Annuaire du Collège de France

122e année

2021
2022

Résumé des cours et travaux

COLLÈGE
DE FRANCE
1530

SCIENCES DU LOGICIEL

Xavier Leroy
Professeur au Collège de France

La série de cours et de séminaires « Sécurité du logiciel : quel rôle pour les langages de programmation ? » est disponible en audio et en vidéo sur le site internet du Collège de France (https://www.college-de-france.fr/agenda/cours/securite-du-logiciel-quel-role-pour-les-langages-de-programmation et https://www.college-de-france.fr/agenda/seminaire/securite-du-logiciel-quel-role-pour-les-langages-de-programmation), ainsi que le colloque « Probabilistic Programming » (https://www.college-de-france.fr/agenda/colloque/probabilistic-programming).

ENSEIGNEMENT

COURS – SÉCURITÉ DU LOGICIEL : QUEL RÔLE POUR LES LANGAGES DE PROGRAMMATION ?

Comment un logiciel peut-il résister aux attaques et à l'utilisation malveillante ? C'est le problème général de la sécurité du logiciel, et c'est un vaste problème tant les failles de sécurité sont nombreuses et variées, de la manipulation des utilisateurs à l'observation et la perturbation des circuits de l'ordinateur, en passant par l'exploitation astucieuse de « bugs » dans le code du logiciel.

Le cours a abordé cette problématique de sécurité du logiciel sous l'angle des langages de programmation et de leurs techniques de typage, d'analyse statique et de vérification déductive. Cette approche est connue sous le nom de *language-based security* dans la littérature. Le cours s'est efforcé de caractériser la contribution de cette approche à la sécurité informatique, ainsi que ses limitations. Pour ce faire,

X. Leroy, « Sciences du logiciel », *Annuaire du Collège de France. Résumé des cours et travaux*, 122e année : *2021-2022*, 2025, p. 21-29, https://journals.openedition.org/annuaire-cdf/20405.

nous avons décrit un certain nombre de problèmes de sécurité du logiciel ainsi que les protections classiques contre ces problèmes, et avons comparé ces dernières aux garanties que l'on peut obtenir par l'approche *language-based security*.

Cours 1 – Sécurité du logiciel : introduction et études de cas

Le 10 mars 2022

Le premier cours a introduit la problématique de la sécurité du logiciel et montré en quoi elle va au-delà de la problématique de la sûreté de fonctionnement : le logiciel doit résister non seulement aux « bugs » et aux pannes accidentelles (sûreté), mais aussi aux attaques et à l'utilisation malveillante (sécurité). Nous avons ensuite étudié trois attaques récentes et représentatives de la diversité des failles de sécurité du logiciel : l'attaque Heartbleed sur les serveurs web utilisant la bibliothèque OpenSSL, à base d'accès hors bornes dans des tampons mémoire; l'attaque Log4Shell sur la bibliothèque de journalisation Log4j, qui injecte du code arbitraire dans des données contrôlées par l'attaquant; l'attaque sur le *smart contract* DAO de la cryptomonnaie Ethereum, exploitant la vulnérabilité de ce contrat vis-à-vis d'opérations ré-entrantes.

Cours 2 – Flux d'information

Le 17 mars 2022

Certaines informations sont plus confidentielles que d'autres, ou plus dignes de confiance que d'autres. Il faut donc restreindre les flux d'information pour garantir que les informations confidentielles ne sont pas divulguées et que les informations intègres ne sont pas corrompues. Pour ce faire, le contrôle d'accès aux ressources informatiques ne suffit pas, il faut mettre en place des politiques de confidentialité, comme celle de Bell et LaPadula, ou d'intégrité, comme celle de Biba. Nous avons étudié comment contrôler les flux d'information à travers un programme, soit dynamiquement, par des vérifications pendant l'exécution du programme, soit statiquement, à l'aide de systèmes de types ou de logiques de programmes appliquées une fois pour toutes au programme. Nous avons introduit la notion de non-interférence, une caractérisation sémantique de l'absence de flux illégaux, et esquissé comment l'étendre à la déclassification de données confidentielles et à l'approbation de données non fiables.

Cours 3 – Isolation logicielle

Le 24 mars 2022

Les mécanismes d'isolation logicielle ont pour but d'exécuter un logiciel potentiellement malveillant en l'empêchant d'attaquer d'autres logiciels s'exécutant dans le même environnement ou de compromettre les mécanismes de sécurité essentiels du

système d'exploitation et du matériel. Le cours a décrit et comparé plusieurs approches de l'isolation logicielle : mémoire virtuelle, architectures à capacités, isolation logicielle des fautes (par transformation du code machine) et isolation par le langage de programmation et ses interfaces logicielles (API, *Application Programming Interfaces*). Nous avons étudié comment cette dernière approche se réalise dans le langage Java (par une API à base de capacités et par l'inspection de la pile d'appels) et, plus péniblement, dans le langage JavaScript (en utilisant finement la portée statique des liaisons).

Cours 4 – *Tempus fugit* : attaques par observation du temps

Le 31 mars 2022

Le temps que prend un programme ou ses opérations élémentaires à s'exécuter révèle beaucoup de choses sur les données qu'il manipule. En partant de l'exemple de la signature RSA, nous avons décrit comment attaquer un logiciel en observant ses temps d'exécution, et comment contrer ces attaques en modifiant l'algorithme (masquage des données) ou son implémentation (quantification du temps, etc.). La mémoire cache des processeurs fournit un autre canal indirect d'information : le temps que prend un accès mémoire révèle des choses sur les cases mémoires qui ont été récemment accédées, ce qui permet de monter des attaques, comme nous l'avons montré sur l'exemple du chiffrement AES. La programmation dite « en temps constant » (ou plus exactement en temps indépendant des données secrètes) est une des manières de contrer ces attaques par observation du temps et des caches. Nous avons vu comment la caractériser en termes de flux d'information et comment la mettre en pratique. Le cours s'est conclu par un aperçu des attaques de type « Spectre », qui combinent observation du cache et manipulation de l'exécution spéculative du processeur.

Cours 5 – Typage et sécurité

Le 7 avril 2022

Qu'il soit vérifié dynamiquement (pendant l'exécution) ou statiquement (par analyse préalable), le typage est un aspect essentiel des langages de programmation de haut niveau. Le cours a étudié les contributions du typage à la sécurité des logiciels, des garanties de base (sûreté des valeurs et de la mémoire) indispensables pour l'isolation logicielle à des garanties d'intégrité plus fines s'appuyant sur l'abstraction de types et sur le typage statique des ressources et de leur possession (*ownership* et *borrowing*, comme dans le langage Rust). Nous avons ensuite montré comment appliquer le typage statique à du code mobile, par vérification de code intermédiaire (comme le *bytecode* JVM du langage Java) ou de code machine, allant jusqu'à la notion très générale de code auto-certifiant (*proof-carrying code*).

Cours 6 – Compilation et sécurité

Le 14 avril 2022

Compiler un programme source en code machine peut être l'occasion de le rendre plus résistant à certaines attaques. Par exemple, des tests systématiques de bornes lors des accès aux tableaux peuvent être introduits pendant la compilation, puis optimisés pour en réduire le surcoût en temps d'exécution, augmentant ainsi grandement la sécurité à coût raisonnable. Cependant, de nombreuses optimisations de compilation, pourtant sémantiquement correctes, peuvent affaiblir la sécurité du programme. C'est notamment le cas des optimisations qui supposent l'absence de comportements indéfinis dans le programme source. Nous avons montré comment caractériser ces différences de sécurité entre un fragment de programme source et son code compilé à l'aide d'outils sémantiques classiques : l'équivalence observationnelle et le problème de la *full abstraction*. Nous avons présenté quelques approches récemment proposées pour compiler « en toute sécurité », c'est-à-dire en préservant les équivalences observationnelles dans les deux directions.

Cours 7 – Calculer sur des données chiffrées ou privées

Le 21 avril 2022

Le chiffrement est très efficace pour protéger le secret des données au repos (stockage) et en transit (réseaux). Pourrait-il le protéger également pendant les calculs sur ces données ? Le cours a introduit la notion de chiffrement homomorphe, permettant de calculer sur des données chiffrées sans avoir la clé de déchiffrement, et décrit les grandes lignes de l'approche de Gentry (2009), la première réalisation de ce concept. Le chiffrement homomorphe, ainsi que d'autres protocoles cryptographiques, fournit des solutions au problème du calcul multipartite sécurisé, où les participants calculent ensemble une fonction de leurs données privées sans rien révéler de plus que le résultat final. Bien qu'encore très coûteuses en temps d'exécution, ces approches cryptographiques apportent une réponse mathématiquement solide au problème de calculer sur des données confidentielles sans fuites d'information.

SÉMINAIRE (EN RELATION AVEC LE SUJET DU COURS)

Séminaire 1 – Influence de la qualité des spécifications sur la sécurité logicielle

Olivier Levillain (Télécom SudParis), le 17 mars 2022

Les systèmes d'information que nous utilisons quotidiennement sont d'une grande complexité. Ils reposent en particulier sur l'implémentation de protocoles réseau et sur l'interprétation de documents aux formats variés. Ce premier séminaire de l'année a porté sur les spécifications décrivant ces protocoles et ces formats. Le

conférencier a montré que la manière dont ils sont spécifiés peut avoir des conséquences sur la sécurité de leurs implémentations, en étudiant trois exemples : MiniPNG, un format d'images utilisé dans un module d'enseignement en programmation ; le format PDF ; le protocole TLS.

Séminaire 2 – *Differential Privacy: From the central model to the local model and their generalization*

Catuscia Palamidessi (Inria), le 24 mars 2022

La confidentialité différentielle (DP, *Differential Privacy*) est l'une des meilleures approches connues pour protéger des données personnelles tout en permettant d'en extraire des informations statistiques utiles. L'idée centrale est d'ajouter du bruit aléatoire aux données publiées, en quantité soigneusement choisie pour préserver à la fois la confidentialité et l'utilité des données.

La conférencière a introduit le problème de l'anonymisation des données personnelles et motivé l'utilisation d'approches probabilistes. Elle a présenté l'approche classique DP, qui repose sur un traitement centralisé des données, puis une variante plus récente, LDP (*Local Differential Privacy*), où le bruitage des données est réalisé directement par les participants. Ensuite, elle a introduit le modèle *d-privacy*, qui unifie le modèle centralisé et le modèle distribué et s'applique à tout domaine métrique, et a montré comment l'appliquer aux données de géolocalisation.

Séminaire 3 – *Verified implementations for real-world cryptographic protocols*

Karthikeyan Bhargavan (Inria), le 31 mars 2022

La sécurité du Web repose sur des protocoles cryptographiques : des programmes distribués qui utilisent le chiffrement et la signature pour protéger les données sensibles des nombreuses attaques que les adversaires qui contrôlent le réseau peuvent mener. Malgré trente ans d'études, on continue à découvrir des vulnérabilités théoriques ou pratiques dans les protocoles cryptographiques du Web, notamment TLS (*Transport Layer Security*). Le conférencier a décrit plusieurs de ces vulnérabilités et montré comment la vérification formelle permet d'exclure certaines classes d'attaques sur les implémentations de protocoles modernes tels que TLS 1.3 et Signal.

Séminaire 4 – Attaques par injection de fautes et protection logicielle

Karine Heydemann (Sorbonne Université), le 7 avril 2022

Les attaques en faute sont une classe particulière d'attaques qui nécessitent *a priori* un accès physique à la cible matérielle et visent à perturber son environnement d'exécution pour altérer le comportement de l'application qui s'y exécute. Ces attaques sont puissantes : elles permettent de compromettre la sécurité d'algorithmes

cryptographiques prouvés sûrs, de retrouver des informations sensibles, de contourner des protections ou de prendre le contrôle d'un système. La conférencière a d'abord présenté les attaques par injection de fautes : moyens d'injection, effets des fautes, et leur exploitation pour réaliser des attaques. Elle a ensuite présenté les protections logicielles connues contre ces attaques et étudié leurs limites.

Séminaire 5 – Obfuscation du logiciel : brouiller le code pour protéger les programmes

Sandrine Blazy (université de Rennes 1), le 14 avril 2022

L'obfuscation de logiciel vise à brouiller le code en langage machine du logiciel afin qu'il soit difficile à comprendre et à analyser. C'est une technique très utile pour la sécurité par l'obscurité. Ce brouillage peut être effectué à différents niveaux et s'applique à des programmes n'ayant pas été conçus avec des objectifs de sécurité. Aussi de nombreuses stratégies de brouillage ont-elles été proposées dans la littérature. La conférencière a présenté un certain nombre de ces stratégies et étudié la correction sémantique des plus récentes.

Séminaire 6 – *Transient execution attacks and defenses*

Frank Piessens (K.U. Leuven), le 21 avril 2022

Depuis la découverte des attaques sur les exécutions transitoires, nous savons que la micro-architecture des ordinateurs a un impact majeur sur la sécurité du *cloud* et autres ordinateurs qui font tourner des codes provenant de plusieurs acteurs. Le conférencier a décrit ces attaques sur les exécutions transitoires et les protections correspondantes. Pour ce faire, il a utilisé des techniques de modélisation provenant de la recherche en langages de programmation, et montré comment décrire les exécutions *out-of-order* et spéculatives, ainsi que les attaquants capables d'observer et de modifier l'état de la micro-architecture. Il a ensuite utilisé ces modèles pour décrire plusieurs attaques sur les exécutions transitoires et pour présenter des protections contre ces attaques, en insistant sur la définition précise des objectifs de sécurité attendus pour ces protections.

COLLOQUE – *PROBABILISTIC PROGRAMMING*

Les 29 et 30 juin 2022

La programmation probabiliste fournit de puissants outils pour la modélisation statistique. En s'appuyant sur la sémantique formelle, les compilateurs et autres outils en provenance des langages de programmation, la programmation probabiliste construit des évaluateurs efficaces pour les modèles et les applications en apprentissage statistique, utilisant les théories et les algorithmes d'inférence provenant des sta-

tistiques. Ce colloque, coorganisé avec Jean-Baptiste Tristan (Boston College), avait pour objectif de décrire des progrès récents et des applications de la programmation probabiliste en croisant les points de vue provenant des langages de programmation, des statistiques et de l'apprentissage. Il s'est tenu les 29 et 30 juin 2022 et a réuni Francis Bach (Inria), Gilles Barthe (Max Planck Institute), Atılım Güneş Baydin (Oxford University) Guillaume Baudart (Inria), Nicolas Chopin (ENSAE), Andrew Gelman (Columbia University), Andrew D. Gordon (Microsoft Research), Maria Gorinova (Twitter), Vikash K. Mansinghka (MIT), Jan-Willem van de Meent (University of Amsterdam), Xavier Rival (Inria), Joseph Tassarotti (Boston College) et Christine Tasson (Sorbonne Université).

RECHERCHE

Les activités de recherche de la chaire Sciences du logiciel s'effectuent dans le cadre de l'équipe-projet Inria Cambium, commune au Collège de France et à l'Inria Paris et dirigée par François Pottier, directeur de recherche Inria.

La recherche de l'équipe Cambium vise à améliorer la fiabilité et la sécurité du logiciel en faisant progresser les langages de programmation et les méthodes formelles de vérification de logiciel. Les principaux résultats de l'équipe pendant l'année universitaire 2021-2022 sont listés ci-dessous. Une description plus détaillée est disponible dans le rapport annuel d'activité Inria de l'équipe, https://raweb.inria.fr/rapportsactivite/RA2021/cambium/.

VÉRIFICATION DÉDUCTIVE DE PROGRAMMES

Léo Stefanesco, ATER du Collège de France, a soutenu sa thèse, dans laquelle il étend la logique de séparation concurrente Iris pour montrer des simulations entre implémentations concrètes et spécifications abstraites de modules. Nous avons également développé une nouvelle logique de programmes pour raisonner sur la consommation mémoire en présence de récupération automatique par GC. Enfin, nous avons formellement vérifié la correction et la complexité d'une structure de donnée transiente (à la fois éphémère et persistante).

VÉRIFICATION DE COMPILATEURS

Basile Clément a poursuivi ses travaux de thèse sur la validation *a posteriori* de compilateurs pour langages tensoriels en développant et en intégrant au compilateur Halide une implémentation de son algorithme de validation. En collaboration avec Andrew Appel (université de Princeton), nous avons vérifié formellement une

structure de données de type arbre préfixe avec représentation canonique, et l'avons intégrée dans le compilateur CompCert afin de réduire les temps de compilation. Nous avons également ajouté à CompCert le traitement de la construction `_Generic` du langage C11 et un mécanisme de génération de syntaxe abstraite CompCert C utilisable par des outils de vérification déductive. Enfin, nous avons démontré la correction de l'optimisation *tail modulo constructor* d'OCaml en utilisant la logique relationnelle d'Iris.

PROGRAMMATION FONCTIONNELLE TYPÉE EN OCAML

Nous avons entamé une formalisation du système de modules du langage OCaml par traduction vers un langage intermédiaire typé plus simple. Cette formalisation met en évidence des irrégularités dans les modules d'OCaml et suggère des pistes d'amélioration. En parallèle, nous avons poursuivi le travail préparatoire à l'intégration du parallélisme à mémoire partagée et des gestionnaires d'effets dans OCaml. Enfin, nous avons étudié la question du *debootstrap* du compilateur OCaml, c'est-à-dire la possibilité de le reconstruire sans amorçage depuis une version antérieure du compilateur.

MODÉLISATION ET TEST DE MODÈLES MÉMOIRE FAIBLEMENT COHÉRENTS

Nous avons poursuivi l'étude de formalismes axiomatiques et sémantiques pour décrire les modèles mémoire fournis par les processeurs multicœurs et les langages de programmation contemporains. Les travaux récents, en collaboration avec l'entreprise ARM Ltd, portent sur la modélisation et le test des mécanismes de gestion de la mémoire virtuelle dans l'architecture ARMv8.

PUBLICATIONS

Bour F. et Pottier F., « Faster reachability analysis for LR(1) parsers », *in* : *SLE 2021: Proceedings of the 14th ACM SIGPLAN International Conference on Software Language Engineering*, Chicago, Association for Computing Machinery (ACM), 2021, p. 113-125, https://doi.org/10.1145/3486608.3486903 [HAL : hal-03478172].

Courant N., Lepiller J. et Scherer G., « Debootstrapping without archeology: Stacked implementations in Camlboot », *The Art, Science, and Engineering of Programming*, vol. 6, nº 3, 2022, art. 13, https://doi.org/10.22152/programming-journal.org/2022/6/13 [HAL : hal-03917754].

Madiot J.-M. et Pottier F., « A separation logic for heap space under garbage collection », *Proceedings of the ACM on Programming Languages*, vol. 6, nº POPL, 2022, art. 11, https://doi.org/10.1145/3498672 [HAL : hal-03478162].

Madiot J.-M., Pous D. et Sangiorgi D., « Modular coinduction up-to for higher-order languages via first-order transition systems », *Logical Methods in Computer Science*, vol. 17, nº 3, 2021, https://doi.org/10.46298/lmcs-17(3:25)2021 [HAL : hal-03350199].

Mével G. et Jourdan J.-H., « Formal verification of a concurrent bounded queue in a weak memory model », *Proceedings of the ACM on Programming Languages*, vol. 5, nº ICFP, 2021, art. 66, https://doi.org/10.1145/3473571 [HAL : hal-03298759].

Moine A., Charguéraud A. et Pottier F., « Specification and verification of a transient stack », *in* : *CPP 2022: Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, Philadelphie, Association for Computing Machinery (ACM), 2022, p. 82-99, https://doi.org/10.1145/3497775.3503677 [HAL : hal-03472028].

Oliveira Vale A., Melliès P.-A., Shao Z., Koenig J. et Stefanesco L., « Layered and object-based game semantics », *Proceedings of the ACM on Programming Languages*, vol. 6, nº POPL, 2022, art. 42, https://doi.org/10.1145/3498703 [HAL : hal-03456034].

Raad A., Maranget L. et Vafeiadis V., « Extending Intel-x86 consistency and persistency: Formalising the semantics of Intel-x86 memory types and non-temporal stores », *Proceedings of the ACM on Programming Languages*, vol. 6, nº POPL, 2022, art. 22, https://doi.org/10.1145/3498683 [HAL : hal-03426997].

Stefanesco L., *Asynchronous and Relational Soundness Theorems for Concurrent Separation Logic*, thèse de doctorat, sous la direction de P.-A. Melliès, université de Paris, 2021, https://theses.hal.science/tel-03526298.