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 06 mars 2017
N° CERTFR-2017-ACT-010

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2017-ACT-010

1 - Utilisation de la méthode d'injection de code "AtomBombing" dans Dridex

La version 4 du code malveillant Dridex implémente partiellement une méthode d'injection originale publiée en octobre 2016.

Rappels sur les méthodes d'injection de code traditionnelles

Les codes malveillants injectent souvent du code dans la mémoire d'un autre processus, que ce soit pour contourner des mécanismes de sécurité en ciblant des processus considérés comme sûrs par les solutions de sécurité, ou pour accéder à des données propres à un processus. On peut citer par exemple les attaques de type "Man-in-the-Browser" où pour modifier des pages web à la volée, le code malveillant doit s'exécuter dans le contexte du navigateur.

Les méthodes traditionnelles d'injection de code passent par l'allocation d'une zone mémoire exécutable dans le processus cible, généralement via l'API VirtualAllocEx, puis par la recopie du code à exécuter au travers de la fonction WriteProcessMemory ou d'une de ses variantes, et enfin par la création d'un thread distant dont le point d'entrée correspond à la zone de mémoire allouée initialement, avec un appel à la fonction CreateRemoteThread.

Présentation de la méthode "AtomBombing"

Les solutions de sécurité étant sensibilisées au détournement de ces fonctions à des fins malveillantes, leurs appels sont particulièrement surveillés. La méthode "AtomBombing", publiée par le blog `Breaking Malware`, permet de s'en affranchir en utilisant notamment la structure Atom, dont l'objectif initial est de stocker des chaînes de caractères qui peuvent être partagées entre processus à l'aide de leur identifiant.

Cet objectif est ici détourné pour injecter du code à une adresse arbitraire, en créant un Atom contenant ce code via la fonction GlobalAddAtom, puis en provoquant l'appel de la fonction GlobalGetAtomName avec l'adresse mémoire cible en paramètre. Cependant, ce deuxième appel doit être fait par le processus dans lequel on souhaite injecter notre code. Les auteurs de la méthode ont trouvé qu'il était possible de réaliser cet appel avec la fonction NtQueueApcThread avec en paramètre :

- un handle vers un thread de notre processus cible,
- l'adresse de la fonction que l'on souhaite appeler (ici GlobalGetAtomName),
- 3 arguments qui seront passés à cette fonction (nombre d'arguments que prend GlobalGetAtomName).

L'appel à ces trois fonctions remplace l'appel à WriteProcessMemory dans les méthodes d'injection traditionnelles.

La première étape de l'injection par AtomBombing est d'utiliser cet enchaînement de fonctions pour recopier la charge malveillante vers la fin de la section "data" de la DLL kernelbase.dll, non utilisée. Le code ainsi copié ne peut toutefois pas être exécuté directement à cause de la protection mémoire des pages où il se trouve. Les auteurs ont choisi de contourner ce problème en forgeant une chaîne ROP qui alloue une zone mémoire exécutable, recopie le code injecté de kernelbase.dll vers cette zone et détourne le flux d'exécution à l'adresse de cette zone mémoire.

La dernière étape est de provoquer l'exécution de cette chaîne ROP via un appel à la fonction NtSetContext-Thread. Cette fonction sera appelée par le processus cible en utilisant une fois de plus la fonction NtQueueApc-Thread.

Utilisation de la méthode dans Dridex

Dans la version 4 de Dridex, les auteurs du malware ont utilisé une variante de la méthode AtomBombing. Ils utilisent bien la suite de 3 méthodes GlobalAddAtom, GlobalGetAtomName et NtQueueApcThread pour écrire du code à une adresse arbitraire, mais ils diffèrent pour la partie allocation de la mémoire et exécution du code injecté.

Dridex utilise des zones mémoires inscriptibles de ntdll qu'il va remettre à zéro en provoquant un appel à memset via l'API NtQueueApcThread. Il recopie son code avec la méthode déjà décrite impliquant les structures Atom.

Une fois son code recopié, il est nécessaire de rendre ces zones mémoires exécutables, ce que Dridex fait en appelant la méthode NtProtectVirtualMemory.

Enfin, pour provoquer l'exécution de ce code injecté, Dridex a choisi de modifier le début du code d'une fonction légitime afin de sauter vers le code injecté. Dridex place son hook sur la fonction GlobalGetAtomA et l'appelle juste après.

On note donc que les auteurs de Dridex effectuent une veille technologique régulière et s'approprient les nouvelles méthodes publiées en les adaptant en fonction de leurs propres contraintes.

Recommandation

Cette nouvelle méthode montre que des fonctions standards de l'API Windows peuvent être détournées de leur objectif initial pour effectuer de l'injection de code. Il convient donc de vérifier les appels aux fonctions GlobalAddAtom, GlobalGetAtomName et NtQueueApcThread dans le cadre d'analyse de codes malveillants.

Documentation

- *AtomBombing: Brand New Code Injection for Windows*, 27 octobre 2016 :
  https://breakingmalware.com/injection-techniques/atombombing-brand-new-code-injection-for-windows/
- *Dridex's Cold War: Enter AtomBombing*, 28 février 2017 :
  https://securityintelligence.com/dridexs-cold-war-enter-atombombing/

2 - Rappel des avis émis

Dans la période du 27 février au 05 mars 2017, le CERT-FR a émis les publications suivantes :

- CERTFR-2017-ALE-003 : Vulnérabilité dans les navigateurs Microsoft
- CERTFR-2017-AVI-060 : Multiples vulnérabilités dans le noyau Linux de SUSE
- CERTFR-2017-AVI-061 : Vulnérabilité dans Cisco NetFlow Generation Appliance
- CERTFR-2017-AVI-062 : Vulnérabilité dans SCADA Siemens SINUMERIK Integrate
- CERTFR-2017-AVI-063 : Multiples vulnérabilités dans SCADA les produits Schneider
- CERTFR-2017-AVI-064 : Vulnérabilité dans VMware Horizon DaaS

Gestion détaillée du document

06 mars 2017 version initiale.