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 25 juillet 2016
N° CERTFR-2016-ACT-030

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2016-ACT-030

1 - Protections Intel CET et RAP

Le bulletin d'actualité CERTFR-2016-ACT-027 présentait les principes du mécanisme CET ("Control-flow Enforcement Technology") prévu par Intel. Ce mécanisme devrait à terme fournir une implantation matérielle de la vérification d'intégrité du flot de contrôle des programmes.

D'autres solutions, logicielles, avaient précédé l'initiative d'Intel, comme le mécanisme RAP proposé par l'équipe PaX Team, décrit dans le bulletin d'actualité CERTFR-2015-ACT-047.

Il est intéressant de comparer ces deux mécanismes, afin d'apprécier leurs forces et faiblesses vis-à-vis de différentes voies d'attaques.

Vérification des branchements indirects

La plupart des programmes, et en particulier ceux écrits dans les langages orientés objet tels que C++, font usage de branches indirectes, par exemple pour appeler une fonction à travers un pointeur ou invoquer une méthode virtuelle d'un objet.

L'exploitation des vulnérabilités de type dépassement de tampon sur le tas, ou utilisation-après-libération, repose fréquemment sur l'usurpation d'un pointeur de fonction ou de méthode, qui se voit remplacé par une destination illégitime tel que l'adresse d'un gadget ROP.

Une facette de la vérification d'intégrité du flot de contrôle consiste donc en la protection des branches indirectes (appels ou sauts) pour s'assurer que leur destination est bien légitime.

A minima, la destination d'une branche indirecte devrait être un point d'entrée de fonction (interne au programme, importée d'une bibliothèque externe, ou éventuellement produite à la volée par un compilateur JIT) ou l'adresse d'un bloc de code prévu comme destination d'un saut indirect, tel qu'une directive `switch` en C.

CET propose précisément une implantation matérielle de cette approche, par le biais de l'instruction `ENDBRANCH`. Les branches indirectes font entrer le processeur dans un état `WAIT_FOR_ENDBRANCH`, qui impose que l'instruction suivante soit de type `ENDBRANCH`, faute de quoi une exception est levée.

Ce mode de vérification est à la fois très contraignant, puisque seul un sous-ensemble restreint de blocs de code, ceux débutant par `ENDBRANCH`, peuvent être cibles de branches indirectes, mais aussi très simplistes, dans la mesure où tous ces blocs se confondent en seule classe.

En effet, on pourrait envisager d'usurper un pointeur de fonction et de le rediriger vers une fonction dont le prototype est différent, telle une fonction n'acceptant pas le même nombre d'arguments, de façon à déphaser la pile. Cette manipulation serait autorisée par CET.

Selon le contexte précis de l'application vulnérable, ce contrôle simpliste des sites de destination pourrait donc laisser la porte ouverte à certaines attaques.

Notons aussi qu'Intel a prévu certaines exceptions au mécanisme de contrôle des branches. Ainsi, l'emploi d'un préfixe permet de désactiver le passage à l'état `WAIT_FOR_ENDBRANCH` à l'issue de branches vers un registre. De plus un tableau de bits ("Legacy Bitmap") permet d'autoriser les branches indirectes vers des pages "legacy"

dont les points d'entrée sont dépourvus d'instruction `ENDBRANCH`, et ce par souci de rétro-compatibilité avec des bibliothèques existantes.

Comme pour l'ASLR, le mécanisme CET devrait progressivement être adopté par une majorité des modules Windows. Quelques angles morts subsisteront pendant un certain temps.

Par contraste, le mécanisme ICTP ("Indirect Control Transfer Protection") de RAP est prévu pour fonctionner en mode "tout-ou-rien". Il est beaucoup plus précis dans sa classification des cibles de branches. Il distingue les sites de destination des appels indirects en prenant en compte le prototype de la fonction appelée. Un condensat placé juste avant le point d'entrée de la fonction indique la classe à laquelle la fonction appartient, et le code au site d'appel compare ce condensat à la valeur attendue.

Vérification des retours de fonctions

Le mécanisme ICTP de RAP s'étend aux retours de fonctions. Avant de retourner à l'adresse située au sommet de la pile, chaque fonction vérifie qu'un condensat situé au site de retour (qui est normalement aussi le site d'appel) concorde avec son propre type.

De plus, RAP compare les valeurs de l'adresse de retour à l'entrée et à la sortie d'une même fonction, pour vérifier qu'elle n'a pas été modifiée entre temps. La valeur d'entrée est stockée dans un registre témoin protégé par un masque aléatoire.

Dans le pire des cas, une fuite mémoire pourrait révéler la valeur du masque et permettre de forger à la fois une adresse de retour et le contenu du registre témoin sauvé sur la pile. Les modalités exactes d'une telle attaque dépendraient de l'application ciblée, mais on ne peut l'exclure.

Ce (petit) point faible de RAP tient à l'accès en écriture de la pile. L'intégrité de l'adresse de retour dépend d'une information redondante la concernant, or cette information est aussi stockée sur la pile, et par conséquent aussi ré-inscriptible.

Dans ce domaine, le mécanisme CET d'Intel est supérieur, dans le sens où la pile fantôme est non-inscriptible. Les instructions d'appels de fonctions y écrivent implicitement, en dépit de la protection mémoire indiquée dans les tables de pages. Seule une implantation matérielle pouvait offrir cette possibilité.

Détails sur la pile fantôme

A priori, la pile fantôme de CET offre une excellente sécurité. À l'issue d'une série d'appels de fonctions, elle assure que les retours qui suivent aboutissent bien aux sites de retours attendus. Par conséquent, l'exploitation d'un dépassement de tampon sur la pile par simple remplacement d'une adresse de retour est exclue.

Pour rappel, le principe du ROP ("Return Oriented Programming") est d'aiguiller le flot d'exécution du programme vers une chaîne de "gadgets" (fragments de code) reliés par des instructions `RET`.

En présence de la pile fantôme, et si l'on suppose l'intégrité de la pile fantôme, le ROP deviendrait impossible, puisque les adresses des gadgets seraient invalides.

Cependant, le ROP n'est pas la seule technique d'exploitation permettant de convertir une corruption mémoire en exécution de code arbitraire. Une technique très proche, le JOP ("Jump Oriented Programming") consiste à relier des gadgets finissant par des sauts indirects. Or nous avons expliqué que la vérification des branches indirectes dans CET laissait quelques libertés.

D'autres techniques s'affranchissent de l'utilisation de gadgets, et aboutissent à l'exécution de code arbitraire par l'action sur des "points d'acupuncture" judicieusement choisis dans l'application cible.

Même en ce qui concerne ROP, l'hypothèse de l'intégrité de la pile fantôme doit être vérifiée dans le contexte d'une application cible particulière. D'une part, la pile fantôme ne doit pas être accessible par le biais d'une adresse virtuelle marquée comme inscriptible. C'est la responsabilité du système de s'en assurer. D'autre part, elle ne doit pas être manipulable indirectement pas certaines instructions introduites avec CET.

En effet, Intel a prévu des instructions de manipulation de la pile fantôme, telles que `RDSSP` pour lire le pointeur de pile fantôme, et `WRSS` pour écrire dans la pile fantôme. Cette dernière instruction peut être rendue privilégiée (c'est-à-dire interdite en mode utilisateur) en configurant un MSR ("Model Specific Register") du processeur.

Ces instructions sont probablement conçues pour permettre l'implantation de mécanismes comme les exceptions SEH (sous Windows) ou C++, où le flot de contrôle du programme peut légitimement "remonter" la pile d'appels sans effectuer tous les retours de fonctions attendus.

En définitive, c'est la chaîne complète de l'application, du compilateur, des bibliothèques, de l'environnement d'exécution ("Run-Time"), et du système d'exploitation, qui déterminera l'efficacité de CET, même si la base en est matérielle.

Documentation

- Bulletin d'actualité CERTFR-2016-ACT-027 (article sur CET)
  http://cert.ssi.gouv.fr/site/CERTFR-2016-ACT-027/
- Bulletin d'actualité CERTFR-2015-ACT-046 (article sur RAP)
  http://cert.ssi.gouv.fr/site/CERTFR-2015-ACT-047/
- Réaction de PaX Team à l'annonce d'Intel
  https://forums.grsecurity.net/viewtopic.php?f=7&t=4490

2 - Vulnérabilité dans Wget

Description

Une vulnérabilité affectant wget en versions 1.17 et antérieure a récemment été publiée sous l'identifiant CVE-2016-4971. Celle-ci concerne la gestion des noms de fichiers téléchargés par wget qui, en raison de la vulnérabilité, est imprévisible et peut entrainer l'exécution de commandes arbitraires exécutées avec les mêmes droits et privilèges que ceux de l'utilisateur exécutant wget. Les grandes distributions ont mis à jour les packages fournissant la commande wget. Il est donc conseillé de rapidement mettre à jour les machines où wget est installé.

Détails techniques

Pour illustrer la vulnérabilité, voici la sortie de wget lors d'une redirection du protocole HTTP vers une ressource disponible *via* ce même protocole :

```
$ wget http://127.0.0.1/simple_fichier.txt
--2016-07-06 13:36:24-- http://127.0.0.1/simple_fichier.txt
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://127.0.0.1/.bash_logout [following]
--2016-07-06 13:36:24-- http://127.0.0.1/.bash_logout
Reusing existing connection to 127.0.0.1:80.
HTTP request sent, awaiting response... 200 OK
Length: 5 [text/plain]
Saving to: 'simple_fichier.txt'
simple_fichier.txt 100%[==================>]

5 --.-KB/s

in 0s

2016-07-06 13:36:24 (564 KB/s) - 'simple_fichier.txt' saved [5/5]
```

Ici, le fichier `simple_fichier.txt` est créé, même si la ressource réellement récupérée s'appelle `.bash_logout`. Si la redirection utilise le protocole FTP, la même commande wget donne la sortie suivante :

```
$ wget http://127.0.0.1/simple_fichier.txt
--2016-07-06 13:36:55-- http://127.0.0.1/simple_fichier.txt
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: ftp://127.0.0.1/.bash_logout [following]
--2016-07-06 13:36:55-- ftp://127.0.0.1/.bash_logout
=> '.bash_logout'
Connecting to 127.0.0.1:21... connected.
Logging in as anonymous ... Logged in!
> SYST ... done. > PWD ... done.
> TYPE I ... done. > CWD not needed.
> SIZE .bash_logout ... 7
> PASV ... done. > RETR .bash_logout ... done.
Length: 7 (unauthoritative)
.bash_logout
```

```
100%[==================>]

7 --.-KB/s

in 0s

2016-07-06 13:36:55 (666 KB/s) - '.bash_logout saved [7]
```

Ici, c'est le fichier `.bash_logout` qui est créé, et non plus `simple_fichier.txt`. Si wget est exécuté dans le répertoire personnel (le *home* d'un utilisateur, les commandes contenues dans le fichier `.bash_logout` seront exécutées à la déconnexion dudit utilisateur. Il est bien sûr possible de trouver d'autres façons d'exploiter ce problème de nommage, notamment en créant des fichiers de configuration pour toutes les commandes usuellement utilisées (dont wget lui-même, qui peut être configuré *via* le fichier .wgetrc). Un autre cas d'exploitation serait un wget exécuté dans un répertoire qui permet l'exécution de fichiers interprétés sur une interface Web, type PHP. Par ailleurs, l'entête `User-Agent` d'une requête HTTP émise par wget contient la version de wget :

```
GET /simple_fichier.txt HTTP/1.1
User-Agent: Wget/1.17.1 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 127.0.0.1
Connection: Keep-Alive
```

Un attaquant qui voudrait se dissimuler pourrait alors viser spécifiquement les versions vulnérables de wget et transmettre un fichier légitime dans le cas où l'exploitation n'est pas possible.

Résumé des conditions d'exploitation

Entendu que la version de wget est vulnérable, il faut que les conditions suivantes soient également remplies :

- la commande wget est lancée avec pour objectif de récupérer un fichier sur un serveur contrôlé par un attaquant (le cas d'un attaquant cherchant à s'étendre sur un réseau partiellement compromis n'est pas à négliger) ;
- le fichier n'existe déjà pas, sinon wget rajoute une extension au fichier ;
- la commande wget est lancée depuis un dossier utilisable pour développer un scénario d'exploitation.

Documentation

- Communiqué de mise à jour de wget (anglais) :
  https://lists.gnu.org/archive/html/info-gnu/2016-06/msg00004.html
- Description originale de l'attaque (anglais) :
  https://blogs.securiteam.com/index.php/archives/2701

3 - Rappel des avis émis

Dans la période du 18 au 24 juillet 2016, le CERT-FR a émis les publications suivantes :

- CERTFR-2016-AVI-239 : Multiples vulnérabilités dans les produits Apple
- CERTFR-2016-AVI-240 : Vulnérabilité dans Drupal
- CERTFR-2016-AVI-241 : Multiples vulnérabilités dans Moodle
- CERTFR-2016-AVI-242 : Multiples vulnérabilités dans Oracle Database Server
- CERTFR-2016-AVI-243 : Multiples vulnérabilités dans Oracle Java SE
- CERTFR-2016-AVI-244 : Multiples vulnérabilités dans Oracle Sun Systems Products Suite
- CERTFR-2016-AVI-245 : Multiples vulnérabilités dans Oracle Linux and Virtualization
- CERTFR-2016-AVI-246 : Multiples vulnérabilités dans Oracle MySQL
- CERTFR-2016-AVI-247 : Multiples vulnérabilités dans Google Chrome
- CERTFR-2016-AVI-248 : Vulnérabilité dans Cisco Unified Computing System Performance Manager
- CERTFR-2016-AVI-249 : Vulnérabilité dans SCADA Schneider SoMachine HVAC Programming SW

Gestion détaillée du document

25 juillet 2016 version initiale.

Conditions d'utilisation de ce document :	http://cert.ssi.gouv.fr/cert-fr/apropos.html
Dernière version de ce document :	http://cert.ssi.gouv.fr/site/CERTFR-2016-ACT-030