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 22 mai 2017
N° CERTFR-2017-ACT-021

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2017-ACT-021

1 - Substitution de bibliothèques

Les documents Vault 7 ont remis d'actualité les risques liés aux substitutions de bibliothèques en environnement Microsoft Windows (DLL). Ce bulletin évoque les causes et solutions envisageables pour réduire les risques associés à ce type d'attaques pour les concepteurs d'application.

Contexte

Dans les documents mise à disposition par Wikileak sous le titre *Vault 7*, il est question du piégeage de l'application Notepad++.

Le rédacteur évoque la substitution d'une version piégée à la bibliothèque Scintilla chargée par le programme au démarrage.

Comme l'explique l'auteur du logiciel dans une publications en ligne, dans ce cas particulier l'attaquant doit déjà disposer de droits suffisants à la modification de l'exécutable pour réaliser sa substitution et l'intérêt de l'opération semble relativement mineur.(
https://notepad-plus-plus.org/news/notepad-7.3.3-fix-cia-hacking-issue.html)

Néanmoins, l'exploitation du mécanisme de chargement de librairies extérieures est suffisamment fréquente pour justifier une revue des méthodes d'attaque et surtout des défenses.

Bien que cette famille de manipulations ne soit pas spécifique à l'environnement Ms Windows, le présent bulletin se concentrera sur ce système, et en particulier, sur les mesures que le développeur d'une application peut mettre en oeuvre pour s'en protéger.

L'attaque

Objectifs de l'attaquant

L'attaquant désire qu'une application légitime charge une bibliothèque illégitime dans son processus.

Les objectifs sont généralement de 2 ordres :

- persistance : permettre à un maliciel d'être exécuté au redémarrage de l'application tout en ne créant pas de nouveau fichier `.EXE` sur le système compromis. Le code ainsi déposé peut rester longtemps sans être détecté : son nom étant légitime et une revue rapide ne détectant pas de nouveau fichier directement exécutable ;
- escalade de privilège : permettre au code contenu dans la bibliothèque de s'exécuter avec les privilèges, plus élevés, de l'application le chargeant.

Mise en oeuvre de l'attaque

Plusieurs approches sont possibles pour réaliser la substitution :

- exploiter le chemin de recherche des bibliothèques : si le chemin complet du fichier à chargé n'est pas transmis lors de l'appel `LoadLibrary`, utilisé pour charger une DLL, la fonction effectue une recherche dans plusieurs répertoires du fichier demandé. Par défaut sont recherchés:
  1. le répertoire contenant l'application,
  2. le répertoire courant,
  3. les répertoires système,
  4. le répertoire du sous-système 16bits,
  5. le répertoire Windows,
  6. les répertoires contenus dans la variable d'environnement `%PATH%`.

  Si l'attaquant peut écrire dans un de ces répertoires, il peut y créer un fichier .DLL homonyme de celui recherché, et en obtenir le chargement si il est rencontré avant le fichier légitime. Le répertoire « courant » de l'application est fréquemment un bon candidat à ce type de manipulation;
- exploiter un élément de configuration inscriptible par l'attaquant : dans certains cas, le programmeur spécifie le chemin complet de chargement de la bibliothèque dans l'appel à la fonction LoadLibrary, mais utilise pour le constituer une information issue d'une source peu protégée. Ce mécanisme est souvent utilisé par des procédures d'installation pour permettre à l'utilisateur de personnaliser les répertoires contenant le programme. Si un fichier de configuration, ou une clé de base de registre est accessible en écriture à un utilisateur non privilégié, celui-ci peut forcer un chargement d'une bibliothèque sous son contrôle dans un répertoire de son choix;
- écrasement du fichier légitime : Si l'attaquant peut écrire sur le fichier .DLL il peut y substituer directement la version illégitime à celle installée par le programme. Cette approche n'est possible que si l'attaquant dispose de privilèges étendus ou si le répertoire de l'application est inscriptible à un utilisateur non privilégié.

Contre-mesures pour les développeurs

Le sujet ayant déjà été abordé dans le bulletin CERTFR-2015-ACT-008 nous proposons à nos lecteur de s'y référer pour le détail des mesures qu'un administrateur système peut mettre en oeuvre pour diminuer l'exposition de ses systèmes à ce type de manipulation.

Spécifier à l'exécution le chemin de chargement de bibliothèques

Les interfaces de programmation Windows permettent de spécifier le chemin de recherche de différentes façon :

- en spécifiant directement le chemin d'accès complet à la bibliothèque dans l'appel à `LoadLibrary` ou `LoadLibraryEx`. Cette approche fonctionne bien pour des fichiers .DLL installés avec l'application, mais peut poser des problèmes de portabilité si ceux-ci sont distribués indépendemment.
- `SetDLLDirectory` et `AddDllDirectory` permettent de privilégier un chemin de recherche propre à l'application et ainsi d'éviter des recherches trop génériques. En particulier, positionner SetDLLDirectory à la chaine vide supprime le répertoire courant du chemin de recherche. Il est à noter que si les chemins sont erronés, après avoir cherché dans le répertoire d'installation de l'application, puis dans les répertoires explicitement spécifiés, l'appel à LoadLibrary reprendra la séquence de recherche habituelle : répertoires système puis PATH . La fonction SetSearchPathMode est disponible à partir de Windows 7 change l'ordre de recherche des répertoires et positionne le répertoire courant en dernier de la liste;
- Utiliser LoadLibraryEx au lieu de LoadLibrary et utiliser le paramètre dwFlags pour restreindre le chemin de recherche au minimum requis.

A l'installation

Les droits sur les fichiers à l'installation sont très importants. Si l'installation se fait avec des privilèges élevés (cas d'une installation normale), les droits sur les répertoires contenant le programme et ses dépendances devraient être restreints en écriture et modification aux utilisateurs privilégiés (généralement Systeme, TrustedInstaller et le groupe Administrateurs).

Il en est de même pour les informations de configuration affectant les répertoires de recherche de bibliothèques : fichier .ini, clés de base de registre.

Ainsi, la ruche `HKEY_CURRENT_USER` devrait être évitée si l'on ne veut pas qu'un utilisateur non privilégié puisse changer des chemins.

Dans les données en base de registre, une attention particulière devra être portée si des chemins sont utilisés dans la ruche `HKEY_CLASSES_ROOT_KEY` car celle-ci peut agréger des informations venant d'espaces protégés et inscriptibles par un utilisateur normal (`HKEY_LOCAL_MACHINE`)

Voir pour une exploitation de ce type de faille :
http://enigma0x3.net/2016/08/fileless-uac-bypass-using-eventvwr-exe-and-registry-hijacking/

Vérifier les informations *Authenticode* des bibliothèques

Avant de procéder au chargement d'une bibliothèque il est possible, si le fichier est signé de vérifier ses informations *Authenticode*.

Ces informations, peuvent être soit stockées dans le fichier lui même, soit dans les catalogues de fichiers propres à Windows.

La vérification d'une bibliothèque *Authenticode* n'est malheureusement pas faisable en une seule étape.

Il est nécessaire de vérifier :

1. la signature du fichier : ce qui permet de détecter que la signature existe, est valide, que le fichier bibliothèque n'a pas été altéré ;
2. le certificat associé à la signature : afin de s'assurer que le signataire est bien celui attendu.

Vérification de la validité de signature d'une bibliothèque

La première étape est réalisée assez facilement à l'aide de la fonction `WinVerifyTrust`. Le comportement de cette fonction est paramétré par une structure de donnée assez complexe.

Plusieurs points importants sont à noter dans ce paramétrage :

– les paramètres sont différents suivant que la bibliothèque à vérifier est signée par catalogue ou fichier. Il faut positionner le paramètre dwUnionChoice à `WTD_CHOICE_FILE et WTD_CHOICE_CATALOG` et charger le pointeur sur une structure `WINTRUST_FILE_INFO` ou `WINTRUST_CATALOG_INFO` respectivement dans les champs `pFile` et `pCatalog`;

– suivant le positionnement du champ fdwRevocationChecks la vérification de révocation de certificat de signature sera effectué ou pas. Cette opération requiers généralement un accès réseau aux serveurs de révocation de l'autorité de révocation. Si ceux-ci sont injoignables (sur un réseau hors ligne par exemple) la vérification échoue après l'attente d'une expiration de délais de réponse qui peut être assez long.

Un exemple d'usage de cette fonction peut être trouvé dans
https://msdn.microsoft.com/fr-fr/library/windows/desktop/aa382384(v=vs.85).aspx

Vérification de l'identité du signataire de la bibliothèque

Une fois la signature vérifiée, il est possible de vérifier l'identité du signataire du fichier. Cette vérification se fait en vérifiant la correspondance de champs du certificat X509 associé à la signature et des valeurs attendues.

Plusieurs champs peuvent être considérés, mais tous n'offrent pas le même degrés de fiabilité :

– le champ *Subject Name* contient un nom structuré long du propriétaire du certificat. Celui-ci devrait être unique par propriétaire de certificat de signature de code. Cette valeur peut rester stable en cas de changement de certificat;

– le champ *Key-Id* contient un identifiant unique de clé (normalement le condensat de la clé publique contenue dans le certificat. Bien qu'il soit permis, sous certaines politiques de certificat, de renouveler un certificat en conservant un même biclé, il est possible que cet identifiant change lors d'un renouvellement de certificat ;

– Le *Simple Display Name* est un nom abrégé fournit par l'interface de programmation *Authenticode* à fins d'affichage et constitué à partir du *Subject Name*. Il est possible, bien que peu probable, que des certificats de propriétaires différents aient le même *Simple Display Name*.

Il est recommandé de vérifier le certificat signataire à partir de son champ *Subject Name* ou du *Key-Id* plutôt que le *Simple Display Name* plus simple d'utilisation mais moins fiable. Pour extraire les informations de signature d'un fichier la fonction CryptQueryObject est utilisée pour les informations de signature. Celle-ci renvoie une donné de type `CRYPTMSG` qui peut être traitée par la fonction `CryptMsgGetParam` pour obtenir les informations de référence du certificat : référence de l'autorité émettrice et numéro de série. `CertFindCertificateInStore`

utilise ces informations pour récupérer un objet `CERT_CONTEXT` contenant les informations du certificat signataire. Il est possible de lire le champ de sujet du certificat en accédant directement au champ `pCertInfo-Subject` en le décodant à l'aide de `CertNameToStr`. Pour accéder à l'identifiant de clé, la fonction CertGetCertificateContextProperty avec l'option `CERT_KEY_IDENTIFIER_PROP_ID` doit être utilisée pour extraire la séquence d'octet caractérisant l'identifiant de clé. La comparaison de l'un ou des deux champs avec des valeurs connues permet de s'assurer que le certificat signataire est celui connu. Un exemple de l'examen des champs de certificats peut être trouvé à l'adresse :
https://support.microsoft.com/fr-fr/help/323809/how-to-get-information-from-authenticode-signed-executables

Synthèse de la vérification Authenticode

Il peut être regretté qu'un appel unitaire ne soit disponible pour effectuer la vérification *Authenticode* simplement, mais il n'existe, à ce jour, pas de substitut connu aux opérations sus-décrites. Pour synthétiser, les vérifications à effectuer sont, dans l'ordre :

1. vérifier la signature avec WinVerifyTrust;
2. vérifier le certificat signataire avec CryptQueryObject et un critère fort : sujet complet ou identifiant de clé ;
3. charger la bibliothèque en spécifiant son chemin complet.

Recommandations

Le CERT-FR recommande aux développeurs :

1. de positionner les droits les plus restrictifs possibles dans les scripts d'installation ;
2. d'éviter de charger des chemins d'accès depuis des sources inscriptibles par des utilisateurs non privilégiés ;
3. de spécifier des chemins complets de bibliothèques à charger ;
4. d'utiliser l'interface de programmation *Authenticode* pour vérifier les fichiers DLL avant chargement dans les cas ou la bibliothèque est connue et signée.

Références

- Bulletin de février 2015 sur les mesures de prévention des substitutions de bibliothèques : http://www.cert.ssi.gouv.fr/site/CERTFR-2015-ACT-008/
- Recommandations Microsoft :
  - https://support.microsoft.com/fr-fr/help/2389418/secure-loading-of-libraries-to-prevent-dll-preloading-attacks
  - https://msdn.microsoft.com/fr-fr/library/windows/desktop/ff919712(v=vs.85).aspx

2 - Exploit EternalBlue

Microsoft a corrigé en mars 2017 une série de vulnérabilités présentes dans la gestion du protocol SMBv1 (Server Message Block version 1) par Windows. Ces failles ont fait l'objet du correctif de sécurité MS17-010.

Le 14 avril, l'entité se surnommant Shadow Brokers a dévoilé un cadriciel d'exploitation nommé FuzzBunch, ainsi qu'une collection d'exploits, subtilisés au groupe d'attaquants Equation Group. Parmi ces exploits, certains ciblent les vulnérabilités MS17-010, dont l'exploit nommé EternalBlue.

Nature de la vulnérabilité

La vulnérabilité exploitée par EternalBlue se situe dans le pilote srv.sys de Windows, gérant la version 1 du protocole SMB. Un message SMBv1 de type NT Trans Request en provenance d'un client est susceptible de fournir au serveur vulnérable une liste d'attributs étendus de fichier, dont l'un des attributs est corrompu de telle manière à provoquer un débordement d'entier dans le calcul de la taille de la liste.

Ce débordement d'entier provoque lui-même un débordement de tampon sur le tas non-paginé du noyau Windows. La taille et le contenu du débordement sont sous le contrôle de l'attaquant.

Fonctionnement de l'exploit

EternalBlue effectue un "massage" du tas à distance pour préparer l'exploitation du débordement de tampon. Pour ce faire, le client ouvre plusieurs connexions SMBv2 dont chacune provoque l'allocation d'un bloc mémoire sur le tas non-paginé du serveur. Chaque bloc est une cible potentielle destinée à être écrasée.

Notons que l'emploi du protocole SMBv2 lors de cette phase n'a pas de lien direct avec la vulnérabilité exploitée, qui dépend bien de la version 1 du protocole SMB.

Ensuite l'exploit tente de libérer un bloc du tas adjacent à l'un des blocs cibles, et d'y réallouer le tampon vulnérable au débordement. S'il y parvient le contenu du bloc cible suivant le tampon est donc corrompu.

Le bloc cible contient une structure du noyau Windows nommée MDL (Memory Descriptor List) décrivant la destination de futures données arrivant sur la connexion SMBv2 propriétaire du bloc. L'exploit remplace la MDL d'origine par un faux pointant vers une zone mémoire à la fois exécutable et inscriptible à une adresse fixe du noyau, dans le tas de la HAL (Hardware Abstraction Layer). Puis il envoie son code encoquillé (shellcode) sur la connexion; celui-ci est copié dans le tas de la HAL à sa réception.

Finalement, l'exploit interrompt toutes les connexions SMBv2 en cours. À ce stade, un autre mécanisme du noyau intervient: les routines de completion des IRPs (I/O Request Packets) associées au connexions en cours sont invoquées. Le bloc corrompu est un contexte d'IRP, qui est déréférencé par la routine de completion. Celle-ci y récupère un pointeur de fonction, détourné par l'exploit vers le point d'entrée du shellcode dans le tas de la HAL. Le shellcode est ainsi invoqué.

Porte dérobée

Le shellcode noyau recherche dans la mémoire du pilote srv.sys une table de pointeurs vers des fonctions gérant les messages Trans2 du protocole SMB. Il crochète ensuite l'entrée de la table correspondant à la sous-commande SESSION_SETUP, ordinairement non utilisée, et y associe le point d'entrée d'une porte dérobée.

La porte dérobée, nommée DoublePulsar, accepte trois classes de requêtes. Selon la requête reçue, la porte dérobée est capable d'annoncer sa présence, de se désinstaller, ou d'exécuter un code arbitraire en noyau. Cette dernière fonctionnalité est utilisée par le client DoublePulsar pour injecter une DLL dans un processus utilisateur.

Réutilisation de l'exploit

La vulnérabilité exploitée par EternalBlue était présente sur la quasitotalité des systèmes Windows, y compris les versions obsolètes. L'exploit lui-même ne fonctionne pas contre toutes les plate-formes vulnérables, mais il est efficace, par exemple, contre les systèmes Windows 7 et Windows Server 2008 R2 non mis à jour.

Cette efficacité a poussé plusieurs attaquants à réutiliser l'exploit, souvent sous forme d'un simple rejeu de traffic réseau légèrement modifié, une technique qui fonctionne très bien dans le cas présent.

Parmi les cas de réutilisation, on peut citer le ver et rançongiciel WannaCry (plusieurs variantes), ainsi que d'autres maliciels. L'exploit a aussi été porté vers Metasploit.

La grande popularité de cet exploit a poussé Microsoft a publier exceptionnellement des correctifs pour certaines plate-formes obsolètes, notamment Windows XP, Server 2003, et Windows 8.

Conclusion

La technique d'exploitation employée par EternalBlue témoigne du niveau d'expertise élevé du groupe Equation. Malheureusement l'utilisation de l'exploit est à la portée de tous les attaquants, y compris les moins sophistiqués, et la propagation du ver WannaCry suggère que de nombreuses machines demeurent vulnérables.

Outre l'installation des correctifs, et l'abandon obligatoire des systèmes obsolètes, le CERT-FR recommande la désactivation du protocole réseau SMBv1, pour se prémunir non seulement de la faille exploitée par EternalBlue, mais aussi de possibles failles similaires dans le futur.

Le bulletin d'actualité CERTFR-2017-ACT-019 détaille les étapes de la désactivation de ce protocole.

Documentation

https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
http://cert.ssi.gouv.fr/site/CERTFR-2017-ACT-019/index.html

3 - Rappel des avis émis

Dans la période du 15 au 21 mai 2017, le CERT-FR a émis les publications suivantes :

- CERTFR-2017-AVI-153 : Multiples vulnérabilités dans Moodle
- CERTFR-2017-AVI-154 : Multiples vulnérabilités dans Microsoft Windows XP, Windows Server 2003 et Windows 8
- CERTFR-2017-AVI-155 : Multiples vulnérabilités dans les produits Apple
- CERTFR-2017-AVI-156 : Multiples vulnérabilités dans le noyau Linux de Suse
- CERTFR-2017-AVI-157 : Multiples vulnérabilités dans WordPress
- CERTFR-2017-AVI-158 : Multiples vulnérabilités dans le noyau Linux d'Ubuntu
- CERTFR-2017-AVI-159 : Vulnérabilité dans Joomla!
- CERTFR-2017-AVI-160 : Multiples vulnérabilités dans les produits Cisco
- CERTFR-2017-AVI-161 : Multiples vulnérabilités dans VMware Workstation

Gestion détaillée du document

22 mai 2017 version initiale.
24 mai 2017 correction orthographe.