Conception et implémentation d'un langage dédié à l'introspection de machine virtuelle

Lionel Hemmerlé, Guillaume Hiet, Frédéric Tronel, Pierre Wilke
CentraleSupélec, Inria, Univ Rennes, CNRS, IRISA

Contexte Afin de pouvoir réagir à une attaque, il est nécessaire de pouvoir la détecter. Cela nécessite de recourir à un système de détection d'intrusion (IDS) que l'on peut placer sur les systèmes que l'on souhaite protéger. Si un attaquant arrive à accéder a un haut niveau de privilège sur une machine infectée, il peut alors essayer de désactiver un IDS ou de lui communiquer des informations fausses, le rendant alors incapable de détecter l'intrusion. Ainsi, pour garantir le fonctionnement d'un IDS, il est nécessaire de le protéger en l'isolant du système qu'il surveille. Pour cela, nous pouvons utiliser les extensions de virtualisation du processeur. Nous proposer de placer l'IDS dans un hyperviseur et le système surveillé dans une machine virtuelle (VM). Dans cette situation, même si la VM est compromise par un attaquant, l'hyperviseur ainsi que l'IDS peuvent continuer de s'exécuter normalement.

Une fois placé dans un hyperviseur, l'IDS a accès à tout le contenu de la mémoire de la VM contrôlée par l'hyperviseur et à ses échanges avec les composants matériels, mais l'IDS perd les abstractions fournies par le système d'exploitation de la VM et doit donc être capable de reconstruire l'état interne de cette VM (par exemple retrouver la liste des processus exécutés) ainsi que les événements qui y ont lieu (par exemple la création d'un nouveau processus). Il s'agit du problème du fossé sémantique.

État de l'art Pour franchir ce fossé sémantique, différentes approches ont été proposées. Il est possible de modifier le code du noyau de la VM pour y intégrer des *hooks* [1] qui permettent de détecter certains événements ayant lieu dans la VM et de communiquer des informations à l'hyperviseur pour que celui-ci puisse déclencher des alertes. Cette méthode nécessitant de modifier le code de la VM, un attaquant peut alors détecter la présence de ces *hooks* et peut essayer de les contourner. Westphal et al. [2] ont développé une autre approche : leur outil permet à un utilisateur d'indiquer à l'hyperviseur dans quelles situations lever des alertes grâce à des programmes écrits dans un langage dédié. Leur approche nécessite cependant d'utiliser des fichiers de configurations supplémentaires pour chaque système d'exploitation utilisé dans les VMs afin de pouvoir effectivement franchir le fossé sémantique. Pour se passer de tels fichiers, il est possible d'utiliser des *hyperupcalls* [3] : une VM envoie des programmes à l'hyperviseur, qui peut alors les exécuter pour obtenir des informations sur l'état de cette VM. Cette approche est très intéressante, mais a été utilisée principalement pour optimiser l'utilisation des ressources matérielles par l'hyperviseur (par exemple en déterminant les zones de la mémoire inutilisées de la VM) et n'a donc pas été développée dans le but de détecter des intrusions. Ainsi, il est possible pour un attaquant de rendre les programmes exécutés par l'hyperviseur inopérants.

Approche Nous souhaitons créer un langage permettant à un hyperviseur de détecter des intrusions dans une VM en exécutant des programmes écrits dans ce langage. Ces programmes doivent alors être envoyés par cette VM à l'hyperviseur et doivent contenir toutes les informations (ou les traitements permettant de les obtenir) nécessaires au franchissement du fossé sémantique. Puisque cette approche permet à l'hyperviseur d'exécuter du code fourni directement par des VMs, il faut s'assurer qu'un attaquant ne puisse pas détourner notre langage pour compromettre l'hyperviseur. Pour éviter cela, il faut d'abord restreindre l'envoi de programmes par la VM à une période initiale où elle peut être considérée comme saine. De plus, puisque ces programmes prennent comme entrées des données qui peuvent être modifiées par l'attaquant (dans la mémoire de la VM), il faut pouvoir s'assurer que l'hyperviseur ne puisse pas être attaqué en exploitant d'éventuels bogues dans les programmes destinés à franchir le fossé sémantique.

Résultats Nous avons développé quatre attaques permettant à un *rootkit* de masquer des processus ou d'élever les privilèges d'un processus en manipulant des structures du noyau. Nous avons conçu des approches pour détecter ces attaques à partir de l'observation d'événements interceptés par l'hyperviseur. Ces travaux permettront d'identifier les fonctionnalités que devra proposer notre langage d'introspection. Afin de tester la détection des attaques reposant sur des événements, nous avons en outre développé un hyperviseur minimaliste en utilisant Bareflank [4].

Bibliographie

[1] Bryan D. Payne et al. "Lares: An Architecture for Secure Active Monitoring Using Virtualization". In: *2008 IEEE Symposium on Security and Privacy (sp 2008)*. 2008, pp. 233–247.

[2] Florian Westphal et al. "VMI-PL: A monitoring language for virtual platforms using virtual machine introspection". In: *Digital Investigation* 11 (2014). Fourteenth Annual DFRWS Conference, S85–S94.

[3] Michael Wei and Nadav Amit. "Leveraging Hyperupcalls To Bridge The Semantic Gap: An Application Perspective." In: *IEEE Data Eng. Bull.* 42.1 (2019), pp. 22–35.

[4] *Bareflank hypervisor*. URL: `https://github.com/Bareflank/hypervisor`.