Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PREMIER MINISTRE

S . G . D . S . N
Agence nationale de la sécurité des systèmes d'information
CERT-FR

Paris, le 24 avril 2017
N° CERTFR-2017-ACT-017

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2017-ACT-017

1 - Fonctionnalités de sécurité windows 10

En novembre 2015, le CERT-FR publiait un bulletin d'actualité [1] introduisant les mécanismes de sécurité basés sur la virtualisation [2]. Il s'agissait là de l'amélioration de sécurité la plus structurante pour Windows 10 et Windows Server 2016. Cependant, au fur et à mesure des mises à jour majeures, Microsoft a continué d'améliorer cette fonctionnalité et en a également rajouté de nouvelles.

Derrière ces changements, on distingue en filigrane les objectifs de Microsoft en termes de sécurité :

- Garantir l'intégrité du code exécuté en mode noyau (KMCI) et en mode utilisateur (UMCI)
- Protéger les secrets (CredentialGuard) et les structures du noyau (PatchGuard/HyperGuard)
- Casser les techniques d'exploitation courantes

Nous vous proposons aujourd'hui d'étudier les mécanismes permettant de garantir l'intégrité du code en mode noyau et en mode utilisateur que propose Windows 10. Ces fonctionnalités sont regroupées sous le terme "Device Guard". Depuis Windows Vista, le chargeur de programme et le noyau peuvent effectuer des vérifications sur les signatures des programmes et des pilotes. Ces mécanismes sont regroupés sous l'appellation `Code Integrity` (CI) et déclinés en `Kernel Mode Code Integrity` (KMCI) et `User Mode Code Integrity` (UMCI) respectivement.

KMCI

Sur les systèmes 64 bits depuis Windows Vista, le noyau refuse de charger les pilotes non signés par une liste restreinte d'autorités de certification. La confiance dans la signature pouvant parfois être abusée, certains codes malveillants peuvent néanmoins posséder une signature valide, au moins jusqu'à la révocation de leur chaine de signature.

Microsoft décida donc d'introduire avec Windows 8 les pilotes ELAM (Early Launch AntiMalware) comme mesure de défense en profondeur. Ces derniers sont chargés avant tous les autres pilotes afin d'évaluer ces derniers et d'empêcher leur chargement fussent-ils reconnus comme malveillants. Les pilotes ELAM doivent être signés par Microsoft et répondre à des critères exigeants (développés par une entreprise de sécurité reconnue, faible latence, faible empreinte, etcâŠ). Windows Defender, notamment, possède un pilote ELAM.

En parallèle de ELAM, les restrictions sur la confiance accordée à la signature des codes chargés en espace noyau s'est renforcée au fil des nouvelles versions de Windows, jusqu'à Windows 10 `Anniversary Update`. En effet, sur cette dernière, les pilotes doivent être signés par un certificat à validation étendue et contresigné par Microsoft.

UMCI / AppLocker

A l'instar du code en espace noyau, des vérifications basées sur la signature peuvent être effectuées sur le code chargé en espace utilisateur. Comme précisé dans le précédent bulletin d'actualité, ces vérifications étaient

jusqu'alors réservées à la plateforme ARM depuis Windows 8/RT. Windows 10 rend possible leur utilisation sur les autres plateformes. UMCI permet donc d'établir une liste blanche (ou noire), basée sur la signature, de codes autorisés à (ou interdit de) s'exécuter sur la machine. En cela, cette solution est similaire à AppLocker qui peut être vu comme une extension plus fine d'UMCI. En effet, il est recommandé d'utiliser AppLocker pour gérer les cas non supportés par UMCI (interdiction de certaines applications signées par Microsoft par exemple, etc...).

De plus, conscient que PowerShell expose une surface d'attaque de plus en plus exploitée, Microsoft a également décidé d'étendre l'emprise d'AppLocker à ce langage. En effet, ce dernier supporte, depuis sa version 3.0, le mode de langage contraint ("Constrained Language Mode") permettant de limiter grandement ses fonctionnalités (pas d'appel direct possible à l'API Win32 ou .NET, pas d'interaction avec les objets COM, etc...) [3]. Ce mode a pour principale conséquence d'empêcher la plupart des utilisations malveillantes de PowerShell, à commencer par les cadriciels d'attaque écrits dans ce langage.

Depuis PowerShell V5, ce mode de langage est automatiquement appliqué lorsque AppLocker est utilisé. Dès lors, seuls les scripts présents dans la liste blanche pourront s'exécuter dans le mode de langage complet.

A noter que ces restrictions s'appliquent également aux scripts Batch et ceux gérés par Windows Scripting Host (VBScript et JScript).

Ces mesures visent à prévenir l'exécution de code non signé (ou d'origine non contrôlée) mais peuvent se révéler difficiles à mettre en oeuvre ou même parfois être contournées [4]. Au titre de la défense en profondeur contre les menaces basées sur les langages de script, Microsoft a également développé une interface appelée AMSI (Anti-Malware Scan Interface) permettant de fournir aux logiciels antivirus un moyen de scanner le code tel qu'il est interprété, c'est à dire sans obfuscation, compression ou autre encodage. Cela vient contrecarrer la facilité qu'avait un attaquant pour rendre son code difficilement détectable par les antivirus.

Mise à profit du Virtual Secure Mode

Bien que l'objectif de KMCI soit d'assurer l'intégrité du code chargé en mode noyau, il n'est pas exclu que du code légitime soit vulnérable et exploité à des fins malveillantes [5]. Avant Windows 10, ce genre d'exploitation aboutissait souvent à la compromission totale de la machine puisque le code s'exécutant dans l'espace noyau du système d'exploitation possédait le plus haut niveau de privilège. Il devenait donc possible, par exemple, de désactiver les vérifications effectuées par KMCI.

Comme décrit dans le précédent bulletin d'actualité, Microsoft a introduit avec Windows 10 la sécurité basée sur la virtualisation. Cette dernière permet la ségrégation du monde "normal" (VTL0, le système d'exploitation classique) et du monde "sécurisé" (VTL1, système d'exploitation minimaliste), tout deux sous l'égide d'un hyperviseur. Dès lors, les accès entre ces deux mondes sont contrôlés et limités par l'hyperviseur ce qui permet, par exemple, d'éviter que la compromission du noyau en VTL0 implique automatiquement la compromission de toute la machine. Profitant de cette nouvelle isolation, Microsoft a donc décidé de migrer dans le monde "sécurisé" plusieurs fonctions de sécurité, dont KMCI qui a été renommé HVCI (Hypervisor-based Code Integrity). Ce dernier peut donc vérifier l'intégrité du code en mode noyau dans le monde "normal" sans pour autant être accessible ni modifiable par ce dernier. Cette vérification est appliquée en ne marquant comme exécutables que les pages mémoire contenant du code dont la signature respecte la politique d'intégrité déployée.

Conclusion

Le modèle historique de sécurité d'un système d'exploitation repose sur la confiance accordée à l'espace noyau. Or celui-ci nécessitera toujours du code tiers pour fournir aux utilisateurs le support matériel le plus complet. Il devient donc extrêmement difficile de maitriser l'intégrité du code privilégié sans perdre une compatibilité désormais considérée comme acquise. Microsoft a donc décidé de déplacer le coeur de confiance dans un noyau minimaliste et maitrisé, excluant de facto le noyau "classique" de cet espace. Ce dernier permet néanmoins de conserver les interfaces accessibles aux développeurs tiers.

Un effort a également été fait pour vérifier l'intégrité du code en espace utilisateur, mais il est évidemment beaucoup plus difficile de changer de paradigme comme cela a été fait pour le noyau. L'augmentation du niveau de sécurité se heurte ici directement aux fonctionnalités et/ou à l'utilisabilité du système.

Bien sûr, l'ensemble des fonctionnalités décrites ici reposent sur la présence de SecureBoot afin de limiter au maximum les possibilités de contournement.

Documentation

1. http://cert.ssi.gouv.fr/site/CERTFR-2015-ACT-045/
2. https://channel9.msdn.com/Blogs/Seth-Juarez/Windows-10-Virtual-Secure-Mode-with-David-Hepkin
3. https://msdn.microsoft.com/powershell/reference/5.1/Microsoft.PowerShell.Core/about/about_Language_Modes
4. https://github.com/subTee/BlueHat2016/blob/master/BlueHat%202016%20Trusted%20Things%20That%20Execute%20Thin
%20Final.pdf
5. http://www.cert.ssi.gouv.fr/site/CERTFR-2014-ACT-016/CERTFR-2014-ACT-016.html

2 - Rappel des avis émis

Dans la période du 17 au 23 avril 2017, le CERT-FR a émis les publications suivantes :

- CERTFR-2017-AVI-117 : Multiples vulnérabilités dans les produits VMware
- CERTFR-2017-AVI-118 : Multiples vulnérabilités dans Oracle Database Server
- CERTFR-2017-AVI-119 : Multiples vulnérabilités dans Oracle Java SE
- CERTFR-2017-AVI-120 : Multiples vulnérabilités dans Oracle Sun Systems Products Suite
- CERTFR-2017-AVI-121 : Multiples vulnérabilités dans Oracle Linux and Virtualization
- CERTFR-2017-AVI-122 : Multiples vulnérabilités dans Oracle MySQL
- CERTFR-2017-AVI-123 : Multiples vulnérabilités dans Google Chrome
- CERTFR-2017-AVI-124 : Vulnérabilité dans Drupal Core
- CERTFR-2017-AVI-125 : Multiples vulnérabilités dans le noyau Linux de Suse
- CERTFR-2017-AVI-126 : Multiples vulnérabilités dans Mozilla Firefox
- CERTFR-2017-AVI-127 : Multiples vulnérabilités dans les produits Cisco

Durant la même période, les publications suivantes ont été mises à jour :

- CERTFR-2017-ALE-008 : Multiples vulnérabilités dans Microsoft Windows XP et Windows Server 2003 (Extension de l'alerte à d'autres composants vulnérables.)
- CERTFR-2017-ALE-008 : Multiples vulnérabilités dans Microsoft Windows XP et Windows Server 2003 (Extension de l'alerte à d'autres composants vulnérables.)
- CERTFR-2017-ALE-008 : Multiples vulnérabilités dans Microsoft Windows XP et Windows Server 2003 (Extension de l'alerte à d'autres composants vulnérables.)

Gestion détaillée du document

24 avril 2017 version initiale