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 04 mai 2015
N° CERTFR-2015-ACT-018

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2015-ACT-018

1 - Vulnérabilité de la pile HTTP de Windows

Une vulnérabilité de type « dépassement d'entier » affectant le pilote `HTTP.sys` de Windows a été corrigée par le bulletin MS15-034 de Microsoft. Selon l'éditeur, l'exploitation de celle-ci peut provoquer un déni de service sur la machine cible, voire une exécution de code arbitraire.

Identification de la vulnérabilité

L'en-tête HTTP `Range` permet à un client de ne télécharger qu'une partie d'une ressource mise à disposition par un serveur. Par exemple, pour la valeur suivante : `Range: bytes=5-100`, le serveur va renvoyer les données de la ressource du 6ème au 101ème octet.

Cet en-tête est utilisé par exemple dans le cas d'un téléchargement corrompu à la suite duquel l'utilisateur aura besoin de récupérer la fin d'un fichier.

Lorsqu'une requête HTTP est effectuée plus d'une fois vers une même ressource, le service HTTP place la réponse de la requête en cache dans le noyau, ceci à des fins de performance. La fonction `UlpParseRange()` est chargée de parser l'en-tête HTTP `Range` et fait appel à la fonction `UlAdjustRangeToContentSize()` qui vérifie que la plage de la donnée demandée ne dépasse pas la taille de celle-ci.

Voici l'algorithme utilisé pour la vérification :

```
if(low_range != ULONG_MAX)
{
    if(low_range < rsrc_size)
    {
        val = (high_range-low_range) + 1;
        if(val != ULONG_MAX)
        {
            tmp = val + low_range;
            if(tmp < rsrc_size)
                return val;
        }
    }
}
```

La vulnérabilité mentionnée précédemment se situe dans cette fonction.

En effet, si l'en-tête envoyé vaut `Range: bytes=3-0xFFFFFFFFFFFFFFFF`, les différentes vérifications sont contournées à cause d'un dépassement d'entier. La variable `tmp` à la ligne 8 vaudra 0, ce qui est bien inférieur à la taille de la ressource demandée. Au final, cette fonction renverra à la ligne 9 la valeur `0xFFFFFFFFFFFFFFFF-3+1`, qui sera la taille utilisée pour créer le cache.

Déclenchement de la vulnérabilité

Dans le cas nominal, le client va envoyer une requête HTTP qui sera interceptée par le pilote `HTTP.sys`. S'il y a eu plus d'une requête vers une même ressource, le serveur HTTP (IIS par exemple) enverra un IOCTL `UlSendHttpResponseIoctl` avec sa réponse au pilote. Ce dernier stockera la réponse en cache via une structure MDL (liste de descripteurs mémoire permettant de décrire l'association faite entre un bloc de mémoire virtuel et une liste de pages dans la mémoire physique) afin de fournir un bloc de mémoire accessible depuis l'espace utilisateur. Le client pourra donc accéder directement à cette structure pour récupérer la réponse.

La fonction `UlpBuildCacheEntry()` permet de construire ce cache.

Elle fera entre autre appel à la fonction `UlBuildFastRangeCacheMdlChain()` qui fera des appels successifs à `IoBuildPartialMdl()` permettant de segmenter un bloc de mémoire trop gros en associant une partie de l'intervalle décrit par une structure MDL dans une autre.

A la fin de l'en-tête d'une structure MDL se trouve un tableau contenant le numéro des pages physiques. Ces pages correspondent aux adresses virtuelles du bloc de mémoire contenant la réponse du serveur :

```
kd> db fffffa80006e9a70+0x30 L200

fffffa80`006e9aa0 25 ed 01 00 00 00 00-26 ed 01 00 00 00 00 %.......&.......
fffffa80`006e9ab0 27 ed 01 00 00 00 00-28 ed 01 00 00 00 00 '.......(.......
fffffa80`006e9ac0 29 ed 01 00 00 00 00-2a ed 01 00 00 00 00 ).......*.......
fffffa80`006e9ad0 2b ed 01 00 00 00 00-2c ed 01 00 00 00 00 +.......,.......
fffffa80`006e9ae0 2d ed 01 00 00 00 00-2e ed 01 00 00 00 00 -...............
fffffa80`006e9af0 2f ed 01 00 00 00 00-30 ed 01 00 00 00 00 /.......0.......
fffffa80`006e9b00 31 ed 01 00 00 00 00-32 ed 01 00 00 00 00 1.......2.......
fffffa80`006e9b10 33 ed 01 00 00 00 00-34 ed 01 00 00 00 00 3.......4.......
[...]
```

La fonction `IoBuildPartialMdl()` prend plusieurs arguments dont un argument `Length` correspondant à la taille en octet des données à associer à la structure MDL cible. Lors d'une tentative d'exploitation de la vulnérabilité, cette valeur est celle renvoyée par la fonction vulnérable `UlAdjustRangeToContentSize()` décrite plus haut convertie sur 32 bits.

Cette fonction met à jour les champs de la structure MDL et copie les numéros de pages physiques de la structure source vers celle ciblée :

```
kd> db fffffa800162bd50+0x30
fffffa80`0162bd80 25 ed 01 00 00 00 00-26 ed 01 00 00 00 00 %.......&.......
fffffa80`0162bd90 27 ed 01 00 00 00 00-28 ed 01 00 00 00 00 '.......(.......
fffffa80`0162bda0 29 ed 01 00 00 00 00-00 00 00 00 00 00 00 ).............. 
fffffa80`0162bdb0 00 00 00 00 08 07 00 00-00 00 00 00 00 00 00 ..............
fffffa80`0162bdc0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 ..............
fffffa80`0162bdd0 00 00 00 00 00 01 01 00-00 00 01 01 00 00 00 ..............
fffffa80`0162bde0 00 00 00 00 02 00 00 00-00 00 00 00 0a 00 00 ..............
fffffa80`0162bdf0 30 75 00 00 e8 03 00 00-c0 27 09 00 03 00 00 00 0u.......'......
```

Les premières copies sont bien réalisées mais cette opération se répète un certain nombre de fois en fonction de l'argument `Length`. Lorsque cette taille est trop grande, comme c'est le cas ici, des données aléatoires de la mémoire vont être aussi copiées et s'il n'y a pas assez de mémoire, la fonction va tenter de lire des données dans de la mémoire non allouée, ce qui provoquera un plantage.

```
BugCheck 50, {fffffa8001a00000, 1, fffff8000167a651, 2}
Probably caused by~: HTTP.sys ( HTTP!UlBuildFastRangeCacheMdlChain+3f7 )
kd> db fffffa8001a00000-10
fffffa80`019ffff0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
fffffa80`01a00000 ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ?? ????????????????
fffffa80`01a00010 ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ?? ????????????????
fffffa80`01a00020 ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ?? ????????????????
fffffa80`01a00030 ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ?? ????????????????
fffffa80`01a00040 ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ?? ????????????????
fffffa80`01a00050 ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ?? ????????????????
fffffa80`01a00060 ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ?? ????????????????
```

Il est donc possible de provoquer un déni de service à distance facilement et, dans le cas de requêtes avec plusieurs en-têtes `Range`, provoquer des fuites de mémoire ce qui peut porter atteinte à la confidentialité des données.

Recommandations

Il est recommandé d'appliquer le correctif de sécurité fourni par Microsoft qui corrige le problème.

Il faut noter que le service HTTP peut être utilisé par différentes applications et services : c'est le cas pour IIS mais aussi pour le service SSDP, UPnP, etc.

La commande `netsh http show servicestate` permet de vérifier la liste des ressources utilisant le service HTTP.

Documentation

- Correctif :
  https://technet.microsoft.com/library/security/MS15-034
- Bulletin d'actualité CERTFR-2015-ACT-016 du 20 avril 2015 :
  http://cert.ssi.gouv.fr/site/CERTFR-2015-ACT-016/index.html

2 - Avantages et limites d'Authenticode Windows (suite)

Révocation et horodatage

Révocation d'un certificat

La révocation de certificat d'un programme peut être un problème coûteux et chronophage. Un éditeur ne peut révoquer un binaire spécifique. En effet, le modèle nécessite la révocation du certificat et donc l'inhibition de tous les binaires signés par ce certificat.

Dans l'hypothèse d'une vulnérabilité impactant un de ses produits, l'éditeur du logiciel devrait :

- obtenir un nouveau certificat ;
- mettre à jour chacun de ses autres produits ;
- corriger la vulnérabilité sur le produit impacté ;
- publier le correctif nouvellement signé du produit vulnérable ;
- faire révoquer son certificat.

Révocation d'un binaire

Il est possible, pour Microsoft, de mettre en liste noire certains condensats de binaires. Cette méthode permet d'éviter l'utilisation d'une version de logiciel vulnérable. Cependant, la mise en liste noire d'un binaire qui s'avère vital à un système de production peut entraîner le dysfonctionnement, voire l'arrêt de celui-ci. C'est pourquoi cette méthode est peu utilisée par Microsoft.

Horodatage de certificat

Authenticode supporte l'horodatage des signatures : ce mécanisme permet, en cas révocation d'un certificat suite à un vol de clef privée notamment, d'être capable d'établir de façon fiable que le binaire a été signé avant la révocation ou l'expiration du certificat.

La fiabilité de cet horodatage repose sur des serveurs d'horodatage propre à chaque autorité de certification. De ce fait, la signature du binaire sera considérée comme valide car ayant été effectuée avant la date de révocation ou d'expiration du certificat.

Conclusion

Authenticode est une implémentation efficace pour imposer des contraintes de signature mais également pour que toute exécution de binaire non-signé ou mal signé ne passe pas inaperçue aux yeux de l'utilisateur. Néanmoins, cette technologie n'élimine pas tous les risques de compromission, comme présenté dans l'article du 23 mars. Le cas échéant, cela accroît la discrétion d'un acteur malveillant. Un autre angle d'attaque est l'algorithme de condensat. Là où MD5 n'est plus fiable, en témoigne l'attaque par collision utilisée dans le cadre de Flame, on peut se questionner quant aux autres algorithmes, notamment SHA-1.

Documentation

– Avantages et limites d'Authenticode Windows, première partie :
http://www.cert.ssi.gouv.fr/site/CERTFR-2015-ACT-012/index.html

3 - Politique d'obsolescence de SHA-1

Depuis 2005, des attaques théoriques de plus en plus efficaces contre la fonction de hachage SHA-1 ont été publiées. Pour éviter de se retrouver dans la même situation que pour la fonction de hachage MD5 (voir article "MD5 Considered Harmful Today Creating a rogue CA certificate" [1] dans laquelle des chercheurs montaient une attaque complète reposant sur une collision MD5), les éditeurs de navigateurs, et les autorités de certification ont proposé de désactiver SHA-1 de manière graduelle, avec une fin de service programmée fin 2016. Cette décision a été entérinée par le CA/B Forum en octobre dernier [2].

Parmi les mesures concrètes mises en place, Google a annoncé une série de mesures consistant à afficher des avertissements de plus en plus visibles pour les certificats expirant au-delà du 31 décembre 2015 [3] (ce qui correspond à une politique plus sévère que la décision prise par le CA/B Forum). Microsoft, qui avait été à l'initiative des actions pour rendre SHA-1 obsolète dispose d'une page décrivant la politique mise en oeuvre [4]. Il en est de même pour Mozilla [5].

Les responsables de site mettant en oeuvre TLS sont donc fortement encouragés à vérifier qu'ils ne disposent pas de certificat utilisant SHA-1 dont la durée de vie dépasse fin 2015. Si c'est le cas, il leur faudra faire renouveler leur certificat en utilisant une fonction de hachage plus moderne telle que SHA-256. Cela concerne évidemment les serveurs HTTPS, mais aussi les autres services (par exemple SMTPS et IMAPS) puisque la politique devrait également être appliquée à ces protocoles en pratique (puisque les implémentations TLS sont souvent partagées).

Documentation

1 MD5 Considered Harmful Today Creating a rogue CA certificate :
https://www.win.tue.nl/hashclash/rogue-ca/downloads/md5-collisions-1.0.pdf

2 CA/B Forum, SHA1 Sunset :
https://cabforum.org/2014/10/16/ballot-118-sha-1-sunset/

3 Google, SHA1 Sunset :
http://googleonlinesecurity.blogspot.fr/2014/09/gradually-sunsetting-sha-1.html

4 Microsoft, SHA1 Sunset :
http://social.technet.microsoft.com/wiki/contents/articles/1760.windows-root-certificate-program-technical-requirements-version-2-0.aspx.

5 Mozilla, SHA1 Sunset :
https://blog.mozilla.org/security/2014/09/23/phasing-out-certificates-with-sha-1-based-signature-algorithms/.

4 - Rappel des avis émis

Dans la période du 27 avril au 02 mai 2015, le CERT-FR a émis les publications suivantes :

– CERTFR-2015-AVI-191 : Vulnérabilité dans IBM Storwize
– CERTFR-2015-AVI-192 : Vulnérabilité dans OpenOffice
– CERTFR-2015-AVI-193 : Vulnérabilité dans WordPress
– CERTFR-2015-AVI-194 : Vulnérabilité dans IBM Websphere
– CERTFR-2015-AVI-195 : Multiples vulnérabilités dans Google Chrome
– CERTFR-2015-AVI-196 : Vulnérabilité dans Cisco IOS XE Software
– CERTFR-2015-AVI-197 : Vulnérabilité dans Huawei E587 Mobile WiFi
– CERTFR-2015-AVI-198 : Multiples vulnérabilités dans le noyau Linux

Gestion détaillée du document

04 mai 2015 version initiale.