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 29 mars 2016
N° CERTFR-2016-ACT-013

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2016-ACT-013

1 - PowerShell JEA (Just Enough Administration)

Introduction

On accorde traditionnellement des droits d'administration à tout utilisateur chargé d'une ou plusieurs tâches d'administration informatique. Selon le système d'exploitation, le niveau d'accès peut varier d'un contrôle total sur tout le système à des niveaux plus granulaires où seuls certains composants peuvent être contrôlés. Ce dernier modèle est appelé contrôle d'accès à base de rôles.

Même le contrôle d'accès à base de rôles fournit plus de droits que nécessaire, ce qui entraîne une surface d'attaque trop importante sur le système. Ce problème n'est pas récent et va de pair avec le principe de moindre privilège : les permissions associées à un compte utilisateur devraient être exactement celles nécessaires à l'accomplissement de la tâche.

Dans les environnements Windows, PowerShell est de plus en plus utilisé pour les tâches d'administration (par exemple pour l'administration de l'annuaire Active Directory, de l'hyperviseur Hyper-V, du serveur de messagerie Exchange Server, de Sharepoint Server, de SQL Server\ldots).

Pour mettre en œuvre un modèle d'accès à base de rôles, PowerShell intègre une solution appelée JEA (*Just Enough Administration*). Cette fonctionnalité est intégrée depuis la version 5 de PowerShell (qui nécessite l'installation de Windows Management Framework 5.0), disponible à partir de Windows 7 SP1 et Windows 2008 R2 SP1. JEA permet à des utilisateurs du domaine d'effectuer certaines tâches d'administration dans un espace d'exécution limité, sans que les droits d'administration ne leur soient accordés. Il est ainsi possible de définir une granularité plus forte dans les autorisations concernant les commandes d'administration PowerShell.

Configuration et fonctionnement

Dans un premier temps, il faut définir un type de session PowerShell restreinte, qui sera appliqué à des profils d'administrateurs selon leur groupe d'appartenance. Cette session correspond à un point de chute (*endpoint*) où les utilisateurs se connectent et accèdent à des fonctionnalités PowerShell limitées selon les paramètres de la session.

La session PowerShell restreinte est lancée avec un compte virtuel (SID de la forme S-1-5-94-ID) qui n'existe que le temps de la session. Ce compte virtuel est, par défaut, membre du groupe Administrateurs intégrés ou du groupe Administrateurs du domaine sur un contrôleur de domaine. La génération d'un fichier de configuration vierge peut être effectuée avec la *cmdlet* `New-PSSessionConfigurationFile`. Notes:

- L'appartenance du compte virtuel à un ou plusieurs groupes est configurable en positionnant le paramètre *RunAsVirtualAccount* à la valeur *$false* et modifiant la valeur du paramètre *RunAsVirtualAccountGroups*.
- Une cmdlet est une commande qui est utilisée dans l'environnement Windows PowerShell.

Voici un exemple de configuration minimale d'un fichier de définition de session :

```
SessionType = 'RestrictedRemoteServer'
RunAsVirtualAccount = $true
```

```
RoleDefinitions = {'DOM-EXEMPLE\JEA_HelpDesk' =
{RoleCapabilities = 'HelpDesk'}}
```

Dans cet exemple, la valeur *RestrictedRemoteServer* permet d'exposer automatiquement le minimum de commandes nécessaires pour faire de l'administration à distance. L'attribut *RunAsVirtualAccount* indique que PowerShell utilise un compte virtuel pour cette session. L'attribut *RoleDefinitions* permet de définir des droits d'exécution de commandes qui sont assignés aux utilisateurs membres du groupe JEA_HelpDesk du domaine DOM-EXEMPLE. La session peut ensuite être enregistrée localement sur la machine Windows à l'aide de la *cmdlet* `Register-PSSessionConfiguration`.

On doit également définir ce qu'un utilisateur a le droit de faire sur la session PowerShell en créant des *Role Capabilities*. Ces *capabilities* représentent une liste blanche de fonctions, de commandes, de paramètres, de scripts et d'applications définie dans un fichier `.psrc`. Ce fichier peut être généré à l'aide de la *cmdlet* `New-PSRoleCapabilityFile`. Voici un exemple de fichier définissant des commandes généralement utilisées par un service d'aide aux utilisateurs :

```
VisibleCmdlets = 'Unlock-ADAccount',
'Search-ADAccount',
'Enable-ADAccount',
'Disable-ADAccount',
{ Name = 'Add-ADGroupMember'; Parameters =
{Name = 'Identity'; ValidateSet = 'Simple utilisateur'},
{Name = 'Members'}},
{ Name = 'Remove-ADGroupMember'; Parameters =
{Name = 'Identity'; ValidateSet = 'Simple utilisateur'},
{Name = 'Members'}}
```

Dans cet exemple, ne sont autorisées que les *cmdlets* suivantes :

- débloquer des comptes (`Unlock-ADAccount`) ;
- effectuer des recherches dans l'annuaire (`Search-ADAccount`) ;
- activer et désactiver des comptes (`Enable-ADAccount` et `Disable-ADAccount`) ;
- ajouter et supprimer des utilisateurs dans des groupes (`Add-ADGroupMember` et `Remove-ADGroupMember`).

Pour les *cmdlets* `Add-ADGroupMember` et `Remove-ADGroupMember`, seuls les paramètres Members et Identity sont autorisés et le paramètre Identity ne peut prendre pour valeur que le groupe Simple utilisateur.

La configuration de JEA requiert une connaissance approfondie de PowerShell, afin d'être en mesure d'identifier les commandes ayant besoin d'être exposées à certains utilisateurs. Par exemple, un service d'assistance aux utilisateurs peut avoir besoin d'ajouter/retirer un utilisateur d'un groupe, de débloquer un compte, de redéfinir un mot de passe, de désactiver un compte, etc. Il y a bien souvent autant d'actions que de commandes à identifier et lister. Pour contrebalancer ces difficultés, l'outil JEA Helper Tool développé par Microsoft vise à aider à la création des *capabilities* et, plus généralement, à l'utilisation et au déploiement de JEA.

Points de vigilance

Il est possible d'exposer des commandes permettant aux utilisateurs de s'affranchir des restrictions imposées par JEA. Dans ce cas, l'utilisateur possède les droits et privilèges du compte virtuel. À titre d'exemple, la commande `Invoke-Expression`, si elle est autorisée, permet d'invoquer n'importe quelle commande et donc de contourner la politique de sécurité imposée par JEA. Il est également important de s'assurer que les fichiers de configuration sont présents dans des répertoires accessibles seulement à des comptes privilégiés. Les utilisateurs ne doivent pas pouvoir modifier ces fichiers. Enfin, il est essentiel de ne jamais donner aux utilisateurs la possibilité de modifier les scripts accessibles dans la session JEA. Il convient donc d'être prudent avec des commandes comme `Copy-Item`, `Move-Item` et les commandes permettant des actions telles que l'ajout d'un nouvel utilisateur.

On retrouve les mêmes problématiques avec la commande Unix *sudo* permettant à l'administrateur de déléguer certaines tâches nécessitant les droits *root* à de simples utilisateurs.

Recommandations

La complexité de certaines commandes peut-être réduite à l'aide de scripts exposés aux administrateurs utilisant dans sessions PowerShell limitées. Les scripts sont exécutés dans un contexte sans restriction : il est possible d'utiliser des variables et plusieurs commandes sans restriction. Cette approche simplifie l'expérience de l'utilisateur final et réduit la probabilité d'exposer involontairement certaines commandes dangereuses. Le seul inconvénient est le coût de la création des scripts et leur maintien dans le temps.

Puisque JEA permet à des utilisateurs non privilégiés de lancer des commandes dans un contexte privilégié, la journalisation et l'audit des journaux sont importants. L'activation de l'audit d'exécution PowerShell est donc vivement conseillée (voir CERTFR-2016-ACT-002).

Documentation

– https://gallery.technet.microsoft.com/JEA-Helper-Tool-20-6f9c49dd

2 - Rançongiciel et autres maliciels en Javascript

Découvert en décembre 2015, Ransom32 est le premier rançongiciel connu écrit en Javascript.

Javascript étant un langage interprété, il a besoin d'un logiciel tiers, un interpréteur, pour être exécuté. Ces interpréteurs sont souvent intégrés à des applications. On peut citer par exemple :

– les navigateurs, qui peuvent ainsi exécuter les scripts javascript des pages web ;
– Adobe Reader, qui peut interpréter du code Javascript parfois inclus dans des fichiers PDF.

Comme le code Javascript de Ransom32 ne peut pas être exécuté directement, il est intégré à l'interpréteur NW.js. Cet exécutable (ainsi que les ressources qu'il utilise) est délivré dans une archive WinRAR auto-extractible avec une configuration permettant de l'exécuter lors de l'ouverture. Le mode d'infection reste donc classique : la victime reçoit un exécutable en pièce jointe d'un courriel et doit le lancer volontairement pour infecter son système. De plus, la présence de NW.js dans l'archive alourdit beaucoup la pièce jointe (22 Mo) et contribue à la rendre suspecte.

Javascript présente malgré tout des atouts : simplicité de développement et portabilité, ainsi que de bonnes possibilités pour rendre le code difficilement lisible (obfuscation). L'utilisation par un maliciel d'un interpréteur déjà présent sur le système d'exploitation ajoute les avantages suivants : une taille réduite (pas besoin de fournir l'interpréteur) et une extension plus discrète que .exe (.js).

Sous Windows, un interpréteur est installé par défaut : il s'agit de Windows Script Host. Il est ainsi possible, par exemple sous Windows 7 et 8, d'exécuter (en demandant l'ouverture ou par un double-clique) un script Javascript contenu dans un fichier ayant une extension en .js. Cette possibilité est déjà utilisée par une version du rançongiciel Locky.

Mesures de prévention

Pour empêcher l'exécution de scripts sous Windows, une solution consiste à créer l'entrée suivante dans la base de registre avec pour valeur 0 :

`HKEY_LOCAL_MACHINE\Software\Microsoft\Windows Script Host\Settings\Enabled`

On peut également limiter cette configuration à l'utilisateur courant :

`HKEY_CURRENT_USER\Software\Microsoft\Windows Script Host\Settings\Enabled`

Il faut souligner que cette solution empêchera également l'exécution des scripts VBS (extension .vbs), y compris via la commande "cmd /c". Il sera par contre toujours possible d'exécuter des scripts PowerShell. Il convient donc de vérifier que cette mesure est compatible avec les procédures d'administration du système d'information.

Une autre solution consiste à associer les fichiers .js à une application qui ne pourra pas interpréter le script, par exemple un éditeur de texte tel que le bloc-notes.

Enfin, si le service de messagerie le permet et que les échanges de codes Javascript ne sont pas nécessaires, il peut être judicieux de filtrer les pièces jointes en .js et les archives en contenant.

Javascript dans des documents PDF

Concernant le code Javascript inclus dans des documents PDF, le CERT-FR rappelle qu'il est possible avec certaines applications d'en interdire l'exécution.

Sous Adobe Acrobat Reader, il faut aller dans le menu Edition puis Préférences et décocher l'option "Activer Acrobat JavaScript" dans la catégorie JavaScript. Lors d'une ouverture de document PDF contenant du Javascript, un bandeau jaune signalera que Javascript est désactivé et proposera un bouton pour l'activer cette fois ou de façon permanente pour ce document.

Pour Foxit Reader, il faut aller dans le menu Edition puis Préférences et décocher l'option "Autoriser les actions JavaScript" dans la rubrique Gestionnaire de sécurité. Avec la version actuelle du logiciel (7), aucun message n'indique si un code Javascript a été bloqué.

Documentation

- http://blog.emsisoft.com/fr/2016/01/01/decouvrez-ransom32-le-premier-rancongiciel-decouvert-dans-javascript/
- http://securitywa.blogspot.fr/2016/02/javascript-ransomware-attack.html
- https://technet.microsoft.com/en-us/library/ee198684.aspx
- https://www.adobe.com/devnet-docs/acrobatetk/tools/AppSec/javascript.html

3 - Rappel des avis émis

Dans la période du 21 au 29 mars 2016, le CERT-FR a émis les publications suivantes :

- CERTFR-2016-AVI-104 : Multiples vulnérabilités dans Moodle (21 mars 2016)
- CERTFR-2016-AVI-105 : Multiples vulnérabilités dans les pilotes Nvidia (21 mars 2016)
- CERTFR-2016-AVI-106 : Multiples vulnérabilités dans les produits Apple (22 mars 2016)
- CERTFR-2016-AVI-107 : Multiples vulnérabilités dans les produits Cisco (24 mars 2016)
- CERTFR-2016-AVI-108 : Vulnérabilité dans Oracle Java (24 mars 2016)
- CERTFR-2016-AVI-109 : Multiples vulnérabilités dans Google Chrome (25 mars 2016)

Gestion détaillée du document

29 mars 2016 version initiale.