Attaques par entrée-sortie et contremesures

Fernand Lone Sang*†, Vincent Nicomette*† et Yves Deswarte*†
*CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France
†Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ; F-31077 Toulouse Cedex 4, France

***Résumé*—Il devient aujourd'hui de plus en plus difficile de protéger efficacement les systèmes informatiques. Au fur et à mesure de l'accroissement de leur complexité, la surface d'attaque de ces systèmes s'est élargie et le nombre d'attaques perpétrées et réussies ne cessent de croître, en dépit des nombreux mécanismes de protection mis en place. Les attaques peuvent être réalisées par du logiciel s'exécutant sur le processeur, mais peuvent également reposer sur l'exploitation des entrées-sorties. Cet article s'intéresse à cette dernière catégorie d'attaques. Nous dressons un état de l'art de telles attaques sur les systèmes informatiques conçus autour de l'architecture Intel x86, notamment les ordinateurs personnels (PC) et, plus récemment, certains *System-On-Chip* (SoC). Nous détaillons ensuite quelques technologies matérielles qui permettent de s'en protéger et nous discutons finalement de leurs limites respectives.**

I. Introduction

De nos jours, les systèmes informatiques font partie intégrante de notre quotidien. Nous les utilisons, par exemple, pour travailler, pour échanger des informations, ou pour effectuer des achats. Malheureusement, ces systèmes sont encore régulièrement victimes d'attaques réussies, malgré les nombreux mécanismes de protection mis en place, en particulier dans l'architecture Intel x86. Les attaques peuvent être réalisées par du logiciel malveillant (*maliciel*) s'exécutant sur le processeur, en exploitant des erreurs d'implémentation matérielle ou logicielle (ex. un débordement de tampon). Mais elles peuvent aussi reposer sur l'exploitation des mécanismes d'entrée-sortie (E/S). C'est ce type d'attaques que nous étudions dans cet article. Nous commençons par rappeler quelques concepts propres à l'architecture Intel x86 en section II. Nous établissons ensuite un rapide état de l'art des attaques par E/S en section III, et nous présentons quelques technologies matérielles qui permettent de s'en protéger en section IV. Finalement, la section V conclut notre article et présente les perspectives envisagées.

II. Quelques éléments d'architecture

Une architecture matérielle Intel x86 typique se compose généralement d'un processeur et d'un *chipset*. Ce dernier se charge d'interconnecter le processeur, la mémoire principale et les différents contrôleurs d'E/S. Il se base pour cela sur deux puces distinctes, le *northbridge* et le *southbridge*. Le *northbridge* se charge d'interconnecter le processeur, la mémoire principale, les contrôleurs d'E/S nécessitant une bande passante importante (ex. la carte graphique) et le *southbridge*, chargé, lui, de relier les contrôleurs d'E/S nécessitant des ressources moindres (ex. les contrôleurs USB). Pour permettre au processeur de dialoguer avec les différents composants du système, l'architecture Intel x86 définit plusieurs espaces d'adressage. Nous nous focalisons, dans la suite de cet article, sur l'espace d'adressage mémoire dans lequel est projeté la mémoire principale ainsi que certains registres des contrôleurs d'E/S, voire même leur mémoire interne. On parle alors de *Memory-Mapped I/O* (MMIO)[1]. Grâce au *chipset*, l'accès à ces différents emplacements s'effectue de manière transparente. Le processeur accède à cet espace par l'intermédiare d'une *Memory Management Unit* (MMU). Il est également possible d'y accéder depuis un contrôleur d'E/S via le mécanisme matériel dit *Direct Memory Access* (DMA). La section qui suit effectue un état de l'art des attaques qui s'appuient sur les entrées-sorties, en particulier le mécanisme de DMA.

III. Classes d'attaques par les entrées-sorties

Il existe de nombreuses attaques par les E/S utilisant les mécanismes de DMA. Cette section détaille certaines de ces attaques, classées en fonction des éléments qu'elles ciblent.

A. Attaques DMA visant la mémoire principale

La plupart des attaques DMA existantes ciblent la mémoire principale. Elles cherchent à violer les propriétés de sécurité (intégrité et confidentialité) des composants logiciels chargés en mémoire ou des données manipulées par ceux-ci. Dans certaines circonstances, il est légitime d'utiliser les fonctionnalités d'un contrôleur d'E/S pour lire la mémoire principale. Tribble [1] et Copilot [2], des contrôleurs PCI dédiés à l'*inforensique* (investigation informatique légale), sont des exemples d'usage légitime. Dans des cas moins légitimes, le DMA peut être exploité pour corrompre la mémoire principale. L. Duflot [3] a montré par exemple qu'un code malveillant, s'exécutant avec peu de privilèges, peut mettre en place des transferts DMA depuis un contrôleur USB UHCI afin d'acquérir plus de privilèges. L'exécution d'un code malveillant sur le système attaqué n'est pas toujours nécessaire et un accès physique peut suffire. M. Dornseif [4], [5] et A. Boileau [6] ont montré qu'un périphérique FireWire (ex. un iPod modifié) peut accéder à la totalité de la mémoire principale au travers du contrôleur FireWire. Par la suite, plusieurs preuves de concept d'attaques au travers du bus FireWire ont été publiées : D.R. Piegdon [7] s'est intéressé, par exemple, aux techniques de compromission des systèmes Linux tandis que D. Aumaitre [8] s'est concentré sur les systèmes Windows. Finalement, L. Duflot *et al.* [9] ont montré récemment qu'un accès DMA peut être commandé à distance, suite à l'exploitation d'une vulnérabilité dans le *firmware* d'une carte réseau. G. Delugré [10] a étendu leur travaux en développant des outils de rétro-ingénierie pour étudier le *firmware* qu'ils ont exploité et a démontré que ce dernier aurait pu être modifié de façon à intégrer un *rootkit* indétectable.

[1]Précisons que l'architecture Intel x86 définit un autre mode d'E/S appelé *Programmed I/O* qui n'est plus guère utilisé de nos jours.

B. Attaques DMA visant les contrôleurs d'E/S

Certaines attaques par E/S ciblent des registres ou la mémoire interne des contrôleurs d'E/S. De telles attaques cherchent à exploiter directement les ressources fournies par le matériel, tout en contournant les mécanismes de protection de la mémoire principale. Afin de souligner que ces attaques s'effectuent directement entre contrôleurs d'E/S, nous désignons de telles attaques par « attaques DMA *peer-to-peer* ». M. Dornseif a probablement été l'un des premiers à démontrer la faisabilité de telles attaques. Pour illustrer ses travaux publiés dans [4], il a développé, sur des *chipsets* d'ancienne génération, une preuve de concept d'attaque DMA dans laquelle il modifie le *framebuffer* d'une carte graphique depuis un iPod connecté via le bus FireWire. Nous avons développé une preuve de concept [11] similaire pour illustrer nos travaux consacrés à ce type d'attaques [12] sur les *chipsets* intégrant davantage de moyens de protection. Finalement, des scénarios d'attaques plus complets ont été développés par A. Triulzi. La modification du *firmware* d'une carte réseau lui a permis de transférer les trames respectant un certain motif dans la mémoire interne de la carte graphique, où a été implanté un serveur *shell* sécurisé [13]. Il a ensuite étendu ses travaux en montrant que cet accès distant peut être utilisé pour charger, depuis la carte graphique, un nouveau *firmware*, téléchargé depuis l'Internet, dans une autre carte réseau [14]. Ce dernier lui a permis d'altérer le comportement de la carte réseau de façon à ce que certaines trames soient directement transférées vers une autre carte réseau, contournant ainsi tout filtrage de paquets mis en œuvre au sein du système d'exploitation.

IV. Contremesures matérielles

Il est possible de s'aider de technologies matérielles, telles que des *Input/Output Memory Management Units* (I/O MMU), pour se protéger de ces différentes attaques DMA. Une I/O MMU est un ensemble de composants matériels intégrés aux *chipsets* récents qui permettent de virtualiser la mémoire principale pour les contrôleurs d'E/S et de filtrer, entre autres, leurs accès à celle-ci. Ces composants sont généralement placés dans le *northbridge*, en coupure entre le bloc formé par les contrôleurs d'E/S et la mémoire principale. Du fait de leur agencement dans l'architecture matérielle, ces composants voient transiter toutes les requêtes d'accès à la mémoire principale depuis les contrôleurs d'E/S et protègent ainsi la mémoire principale de tout accès frauduleux. En revanche, ces composants sont moins efficaces contre les attaques entre contrôleurs d'E/S, en particulier ceux internes au *southbridge*. Il est probable, par exemple, que la preuve de concept développée par A. Triulzi [14] continue à fonctionner malgré l'utilisation d'une I/O MMU. Nous avons d'ailleurs montré qu'une I/O MMU présente généralement certaines limites qui peuvent être exploitées par des attaquants [15]. Pour répondre à ces problématiques de sécurité, le PCI-SIG, consortium en charge de la spécification du bus d'E/S PCI Express, a défini un ensemble d'extensions regroupées sous le nom d'*Access Control Services* (ACS). Les composants implémentant ces extensions peuvent être configurés pour mettre en œuvre différents types de contrôle d'accès, en particulier pour transférer toute communication entre contrôleurs d'E/S au *northbridge* afin d'être validée par une I/O MMU par exemple, ou tout simplement pour les bloquer. Il est important de noter que l'activation de ces services peut avoir un impact significatif sur les performances. J. Rutkowska [16] a proposé une technique permettant de protéger des régions quelconques de la mémoire principale contre d'éventuelles attaques DMA pour des systèmes dépourvus d'I/O MMU. Son idée consiste à reconfigurer le *chipset* de façon à ce que le *northbridge* redirige automatiquement tous les accès depuis les contrôleurs d'E/S aux régions de mémoire que l'on souhaite protéger vers un autre emplacement. Une tentative d'accès à ces régions de mémoire protégées depuis un contrôleur d'E/S peut provoquer, par exemple, un blocage complet du système en redirigeant l'accès vers un emplacement invalide. Il est cependant difficile de mettre en œuvre cette technique pour bloquer les attaques entre contrôleurs d'E/S.

V. Conclusion

Dans ce papier, nous avons effectué un rapide état de l'art des attaques par E/S ciblant les systèmes informatiques conçus autour de l'architecture Intel x86. Notre étude a permis de dégager deux familles d'attaques par E/S : (1) celles qui ciblent la mémoire principale et (2) celles qui ciblent les éléments d'autres contrôleurs d'E/S. Nous avons également présenté quelques technologies matérielles qui permettent de s'en protéger et nous avons évoqué certaines de leurs limites. Comme suite à nos travaux, nous envisageons de caractériser plus finement les attaques par E/S afin d'en déduire une classification plus exhaustive et de proposer quelques contremesures génériques pour chaque classe identifiée.

Références

[1] B. Carrier and J. Grand, "A Hardware-based Memory Acquisition Procedure for Digital Investigations," *Digital Investigation Journal*, vol. 1, no. 1, pp. 50–60, Feb. 2004.

[2] J. Nick L. Petroni, T. Fraser, J. Molina, and W. A. Arbaugh, "Copilot - a Coprocessor-based Kernel Runtime Integrity Monitor," in *Proceedings of the 13th USENIX Security Symposium*, 9-13 Aug. 2004.

[3] L. Duflot, "Contribution à la sécurité des systèmes d'exploitation et des microprocesseurs," Thèse de Doctorat, Université de Paris XI, Oct. 2007. [Online]. Available : http://www.ssi.gouv.fr/archive/fr/sciences/fichiers/lti/these-duflot.pdf

[4] M. Dornseif, "0wned by an iPod - hacking by Firewire," in *PacSec/core04*, 11-12 Nov. 2004.

[5] M. Becher, M. Dornseif, and C. N. Klein, "FireWire - all your memory are belong to us," in *CanSecWest/core05*, 4-5 May 2005.

[6] A. Boileau, "Hit by a Bus : Physical Access Attacks with FireWire," in *RUXCON 2006*, Oct. 2006.

[7] D. R. Piegdon, "Hacking in Physically Addressable Memory," in *Seminar of Advanced Exploitation Techniques, WS 2006/2007*, 12 Apr. 2007.

[8] D. Aumaitre, "A Little Journey Inside Windows Memory," *Journal in Computer Virology*, vol. 5, no. 2, pp. 105–117, 2009.

[9] L. Duflot, Y.-A. Perez, G. Valadon, and O. Levillain, "Can you still trust your Network Card ?" in *CanSecWest/core10*, 24-26 Mar. 2010.

[10] G. Delugré, "Closer to metal : reverse-engineering the Broadcom NetExtreme's firmware," in *Hack.lu*, Luxembourg, 27-29 Oct. 2010.

[11] F. Lone Sang, V. Nicomette, and Y. Deswarte, "Démonstration d'une attaque DMA *peer-to-peer* sur une carte graphique," Jan. 2011. [Online]. Available : http://homepages.laas.fr/nicomett/Videos/

[12] F. Lone Sang, V. Nicomette, Y. Deswarte, and L. Duflot, "Attaques DMA *peer-to-peer* et contremesures," in *Actes du 9ème Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC)*, Jun. 2011, (à paraître : http ://www.sstic.org/2011/actes/).

[13] A. Triulzi, "Project Maux Mk.II - « I 0wn the NIC, Now I want a Shell ! »," in *PacSec/core08*, 12-13 Nov. 2008.

[14] ——, "The Jedi Packet Trick takes over the Deathstar (or : « Taking NIC Backdoors to the Next Level »)," in *CanSecWest/core10*, Mar. 2010.

[15] F. Lone Sang, E. Lacombe, V. Nicomette, and Y. Deswarte, "Exploiting an I/OMMU Vulnerability," in *Proceedings of the 5th IEEE Intl. Conf. on Malicious and Unwanted Software (MALWARE)*, Oct. 2010, pp. 7–14.

[16] J. Rutkowska, "Beyond The CPU : Defeating Hardware Based RAM Acquisition," in *BlackHat DC*, 28 Feb. 2007.