COGITO: Génération de code au runtime pour la sécurité des systèmes embarqués

Contact: Damien Couroussé – damien.courousse@cea.fr

COGITO («Runtime Code Generation to Secure Devices» – appel ANR INS 2013) est un projet de recherche académique centré sur la sécurité des composants embarqués. Le consortium du projet est constitué de deux laboratoires du CEA, de l'École des Mines de Saint-Étienne et de l'INRIA de Rennes (auparavant laboratoire XLIM de l'université de Limoges). Le projet a démarré en octobre 2013 et se terminera fin mars 2017.

1 Les attaques physiques dans les systèmes embarqués

Dans le paysage de la cyber-sécurité, les composants embarqués ont une place particulière puisque leur plus grande menace sécuritaire vient des *attaques physiques*. Cette grande famille d'attaques vise à compromettre les protections mises en oœuvre en exploitant l'implémentation matérielle ou logicielle des mécanismes de sécurité. Les attaques physiques se révèlent très efficaces pour attaquer des composants cryptographiques par ailleurs robustes à la cryptanalyse, mais sont aussi par exemple utilisées pour attaquer les fonctions système d'un composant Java Card ; la gestion du code d'authentification d'un utilisateur (Verify PIN) est un autre composant logiciel qui n'utilise pas la cryptographie mais qui doit néanmoins être protégé contre les attaques physiques.

On distingue deux grandes classes d'attaques physiques : les attaques par canaux cachés (*side channel attacks*) et attaques en fautes (*fault attacks*). Les attaques par canaux cachés sont des attaques passives qui reposent sur l'observation de grandeurs physiques mesurables pendant que le mécanisme sécuritaire attaqué est en fonctionnement. Ces attaques servent le plus souvent à révéler un secret (par exemple une clé de chiffrement) en mettant en corrélation les observations de grandeurs physiques faites sur la cible attaquées avec les valeurs hypothétiques du secret. Les attaques en fautes sont des attaques actives. Elles consistent à introduire une erreur pendant que le mécanisme de sécurité s'exécute. L'exploitation de l'erreur introduite peut elle aussi servir par exemple à révéler un secret ou à outrepasser les droits d'un utilisateur.

Les attaques physiques ont été introduites au cours des années 1990 et font aujourd'hui l'objet d'un domaine à part entière de recherches scientifiques ; il existe dans la littérature un grand nombre de travaux consacrés à des attaques plus efficaces, mais aussi aux protections contre ces attaques. Très succinctement, on peut dire que les protections contre les attaques par canaux cachés reposent sur deux principes : diminuer le rapport signal-sur-bruit entre l'information utile et le reste de l'activité observable

sur le circuit (*hiding*), et mélanger les valeurs intermédiaires d'un algorithme de chiffrement avec de l'aléa (*masking*). Les protections contre les attaques en fautes reposent sur les principes de redondance (répéter plusieurs fois une opération pour que le résultat soit correct même si une opération est fautée) et de vérification de l'intégrité du calcul. Par ailleurs, des mécanismes matériels permettent de désactiver ou de tuer le circuit lorsqu'une attaque est détectée.

Nous observons cependant que, s'il existe une grande variété de moyens de protections, chaque modèle de protection répond à un modèle d'attaque en particulier : ainsi un produit est sécurisé par la superposition de nombreux mécanismes de protection, à la fois matériels et logiciels. En outre, chaque protection ajoute un coût supplémentaire au produit final (complexité de réalisation, surface siliciunm, consommation électrique, temps d'exécution du programme...). La difficulté de réaliser un produit sécurisé consiste alors à identifier les mécanismes de protections adéquats pour un modèle d'attaque, et à les intégrer dans un produit en limitant l'impact sur les performances et le coût de réalisation du produit.

2 Polymorphisme par génération de code au runtime

Dans le projet COGITO, nous nous intéressons à la génération de code au runtime et à son intérêt pour la sécurité des composants logiciels embarqués. Nous développons dans le projet un outil capable de donner la propriété de *polymorphisme* à un composant logiciel. Nous définissons le polymorphisme comme la capacité d'un composant logiciel de changer de forme sans changer de fonctionalité. Concrètement, nous utilisons la génération de code au runtime pour produire régulièrement une nouvelle implémentation en code machine du composant polymorphique, chaque implémentation étant différente des implémentations précédentes. Nous appelons *instance polymorphique* une implémentation du composant polymorphique, résultat d'une génération de code.

La génération de code se déroule *in situ* sur le système embarqué lui-même. Les générateurs de code polymorphique présentent une empreinte mémoire réduite, et la génération de code est rapide, afin de répondre aux contraintes de capacité mémoire et de limitation des ressources de calcul des systèmes embarqués. Dans le cadre de ce projet, notre plateforme d'expérimentation est la carte de développerement STM32F1-discovery, que nous avons volontairement choisie en raison de ses faibles capacité mémoire (8kO de RAM et 256kO de flash) et de sa faible puissance de calcul (la carte embarque un cœur 32-bits ARM Cortex-M3). Actuellement, la génération de code polymorphisme exploite plusieurs leviers pour apporter de la variabilité au code machine généré : allocation aléatoire de registres, sélection aléatoire d'instructions équivalentes, mélange aléatoire des instructions, et insertions d'instructions factices.

3 Intérêt du polymorphisme contre les attaques physiques

Le polymorphisme apporte une propriété intéressante contre les attaques physiques : chaque instance ayant une implémentation différente, son exécution présentera un com-

portement observable différent des autres instances polymorphiques (à la fois dans le temps et dans l'espace du circuit). Nous observons en effet que la variabilité temporelle et spatiale dans l'exécution d'un mécanisme de sécurité, si elle n'est pas corrélée au secret à protéger, est de nature à augmenter la difficulté des attaques physiques. Cette variabilité demande d'acquérir un plus grand nombre de mesures en attaque par canaux cachés, et peut diminuer les chances de succès d'une attaque en fautes.

Nos premiers résultats expérimentaux sur des attaques par canaux cachés valident cette hypothèse : alors qu'il est possible de retrouver une clé de chiffrement en moins de 50 mesures sur un composant AES logiciel démuni de protections, la même implémentation polymorphique est résistante à une attaque utilisant plusieurs milliers de courbes.

4 Conclusion

Dans cette présentation, nous proposons de dresser un rapide panorama des problèmes de cyber-sécurité dans les systèmes embarqués liés aux attaques physiques. Nous présenterons l'intérêt potentiel du polymorphisme de code. Dans le cadre du projet COGITO, notre mise en œuvre exploite la génération de code au runtime avec une technologie adéquate aux contraintes de mémoire et de ressources de calcul des petits composants embarqués (SmartCard, noeuds d'un réseau de capteur, etc.). Nous présenterons les premiers résultats expérimentaux sur des attaques par canaux cachés, et les grandes lignes des questions soulevées par notre approche scientifique.