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 09 novembre 2015
N° CERTFR-2015-ACT-045

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2015-ACT-045

1 - Les mécanismes de sécurité basés sur la virtualisation dans Windows 10 et Windows Server 2016

Avec la sortie de Windows 10, Microsoft a introduit de nouvelles fonctionnalités de sécurité. Certaines sont des évolutions de fonctionnalités déjà existantes dans les précédentes versions de Windows, d'autres sont plus innovantes, comme celles basées sur la virtualisation afin d'ajouter de nouvelles frontières de sécurité.

Une base commune : *Virtual Secure Mode*

Les fonctionnalités de sécurité basées sur la virtualisation reposent sur une base commune appelée *Virtual Secure Mode* (VSM), constituée d'une nouvelle version de l'hyperviseur Microsoft (Hyper-V) et du noyau Windows.

Dans le contexte d'Hyper-V, l'unité d'isolement dans lequel s'exécute un système d'exploitation est appelée une partition. VSM utilise deux partitions :

- une partition dans laquelle s'exécute le système Windows normal, avec un niveau de privilège virtuel VTL 0 (*Virtual Trust Level* 0). Nous appellerons cette partie « monde normal » ;
- une partition dans laquelle s'exécute un système Windows minimal hébergeant uniquement des composants critiques, avec un niveau de privilège virtuel VTL 1, plus privilégié que VTL 0. Nous appellerons cette partie « monde sécurisé ». Dans ce monde, la partie noyau est appelée *Secure Kernel Mode* (SKM) et la partie utilisateur *Isolated User Mode* (IUM).

Les deux mondes peuvent communiquer par des interfaces bien définies. En revanche, du code s'exécutant dans le monde normal, qu'il soit en mode utilisateur ou noyau, ne peut pas accéder à la mémoire du monde sécurisé. Inversement, du code s'exécutant dans la partie IUM, n'a pas d'accès privilégié au monde normal.

Pour plus de détails, se référer aux liens dans la partie Documentation.

Les utilisations possibles

Les applications de la sécurité basée sur la virtualisation sont multiples. Microsoft en a d'ores et déjà présenté plusieurs :

- *Credential Guard* : le service d'authentification LSASS est maintenant divisé en deux parties. Une partie sensible, (`LsaIso.exe`), contenant notamment des empreintes de mots de passe, s'exécute dans la partie IUM du monde sécurisé. L'autre partie, (`lsass.exe`), s'exécute dans le monde normal. Cette fonctionnalité est une contre-mesure efficace contre les extracteurs de mots de passe actuels, comme Mimikatz qui, à partir du monde normal, ne peuvent plus extraire les secrets se trouvant dans la mémoire de la partie sensible ;
- *Hypervisor Code Integrity* (HVCI) : la vérification obligatoire de l'authenticité et de l'intégrité de code noyau est présente depuis plusieurs années pour les systèmes Windows 64 bits (*Driver Signature Enforcement*). Avec HVCI, ce processus de vérification se déroule maintenant dans la partie SKM du monde sécurisé, augmentant la difficulté d'un potentiel contournement. Il est également possible de configurer une politique

pour la vérification obligatoire du code s'exécutant en mode utilisateur, jusqu'à maintenant disponible, avec moins de contrôle, seulement sur Windows RT ou Windows Phone (tablettes et téléphones mobiles) ;
– vTPM : un TPM (*Trusted Platform Module*) virtuel pour protéger des secrets (par exemple des clés de chiffrement).

Les pré-requis matériels

Il existe des pré-requis pour utiliser VSM. Certains pré-requis ne sont pas obligatoires pour le fonctionnement de VSM, mais sont néanmoins essentiels pour la validité du modèle de sécurité proposé.

VSM utilise des fonctionnalités intégrées au matériel comme :
– les extensions de virtualisation intégrées aux processeurs, par exemple Intel VT-x ou AMD RVI ;
– le support de la translation d'adresse de second niveau (*Second-Level Address Translation* ou SLAT), comme *Intel Extended Page Tables* (EPT) ou AMD *Rapid Virtualization Indexing* (RVI). VSM utilise notamment cette technologie pour contrôler finement les droits d'accès aux pages de la mémoire physique ;
– une IOMMU (*I/O Memory Management Unit*) permettant de contrôler les accès à la mémoire physique effectués par des périphériques. Cette technologie permet d'éviter qu'un périphérique malveillant puisse accéder directement à la mémoire physique du monde sécurisé ;
– l'UEFI avec *Secure Boot* permettant de s'assurer que la chaîne de démarrage de la machine est de confiance. Ceci permet par exemple d'éviter qu'un *firmware* malveillant modifie l'hyperviseur ou sa configuration au moment de son initialisation, remettant en cause le modèle de sécurité fourni par VSM ;
– un TPM (*Trusted Platform Module*) pour protéger des secrets.

Les risques d'utilisation non désirée

Il est légitime de s'interroger sur l'éventuelle utilisation malveillante de ces nouvelles fonctionnalités de sécurité. En effet, un code malveillant situé dans le monde sécurisé serait alors protégé. Une copie mémoire classique en mode noyau depuis le monde normal ne permettrait pas de le collecter.

Pour s'exécuter dans le monde sécurisé, le code doit posséder une signature avec un niveau de confiance élevé (une vulnérabilité de l'IOMMU/SLAT ou de l'hyperviseur pourrait permettre de s'en affranchir). À ce jour, Microsoft n'a pas annoncé la possibilité pour des tiers d'obtenir une telle signature.

Dans le cas où le code malveillant se trouverait dans la partie IUM, un deuxième code contrôlé par l'attaquant sera nécessaire dans le VTL0, du fait du cloisonnement. En effet, c'est dans le monde normal que l'activité des utilisateurs se déroule. Cet agent pourrait être détecté par des moyens plus classiques.

Il faut enfin noter qu'un acteur malveillant peut mettre en place une isolation similaire, sans la présence de VSM, en développant entièrement la partie gérant la virtualisation.

Conclusion

Le modèle de sécurité offert par *Virtual Secure Mode* semble robuste et les possibilités d'utilisation sont intéressantes. Il faut toutefois garder à l'esprit que ce modèle repose sur la présence et la fiabilité de fonctionnalités bas niveau/matérielles.

Documentation

– *Windows 10 security overview*, 1er octobre 2015 :
https://technet.microsoft.com/en-us/library/mt601297(v=vs.85).aspx
– Alex Ionescu, vidéo de la présentation *Battle Of The SKM And IUM: How Windows 10 Rewrites OS Architecture*, Black Hat USA 2015, août 2015 :
https://www.youtube.com/watch?v=BXuCOlTUWgs
– Alex Ionescu, support de la présentation *Battle Of The SKM And IUM: How Windows 10 Rewrites OS Architecture*, Black Hat USA 2015, août 2015 :
http://www.alex-ionescu.com/blackhat2015.pdf
– Présentation de l'*Isolated User Mode*, partie 1, 5 août 2015 :
https://channel9.msdn.com/Blogs/Seth-Juarez/Isolated-User-Mode-in-Windows-10-with-Dave-Probert
– Présentation de l'*Isolated User Mode*, partie 2, 20 août 2015 :
https://channel9.msdn.com/Blogs/Seth-Juarez/Isolated-User-Mode-Processes-and-Features-in-Windows-10-with-Logan-Gabriel

– Présentation de l'*Isolated User Mode*, partie 3, 28 août 2015 :
https://channel9.msdn.com/Blogs/Seth-Juarez/More-on-Processes-and-Features-in-Windows-10-Isolated-User-Mode-with-Dave-Probert
– *Windows 10 Virtual Secure Mode with David Hepkin*, 14 octobre 2015 :
https://channel9.msdn.com/Blogs/Seth-Juarez/Windows-10-Virtual-Secure-Mode-with-David-Hepkin

2 - EMET 5.5 et configuration de la protection « Untrusted font » sous Windows 10

Introduction

La version 5.5 d'EMET est sortie le 1er Octobre 2015. Cette version introduit notamment la configuration de la protection « Untrusted Font » par processus protégé, uniquement supportée par Windows 10.

Cette configuration s'effectue simplement en modifiant pour chaque processus la valeur « MitigationOptions », de la clé de registre « HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ » qui lui est associée.

Deux clés de registre supplémentaires permettent de configurer cette nouvelle fonctionnalité au niveau du système lui-même :

– « HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Kernel » : « MitigationOptions », réalisé localement ;
– « HKLM\SOFTWARE\Policies\Microsoft\Windows NT\MitigationOptions » : « MitigationOptions_FontBocking », réalisé via les objets de politique globale (GPO).

Description des changements de configuration

Tout d'abord, la variable globale *PspSystemMitigationOptions* est initialisée au chargement du système d'exploitation par la fonction *CmGetSystemControlValues*, qui va chercher la valeur « MitigationOptions » dans le registre. Cette valeur n'est plus mise à jour après le chargement de l'OS ; ce dernier doit donc être redémarré explicitement pour prendre en compte un éventuel changement.

Lors de l'application de la GPO, la fonction *ProcessGroupPolicyMitigationOptions* de « gpprefcl.dll » va appliquer un masque de 0xFFF0FFFF‘FFFFFFFF sur la valeur existante et surcharger l'octet ainsi mis à zéro par sa propre valeur. Cette application nécessite donc un redémarrage du système.

Enfin, lors de la création d'un processus, la fonction *PspAllocateProcess* du noyau Windows va générer une valeur héritant de la variable globale *PspSystemMitigationOptions* ainsi que celle associée au processus via le registre (obtenue via la fonction *PspReadIFEOMitigationOptions*). La gestion de cet héritage est effectuée avec *PspInheritMitigationOptions*. Par la suite, selon la valeur ainsi obtenue, les bits *DisableNonSystemFonts* et *AuditNonSystemFontLoading* de la variable *EPROCESS.Flags3* du processus courant sont initialisés :

– *DisableNonSystemFonts* est mis à 1 si l'application du masque 0x00010000‘00000000 ne donne pas un résultat nul ;
– *AuditNonSystemFontLoading* est mis à 1 si l'application du masque 0x00030000‘00000000 ne donne pas un résultat nul.

Description de la fonctionnalité

Lors du chargement d'une police de caractères par la fonction *LoadFont* du pilote « win32kfull.sys » (par exemple au travers d'un appel à la fonction *NtGdiAddFontResourceW*), un appel à *GetCurrentProcessFontLoadingOption* est effectué. Cette fonction va en réalité effectuer un appel à la fonction *NtQueryInformationProcess* avec en paramètre une valeur « ProcessInformationClass » valant 0x34 et un paramètre « ProcessInformation » initialisé à 0x9. L'appel système va ainsi lire la valeur de la variable *EPROCESS.Flags3* du processus courant, et retourner 0, 1 ou 2 en fonction des valeurs des deux bits (notons au passage que cette valeur peut être modifiée à la volée par l'appel système *NtSetInformationProcess* en utilisant le même valeur pour le paramètre « ProcessInformationClass »).

Le type de la police spécifiée est obtenu via un appel à *GetFirstNonSystemFontPath*. Si ce dernier n'est pas concluant, selon la valeur obtenue précédemment :

– si *DisableNonSystemFonts* est présent, l'appel est interdit et la tentative de chargement est journalisée ;

– si *AuditNonSystemFontLoading* est présent, l'appel est autorisé mais la tentative de chargement est journalisée.

Le test effectué par *GetFirstNonSystemFontPath* est extrêmement simple : il appelle *IsFileInSystemFontsDir* qui, comme son nom l'indique, teste simplement si le chemin absolu du fichier débute bien par le dossier des polices système, lui-même initialisé à la valeur « \SystemRoot\Fonts ».

Conclusion

Cette évolution témoigne d'une volonté globale de Microsoft de gérer le problème de la recrudescence des vulnérabilités impactant des polices de caractères (voir également le bulletin d'actualité CERTFR-2015-ACT-044).

Le CERTFR recommande donc l'utilisation de cette option (déployable via le logiciel EMET, ou simplement sur des parcs en GPO) permettant d'éviter ainsi des vulnérabilités liées au chargement de polices de caractères depuis des applicatifs tournant en espace utilisateur.

Documentation

– http://www.cert.ssi.gouv.fr/site/CERTFR-2015-ACT-044/index.html
– Enhanced Mitigation Experience Toolkit :
https://technet.microsoft.com/en-us/security/jj653751

3 - Rappel des avis émis

Dans la période du 02 au 08 novembre 2015, le CERT-FR a émis les publications suivantes :

– CERTFR-2015-AVI-459 : Vulnérabilité dans Huawei P8
– CERTFR-2015-AVI-460 : Multiples vulnérabilités dans Cisco FireSIGHT Management Center
– CERTFR-2015-AVI-461 : Multiples vulnérabilités dans Mozilla Firefox
– CERTFR-2015-AVI-462 : Multiples vulnérabilités dans les produits Huawei
– CERTFR-2015-AVI-463 : Multiples vulnérabilités dans Apache OpenOffice
– CERTFR-2015-AVI-464 : Multiples vulnérabilités dans Google Android

Gestion détaillée du document

09 novembre 2015 version initiale.