Projet ANR BINSEC
analyse formelle de code binaire pour la sécurité

Sébastien Bardin (coordinateur)
prenom.nom@cea.fr
Airbus Group, CEA LIST, IRISA, LORIA, Université Grenoble Alpes

BINSEC : BINary-code analysis for SECurity
- Projet ANR INS, appel 2012
- Axes 1 (sécurité) et 2 (génie logiciel)
- Projet de recherche fondamentale sur 4 ans (2013-2017)
- Sujet : Techniques formelles d'analyse de sécurité au niveau binaire

Contexte, objectifs et défis. L'objectif général du projet BINSEC est de combler le fossé actuel entre le succès des méthodes formelles pour la sûreté logicielle (niveau source ou modèle) et les besoins des analyses de sécurité niveau binaire (analyse de vulnérabilité, analyse de malware), pour l'instant peu outillées (approches syntaxiques). Pour les malware, nous nous concentrons sur les problèmes de camouflage et d'obscurcissement (*obfuscation*). Pour les vulnérabilités, nous nous concentrons sur la découverte de bugs et sur la distinction entre bugs bénins et bugs exploitables.
Les défis principaux viennent de la structure même du code binaire (données et contrôle bas niveau), de la complexité des architectures modernes, du manque de compréhension claire de certaines propriétés considérées (obscurcissement, exploitabilité), de la présence d'un attaquant et enfin de la volonté de considérer des programmes non critiques (robustesse et passage à l'échelle).

Bien qu'a priori distincts, les domaines que nous attaquons partagent des problèmes communs, par exemple le manque de sémantique claire et unifiée, la reconstruction précise du flot de contrôle (même avec obscurcissement), le besoin de raisonnement bas niveau précis, etc. Le projet est ainsi architecturé autour de quelques problématiques générales (sémantique, analyses de base) sur lesquelles se basent les travaux applicatifs. Les résultats sont en partie intégrés dans la plate-forme open-source BINSEC.

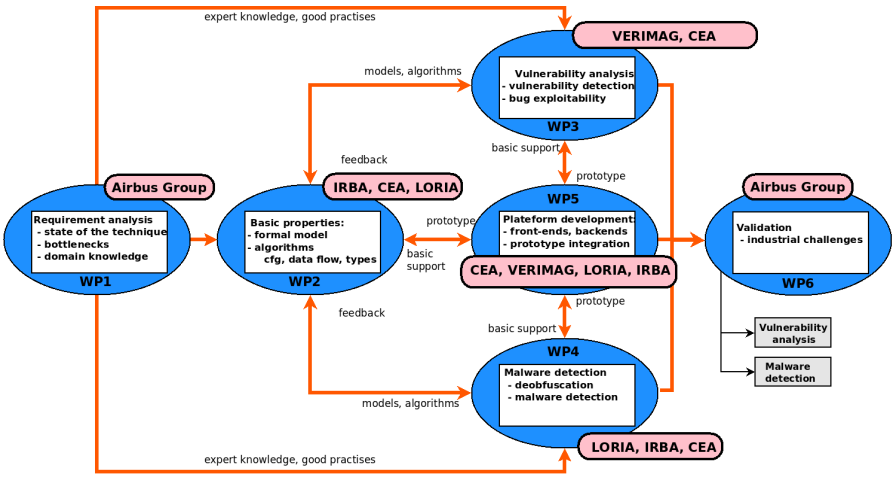

FIGURE 1 – Aperçu du projet

Quelques résultats. Les résultats du projet jusqu'à ce jour incluent : une représentation intermédiaire formelle et concise pour l'analyse de code binaire [5], basée entre autre sur un nouveau modèle mémoire à régions "bas niveau" [1, 2] raffinant le modèle usuel "à la CompCert" [7] ; une technique originale et un prototype de détection de "use-after-free" sur code exécutable [6] ; une technique et un prototype de désassemblage de code obscurci par auto-modification et chevauchement d'instruction [3] ; une plate-forme open-source d'analyse de code binaire [5] proposant des moteurs originaux d'analyse statique (reconstruction sûre de graphe de contrôle) et d'exécution symbolique [4] (exploration partielle précise). La plate-forme sera disponible à partir de juin 2016 (`http://binsec.gforge.inria.fr`) .

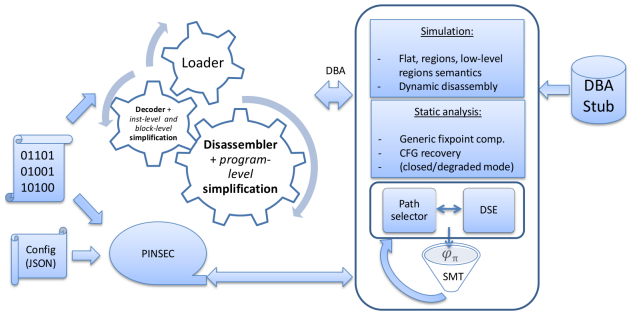

FIGURE 2 – Plate-forme Binsec

Participants. Sébastien Bardin, Frédéric Besson, Sandrine Blazy, Guillaume Bonfante, Richard Bonichon, Robin David, Adel Djoudi, Benjamin Farinier, Josselin Feist, Colas Le Guernic, Jean-Yves Marion, Laurent Mounier, Marie-Laure Potet, Than Dihnh Ta, Franck Védrine, Pierre Wilke, Sara Zennou.

Références

[1] Blazy S., Besson F., Wilke P. : A Precise and Abstract Memory Model for C using Symbolic Values. In : APLAS 2014. Springer, Heidelberg (2014)

[2] Besson F., Blazy S., Wilke P. : A Concrete Memory Model for CompCert. In : ITP 2015. Springer, Heidelberg (2015)

[3] G. Bonfante, J. Fernandez, JY. Marion, B. Rouxel, F. Sabatier, A. Thierry : CoDisasm : Concatic disassembly of self-modifying binaries with overlapping. In : CCS 2015. ACM (2015)

[4] R. David, S. Bardin, T. Thanh Dinh, J. Feist, L. Mounier, M.-L. Potet, and J.-Y. Marion. BINSEC/SE : A dynamic symbolic execution toolkit for binary-level analysis. In : SANER '16. IEEE, 2016

[5] A. Djoudi and S. Bardin. BINSEC : Binary code analysis with low-level regions. In : TACAS '15. Springer, 2015

[6] J. Feist, L. Mounier and M.L. Potet : Statically detecting Use-After-Free on Binary Code. Journal of Computer Virology and Hacking Techniques, 2014.

[7] Leroy, X., Appel, A.W., Blazy, S., Stewart, G. : The CompCert memory model. In : Program Logics for Certified Compilers. Cambridge University Press (2014)