Sujet de thèse Labex : Environnement pour l'analyse de sécurité d'objets communicants

Directeurs de thèse :
Ludovic Apvrille (Enseignant-chercheur LTCI, HDR)
Aurélien Francillon (Enseignant-chercheur EURECOM)

20 janvier 2014

Les récents congrès traitant de l'électronique grand public (e.g., CES'2014) ont confirmé ce qui était pressenti ces dernières années : les "objets connectés" vont bientôt envahir nos vies. L'on peut citer notamment les objets personnels d'analyse du comportement (activité sportives, sommeil), les véhicules connectés (transports intelligents), les objets médicaux mobiles et communicants, et enfin la domotique. Ces équipements sont mis sur le marché après des années de recherche et développement concernant d'une part l'étude et la mise au point des protocoles et systèmes distribués, et d'autre part les évolutions technologiques continues des communications sans fil.

La sécurité et le respect de la vie privée sont bien entendu au centre des problématiques liés à la vulgarisation de ces nouveaux objets. Si les attaques sur les ordinateurs classiques restent d'actualité, la cyber-criminalité a pris un essor tout autre avec les smartphones [1]. En effet, ces derniers concentrent de nombreuses informations utilisateurs : contacts, calendriers, paiement, etc. L'interconnexion de nouveaux objets, parfois critiques (systèmes automobiles) ou très personnels (moniteur de sommeil) n'ouvrira que de nouvelles opportunités à cette cyber-criminalité. Prenons l'exemple des systèmes embarqués automobiles dans lesquels la diffusion de ransomware [2] pourrait avoir un impact financier et d'image très important. Dans le cadre d'un moniteur médical, les informations personnelles pourraient être revendues à des sociétés d'assurance.

Les attaques dans les systèmes embarqués sont actuellement réalisées de plusieurs façons. (1) L'exploitation dans le matériel d'une fuite d'information (consommation [13], temps de calcul variable [5, 14]) ou d'une injection de faute afin de récupérer du matériel cryptographique (par exemple, une clé) [2]. (2) L'exploitation d'une vulnérabilité logicielle afin d'obtenir des droits supérieurs sur le système connecté [12]. L'on peut citer par exemple une faille dans un noyau Linux ancien installé sur des caméras de vidéo-surveillance. (3) L'exploitation de différentes vulnérabilités matérielles et logicielles dont l'utilisation

1. l'on pourrait citer le nombre de malware Android : 400k fin nov 2013
2. Logiciel malveillant qui bloque l'usage du système tant qu'une rançon n'a pas été payée.

en séquence permet de réaliser des attaques sur le système [4]. Par exemple, la récupération d'informations sur un bus d'un système automobile, puis l'injection d'information sur ce bus permettent d'entrer sur un calculateur du système.

Bien entendu, la sécurisation de ces systèmes mobiles connectés est au centre des préoccupations de nombreux travaux de recherche. Ces travaux s'intéressent d'une part à la sécurisation du matériel, et d'autre part à la sécurisation du logiciel. Dans le cadre du matériel, il peut s'agir de stabiliser la consommation d'énergie d'un circuit quelque soit les calculs qu'il effectue. Dans le cadre du logiciel, il peut s'agir de prouver formellement que le logiciel ne comporte pas de faille de sécurité avant sa mise en service [10] ou de vérifier son intégrité [11].

Dans le cadre des systèmes embarqués communicants, les couches logicielles basses (drivers, firmwares) ainsi que le matériel en relation avec ces logiciels bas niveau constituent des points d'entrées critiques dans ces système [16]. Nous avons cité plus haut le cas d'un bus dans un système automobile. Il a aussi été montré récemment que les vulnérabilités dans les firmwares avaient un impact fort sur la sécurité de ces systèmes [8, 15], et sur la vie privée dans le cadre des caméras de surveillance [1]. Toutefois, comme mentionné ci-dessus, les travaux actuels se focalisent plutôt soit sur la conception ou l'analyse de la sécurisation du matériel, ou du logiciel, mais rarement l'analyse des deux conjointement. Pourtant, analyser le logiciel et matériel conjointement pourrait améliorer la sécurité du produit par l'identification de traces d'attaques complexes, c'est à dire impliquant des composants à la fois matériel et logiciel.

L'analyse de la sécurité des systèmes repose notamment sur les techniques de preuve formelle, et en particulier sur le model-checking classique [3], sur le model-checking symbolique [9, 6, 17], sur l'interprétation abstraite [7]. Un grosse limite de ces travaux repose sur leur orientation grandement logicielle, c'est à dire que des vulnérabilités complexes logicielles/matérielles ne peuvent être analysées. Repartant d'une approche particulièrement prometteuse appliquée avec succès pour purement du logiciel et basée sur du model-checking symbolique[6], nous proposons de mettre au point un nouvel environnement acceptant des modèles usuels matériel (e.g., code verilog), logiciels (C, assembleur) afin de produire des traces d'attaques. Notre environnement est bien entendu à même de s'appuyer sur KLEE/LLVM[6, 18], même si un état de l'art complet sera effectué afin de s'appuyer sur les meilleurs travaux dans ce domaine.

Ainsi, la thèse pourra être organisée de la façon suivante.

— Bibliographie sur les techniques d'analyse des systèmes embarqués, et plus généralement les techniques formelles d'analyse de sécurité pour le matériel et le logiciel. Une attention particulière sera portée aux techniques appliquées récemment avec succès, en particulier le model-checking symbolique.
— Définition d'un environnement d'analyse de sécurité. Cet environnement devra être conçu de façon à produire des traces mettant en évidence la séquence ds vulnérabilités / faiblesses à exploiter pour produire les attaques identifiées.
— Choix d'un nombre conséquent d'objets communicants de nouvelle génération (exemple : bracelets de monitoring, caméras de surveillance) pour lesquels

leur code verilog/C est accessible, et évaluation de l'environnement sur ces objets.

— Diffusion de l'environnement sous une licence libre

L'encadrement de la thèse sera effectué par Ludovic Apvrille et Aurélien Francillon.

Ludovic Apvrille travaille depuis des années à la conception de systèmes embarqués communicants sûr et sécurisés. Il a en particulier contribué au projet européen EVITA consistant à sécuriser les architectures automobiles de nouvelle génération, et notamment en terme d'analyse d'exigences de sécurité, d'attaques, de définition nd'une architecture de sécurité et de preuves de résistances à des attaques [ref]. Il a aussi développé un environnement, AVATAR, dédié à la conception et la preuve de sécurité de systèmes logiciels depuis des modèles SysML de haut-niveau [ref]. Enfin, Il a aussi contribué à l'identification automatique de malware sous Android [ref].

Aurélien Francillon travaille dans la sécurité des systèmes embarqués depuis de nombreuses années. En particulier sur les attaques logicielles[12], l'attestation [**?**], et sur les techniques qui peuvent améliorer significativement la sécurité logicielle en fournissant un support matériel efficace et peux coûteux [11, **?**].

L'équipe ainsi constituée présente les compétences en recherche fondamentale et appliquée.

1 Bibliography

Références

[1] I. Acre. The rise of the gadgets. *Security Privacy, IEEE*, 1(5) :78–81, 2003.

[2] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2) :370–382, 2006.

[3] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1) :3–51, Feb.–Mar. 2008.

[4] G. Bouffard, J. Iguchi-Cartigny, and J.-L. Lanet. Combined software and hardware attacks on the java card control flow. In E. Prouff, editor, *Smart Card Research and Advanced Applications*, volume 7079 of *Lecture Notes in Computer Science*, pages 283–296. Springer Berlin Heidelberg, 2011.

[5] D. Brumley and D. Boneh. Remote timing attacks are practical. In *In Proceedings of the 12th USENIX Security Symposium*, pages 1–14, 2003.

[6] C. Cadar, D. Dunbar, and D. Engler. KLEE unassisted and automatic generation of high-coverage tests for complex systems programs. In *OSDI*, 2008.

[7] G. Canet, P. Cuoq, and B. Monate. A value analysis for c programs. In *Source Code Analysis and Manipulation, 2009. SCAM '09. Ninth IEEE International Working Conference on*, pages 123–124, 2009.

[8] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*, 2011.

[9] V. Chipounov, V. Kuznetsov, and G. Candea. The S2E Platform : Design, Implementation, and Applications. *ACM Trans. Comput. Syst.*, 30(1) :2 :1–2 :49, Feb. 2012.

[10] D. W. Currie, X. Feng, M. Fujita, A. J. Hu, M. Kwan, and S. P. Rajan. Embedded Software Verification Using Symbolic Execution and Uninterpreted Functions. *International Journal of Parallel Programming*, 34 :61–91, 2006.

[11] K. E. Defrawy, A. Francillon, D. Perito, and G. Tsudik. SMART : Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust. In *Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego*, 2012.

[12] A. Francillon and C. Castelluccia. Code injection attacks on harvard-architecture devices. In *CCS '08 : Proceedings of the 15th ACM Conference on Computer and Communications Security*, pages 15–26, New York, NY, USA, October 2008. ACM.

[13] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. pages 388–397. Springer-Verlag, 1999.

[14] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. pages 104–113. Springer-Verlag, 1996.

[15] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE, 2010.

[16] V. Kuznetsov, V. Chipounov, and G. Candea. Testing closed-source binary device drivers with ddt. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, pages 12–12. USENIX Association, 2010.

[17] V. Kuznetsov, J. Kinder, S. Bucur, and G. Candea. Efficient state merging in symbolic execution. In *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '12, pages 193–204, New York, NY, USA, 2012. ACM.

[18] C. Lattner and V. Adve. LLVM : A compilation framework for lifelong program analysis & transformation. In *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, pages 75–86. IEEE, 2004.