthodes formelles Lors de la mise en place de systèmes informatiques (CS), la détection et la correction complètes des erreurs de conception restent remarquablement difficiles dans le contexte de simples procédures de vérification manuelle et d'activités de test fonctionnel. Par conséquent, au début des années 80, les scientifiques ont commencé à rendre les méthodologies de vérification CS plus rigoureuses, spécialement en les rendant plus automatiques (15; 48). En fait, avec l'émergence de nouveaux langages mathématiques pour la spécification et la description de systèmes dynamiques (34), les premières méthodologies de vérification formelle sont apparues. Nous pouvons différencier deux classes principales de procédures informatisées de vérification formelle à savoir la vérification de modèle et le théorème automatisé prouvant. D'une part, l'approche de vérification de modèle, introduite indépendamment deux travaux différents (15) en 1981 et (48) en 1982, automatise la vérification d'une spécification formelle par rapport à une propriété donnée en parcourant tout l'espace d'états du système considéré. D'autre part, la démonstration automatisée de théorèmes consiste à démontrer des théorèmes à l'aide d'un ensemble d'axiomes et de règles d'inférence. Les auteurs de (12) ont proposé une bonne enquête sur les travaux existants dans le domaine des techniques de méthodes formelles dans le domaine de la sécurité web.

3 Test basé sur un modèle pour la sécurité Web

Modélisation des requêtes HTTP Dans (13), Calvi et Viganò ont proposé une approche nommée Chained Attack qui considère les requêtes HTTP en entrée, produit un modèle et extrait scénarios d'attaques du modèle à l'aide de procédures de vérification du modèle. Une approche similaire a été adoptée par les auteurs de (3; 4).

Formalisation des vulnérabilités en objectifs de test Les auteurs de (37; 57) ont proposé une approche de test de sécurité basée sur un modèle qui permet de formaliser des modèles de test de vulnérabilité sous forme de tests. Les auteurs ont défini à la fois le comportement des applications Web et des objectifs de test considérés, et ont adopté des procédures de vérification des modèles pour générer des scénarios de test abstraits.

Examen d'un modèle d'attaquant Les auteurs de (6) ont adopté une méthodologie TBM dans laquelle les modèles formels d'attaquant sont considérés pour valider les

applications et fonctionnalités Web. Une autre méthodologie formelle qui a introduit un modèle d'attaquant distinct a été introduite par les auteurs de (50), où ils ont adopté le langage formel ASLan++ (58).

Technique inspirée des tests de mutation Dans (9; 10; 11), les auteurs ont présenté une méthodologie basée sur des modèles pour tester les propriétés de sécurité des applications Web. L'approche proposée est étroitement inspirée de la mutation techniques de test. En effet, les auteurs ont commencé avec un modèle formel sécurisé et ont utilisé des opérateurs de mutation pour insérer systématiquement des vulnérabilités dans le modèle considéré.

Utilisation de l'outil UMLsec Dans (26), les auteurs ont proposé une approche de test basée sur un modèle pour la génération automatique de scénarios de test de sécurité. L'approche tire parti de l'outil UMLsec. Il vise à tester les propriétés de sécurité des "Common Electronic Purse Specifications". De la même manière, les auteurs de (20) ont présenté une méthodologie de vérification des aspects de sécurité spécifiques aux cartes à puce qui combine les techniques TBM et les procédures de vérification UMLsec.

Utilisation d'Alloy Analyzer Dans (1), une méthodologie utilisant l'Alloy Analyzer pour inspecter plusieurs applications Web et des mécanismes a été proposé. Les auteurs ont adopté des modèles de menace tels qu'un intrus prenant le contrôle d'un site Web ou d'une partie entière du réseau.

Mobster Tool Dans (45; 41), les auteurs ont présenté l'outil MobSTer qui est un cadre de test de sécurité basé sur un modèle qui peut aider un analyste de sécurité à tester la sécurité aspects des applications Web. Ce cadre combine les procédures de vérification des modèles avec les connaissances acquises des lignes directrices et des listes de contrôle pour les tests d'intrusion.

4 Méthodes formelles pour la sécurité Web

Modélisation, vérification et application D'autres travaux de recherche adoptent une autre stratégie qui consiste à envisager des algorithmes et des modèles appropriés afin de traiter la vérification des propriétés de sécurité des technologies Web modernes. Ces travaux de recherche tentent d'exploiter au mieux les normes et cadres disponibles. Cette approche peut être dans de nombreux cas sous-optimale et peu efficace. Cependant, le principal avantage est que cette procédure n'impacte pas beaucoup les technologies Web existantes. Par exemple en ce qui concerne les langages de script, différentes solutions (55; 46) basées sur l'adoption d'une sémantique rigoureuse pour le langage considéré ont été adoptées. De même et à un niveau plus élevé, d'autres projets de recherche ont réussi à envisager des modèles adéquats pour les navigateurs Web, soit complètement (7; 60), soit partiellement (2; 49). Dans (53), les auteurs ont proposé une nouvelle syntaxe dans la logique d'alliage et une toile modèle de sécurité,

y compris les caches. En ce qui concerne la partie serveur, des langages de programmation spécifiques possédant une sémantique formelle ont également été adoptés (par exemple, Python (47) et PHP (19)).

Sécurité par construction Certains ouvrages de la littérature visent à définir de nouveaux langages formels et techniques d'abstraction afin de rendre le Web plus sûr. Pour cela, ces travaux tentent d'identifier les principales limites des techniques actuelles de conception du Web et suggèrent une évolution de paradigme pour l'améliorer. Ces suggestions sont adéquates pour résoudre la source racine des problèmes de sécurité, mais elles nécessitent généralement des changements profonds dans les applications et technologies Web actuelles. Par exemple, en ce qui concerne les langages de script, il peut exister plusieurs incohérences entre les différentes versions et implémentations du navigateur. Pour cela, de nombreux travaux de recherche suggèrent l'adoption de langages de programmation plus sophistiqués basés sur des bases formelles solides pour établir la partie client des applications web (21; 54; 5). En outre, il existe d'autres travaux qui explorent de nouvelles conceptions de navigateur intéressantes sur lesquelles la vérification formelle des propriétés de sécurité peut être directement appliquée (52; 22). Dans le même sens, les auteurs de (24) proposent la synthèse d'un navigateur Web à partir d'un noyau vérifié. À un niveau de complexité plus élevé, d'autres travaux de recherche ont proposé une variété de langages de programmation multi-tiers (18; 8; 14; 42) qui permettent de gérer la complexité d'intégrer des technologies distinctes en fournissant une couche d'abstraction unifiée.

Une application industrielle Dans (16; 44), les auteurs souhaitaient appliquer une vérification formelle pour la sécurité des "Amazon Web Services" (AWS). Deux objectifs principaux ont été envisagés à cet égard. Le premier consiste en élever le niveau de sécurité des produits fournis et le second aider les clients à sécuriser eux-mêmes contre d'éventuelles attaques. Dans ce contexte, les auteurs ont travaillé sur l'adaptation de plusieurs projets open-source. Par exemple, ils ont adapté CBMC [1] afin de faciliter sa utiliser pour le cas des centres de données Amazon (17).

Références

[1] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song. Towards a formal foundation of web security. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 290–304, July 2010.

[2] Ana Almeida-Matos, José Fragoso Santos, and Tamara Rezk. An information flow monitor for a core ofÂ dom. In Matteo Maffei and Emilio Tuosto, editors, *Trustworthy Global Computing*, pages 1–16, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[3] A. Armando, R. Carbone, L. Compagna, K. Li, and G. Pellegrino. Model-checking driven security testing of web-based applications. In *2010 Third International*

1. https ://github.com/diffblue/cbmc

Conference on Software Testing, Verification, and Validation Workshops, pages 361–370, April 2010.

[4] Alessandro Armando, Giancarlo Pellegrino, Roberto Carbone, Alessio Merlo, and Davide Balzarotti. From model-checking to automated testing of security protocols : Bridging the gap. In Achim D. Brucker and Jacques Julliand, editors, *Tests and Proofs*, pages 3–18, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, and Sergio Maffeis. Language-based defenses against untrusted browser origins. In *USENIX Security Symposium*, 2013.

[6] A. Blome, M. Ochoa, K. Li, M. Peroli, and M. T. Dashti. Vera : A flexible model-based vulnerability testing tool. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, pages 471–478, March 2013.

[7] Aaron Bohannon and Benjamin C. Pierce. Featherweight firefox : Formalizing the core of a web browser. In *Proceedings of the 2010 USENIX Conference on Web Application Development*, WebApps'10, pages 11–11, Berkeley, CA, USA, 2010. USENIX Association.

[8] Gérard Boudol, Zhengqin Luo, Tamara Rezk, and Manuel Serrano. Reasoning about web applications : An operational semantics for hop. *ACM Trans. Program. Lang. Syst.*, 34 :10 :1–10 :40, 2012.

[9] Matthias Büchler. *Semi-Automatic Security Testing of Web Applications with Fault Models and Properties*. PhD thesis, Technical University Munich, 2015.

[10] Matthias Büchler, Johan Oudinet, and Alexander Pretschner. Semi-automatic security testing of web applications from a secure model. In *Sixth International Conference on Software Security and Reliability, SERE 2012, Gaithersburg, Maryland, USA, 20-22 June 2012*, pages 253–262, 2012.

[11] Matthias Büchler, Johan Oudinet, and Alexander Pretschner. Spacite - web application testing engine. In *Fifth IEEE International Conference on Software Testing, Verification and Validation, ICST 2012, Montreal, QC, Canada, April 17-21, 2012*, pages 858–859, 2012.

[12] Michele Bugliesi, Stefano Calzavara, and Riccardo Focardi. Formal methods for web security. *Journal of Logical and Algebraic Methods in Programming*, 87 :110 – 126, 2017.

[13] Alberto Calvi and Luca Viganò. An automated approach for testing the security of web applications against chained attacks. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 2095–2102, New York, NY, USA, 2016. ACM.

[14] Adam Chlipala. Ur/web : A simple model for programming the web. *Commun. ACM*, 59 :93–100, 2015.

[15] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, Berlin, Heidelberg, 1982. Springer-Verlag.

[16] Byron Cook. Formal reasoning about the security of amazon web services. In

Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I, pages 38–47, 2018.

[17] Byron Cook, Kareem Khazem, Daniel Kroening, Serdar Tasiran, Michael Tautschnig, and Mark R. Tuttle. Model checking boot code from AWS data centers. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, pages 467–486, 2018.

[18] Ezra Cooper, Sam Lindley, Philip Wadler, and Jeremy Yallop. Links : Web programming without tiers. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem-Paul de Roever, editors, *Formal Methods for Components and Objects*, pages 266–296, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[19] Daniele Filaretti and Sergio Maffeis. An executable formal semantics of php. In *Proceedings of the 28th European Conference on ECOOP 2014 — Object-Oriented Programming - Volume 8586*, pages 567–592, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[20] E. Fourneret, M. Ochoa, F. Bouquet, J. Botella, J. Jurjens, and P. Yousefi. Model-based security verification and testing for smart-cards. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 272–279, Aug 2011.

[21] Cedric Fournet, Nikhil Swamy, Juan Chen, Pierre-Evariste Dagand, Pierre-Yves Strub, and Benjamin Livshits. Fully abstract compilation to javascript. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '13, pages 371–384, New York, NY, USA, 2013. ACM.

[22] Chris Grier, Shuo Tang, and Samuel T. King. Designing and implementing the op and op2 web browsers. *ACM Trans. Web*, 5(2) :11 :1–11 :35, May 2011.

[23] Mehdi Hariati. Formal verification issues for component-based development. *Informatica*, 44(4), 2020.

[24] Dongseok Jang, Zachary Tatlock, and Sorin Lerner. Establishing browser security guarantees through formal shim verification. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 8–8, Berkeley, CA, USA, 2012. USENIX Association.

[25] Ines Jemal, Omar Cheikhrouhou, Habib Hamam, and Adel Mahfoudhi. Sql injection attack detection and prevention techniques using machine learning. *International Journal of Applied Engineering Research*, 15(6) :569–580, 2020.

[26] Jan Jürjens. Model-based security testing using umlsec. *Electron. Notes Theor. Comput. Sci.*, 220(1) :93–104, December 2008.

[27] Moez Krichen. A formal framework for black-box conformance testing of distributed real-time systems. *International Journal of Critical Computer-Based Systems*, 3(1/2) :26–43, 2012.

[28] Moez Krichen. Improving formal verification and testing techniques for internet of things and smart cities. *Mobile Networks and Applications*, pages 1–12, 2019.

[29] Moez Krichen and Roobaea Alroobaea. A new model-based framework for testing security of iot systems in smart cities using attack trees and price timed automata. In *14th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2019*, 2019.

[30] Moez Krichen, Omar Cheikhrouhou, Mariam Lahami, Roobaea Alroobaea, and Afef Jmal Maâlej. Towards a model-based testing framework for the security of internet of things for smart city applications. In *International Conference on Smart Cities, Infrastructure, Technologies and Applications*, pages 360–365. Springer, Cham, 2017.

[31] Moez Krichen, Mariam Lahami, Omar Cheikhrouhou, Roobaea Alroobaea, and Afef Jmal Maâlej. Security testing of internet of things for smart city applications : A formal approach. In *Smart Infrastructure and Applications*, pages 629–653. Springer, Cham, 2020.

[32] Moez Krichen, Afef Jmal Maâlej, and Mariam Lahami. A model-based approach to combine conformance and load tests : an ehealth case study. *International Journal of Critical Computer-Based Systems*, 8(3-4) :282–310, 2018.

[33] Moez Krichen and Stavros Tripakis. Interesting properties of the real-time conformance relation tioco. In *International Colloquium on Theoretical Aspects of Computing*, pages 317–331. Springer, Berlin, Heidelberg, 2006.

[34] Saul A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16(1963) :83–94, 1963.

[35] Mariam Lahami, Fairouz Fakhfakh, Moez Krichen, and Mohamed Jmaiel. Towards a ttcn-3 test system for runtime testing of adaptable and distributed systems. In *IFIP International Conference on Testing Software and Systems*, pages 71–86. Springer, 2012.

[36] Mariam Lahami, Moez Krichen, and Roobaea Alroobaea. Tepaas : test execution platform as-a-service applied in the context of e-health. *International Journal of Autonomous and Adaptive Communications Systems*, 12(3) :264–283, 2019.

[37] F. Lebeau, B. Legeard, F. Peureux, and A. Vernotte. Model-based vulnerability testing for web applications. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, pages 445–452, March 2013.

[38] Afef Jmal Maâlej and Moez Krichen. A model based approach to combine load and functional tests for service oriented architectures. In *VECoS*, pages 123–140, 2016.

[39] Afef Jmal Maâlej, Moez Krichen, and Mohamed Jmaiel. Conformance testing of ws-bpel compositions under various load conditions. In *2012 IEEE 36th Annual Computer Software and Applications Conference*, pages 371–371. IEEE, 2012.

[40] Afef Jmal Maâlej, Mariam Lahami, Moez Krichen, and Mohamed Jmaïel. Distributed and resource-aware load testing of ws-bpel compositions. In *ICEIS (2)*, pages 29–38, 2018.

[41] Federico De Meo and Luca Viganò. A formal approach to exploiting multi-stage

attacks based on file-system vulnerabilities of web applications. In *Engineering Secure Software and Systems - 9th International Symposium, ESSoS 2017, Bonn, Germany, July 3-5, 2017, Proceedings*, pages 196–212, 2017.

[42] Tom Murphy, VII., Karl Crary, and Robert Harper. Type-safe distributed programming with ml5. In *Proceedings of the 3rd Conference on Trustworthy Global Computing*, TGC'07, pages 108–123, Berlin, Heidelberg, 2008. Springer-Verlag.

[43] Arilo Claudio Dias Neto and Guilherme Horta Travassos. A picture from the model-based testing area : Concepts, techniques, and challenges. *Advances in Computers*, 80 :45–120, 2010.

[44] Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff. How amazon web services uses formal methods. *Commun. ACM*, 58(4) :66–73, March 2015.

[45] Michele Peroli, Federico De Meo, Luca Viganò, and Davide Guardini. Mobster : A model-based security testing framework for web applications. *Software Testing, Verification and Reliability*, 28(8) :e1685, 2018. e1685 stvr.1685.

[46] Joe Gibbs Politz, Matthew J. Carroll, Benjamin S. Lerner, Justin Pombrio, and Shriram Krishnamurthi. A tested semantics for getters, setters, and eval in javascript. In *Proceedings of the 8th Symposium on Dynamic Languages*, DLS '12, pages 1–16, New York, NY, USA, 2012. ACM.

[47] Joe Gibbs Politz, Alejandro Martinez, Matthew Milano, Sumner Warren, Daniel Patterson, Junsong Li, Anand Chitipothu, and Shriram Krishnamurthi. Python : The full monty. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*, OOPSLA '13, pages 217–232, New York, NY, USA, 2013. ACM.

[48] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in cesar. In *Proceedings of the 5th Colloquium on International Symposium on Programming*, pages 337–351, London, UK, UK, 1982. Springer-Verlag.

[49] V. Rajani, A. Bichhawat, D. Garg, and C. Hammer. Information flow control for event handling and the dom in web browsers. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 366–379, July 2015.

[50] Marco Rocchetto, Martín Ochoa, and Mohammad Torabi Dashti. Model-based detection of csrf. In Nora Cuppens-Boulahia, Frédéric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans, editors, *ICT Systems Security and Privacy Protection*, pages 30–43, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[51] Markus Roggenbach, Antonio Cerone, Bernd-Holger Schlingloff, Gerardo Schneider, and Siraj Shaikh. Formal methods for software engineering : Languages, methods, application domains. 2020.

[52] Ralf Sasse, Samuel T. King, José Meseguer, and Shuo Tang. IBOS : A correct-by-construction modular browser. In *Formal Aspects of Component Software, 9th International Symposium, FACS 2012, Mountain View, CA, USA, September 12-14, 2012. Revised Selected Papers*, pages 224–241, 2012.

[53] H. Shimamoto, N. Yanai, S. Okamura, J. P. Cruz, S. Ou, and T. Okubo. Towards

further formal foundation of web security : Expression of temporal logic in alloy and its application to a security model with cache. *IEEE Access*, 7 :74941–74960, 2019.

[54] Nikhil Swamy, Cedric Fournet, Aseem Rastogi, Karthikeyan Bhargavan, Juan Chen, Pierre-Yves Strub, and Gavin Bierman. Gradual typing embedded securely in javascript. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, pages 425–437, New York, NY, USA, 2014. ACM.

[55] A. Taly, Ú. Erlingsson, J. C. Mitchell, M. S. Miller, and J. Nagra. Automated analysis of security-critical javascript apis. In *2011 IEEE Symposium on Security and Privacy*, pages 363–378, May 2011.

[56] Mark Utting, Alexander Pretschner, and Bruno Legeard. A taxonomy of model-based testing approaches. *Softw. Test. Verif. Reliab.*, 22(5) :297–312, August 2012.

[57] Alexandre Vernotte, Cornel Botea, Bruno Legeard, Arthur Molnar, and Fabien Peureux. Risk-driven vulnerability testing : Results from ehealth experiments using patterns and model-based approach. In Fredrik Seehusen, Michael Felderer, Jürgen Großmann, and Marc-Florian Wendland, editors, *Risk Assessment and Risk-Driven Testing*, pages 93–109, Cham, 2015. Springer International Publishing.

[58] David von Oheimb and Sebastian Mödersheim. Aslan++ — a formal security specification language for distributed systems. In Bernhard K. Aichernig, Frank S. de Boer, and Marcello M. Bonsangue, editors, *Formal Methods for Components and Objects*, pages 1–22, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[59] Fujun Wang, Zining Cao, Lixing Tan, and Hui Zong. Survey on learning-based formal methods : Taxonomy, applications and possible future directions. *IEEE Access*, 8 :108561–108578, 2020.

[60] Sachiko Yoshihama, Takaaki Tateishi, Naoshi Tabuchi, and Tsutomu Matsumoto. Information-flow-based access control for web browsers. *IEICE Transactions*, 92-D :836–850, 05 2009.