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 24 novembre 2015
N° CERTFR-2015-ACT-047

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2015-ACT-047

1 - Mitigations RAP et ICTP dans Linux

L'équipe PaX Team, à l'origine du mécanisme de sécurité PaX, a présenté à la conférence H2HC 2015 une nouvelle série de protections destinées à empêcher l'exploitation de certaines classes de bogues sous Linux.

Ces protections sont implantées dans un greffon du compilateur en source ouverte GCC et ne nécessitent qu'une recompilation avec les bons paramètres pour s'appliquer à un code source existant.

RAP

Une première protection dénommée RAP ("Return Address Protection") est une variante de l'idée de biscuit de pile ("stack cookie"). Il s'agit de vérifier que l'adresse de retour d'une fonction n'est pas altérée lorsque celle-ci s'exécute, ce qui pourrait arriver en cas de vulnérabilité de type dépassement de tampon.

Pour ce faire, le prologue de fonction copie l'adresse de retour depuis la pile vers un registre témoin et lui applique un masque disjonctif (XOR). Lors de l'épilogue, avant d'effectuer son retour, la fonction répète la même opération et compare le résultat avec celui calculé lors du prologue. Si les deux valeurs ne sont pas identiques, cela indique une tentative de corruption de l'adresse de retour sur la pile et l'exécution du programme est avortée.

Le masque disjonctif (le "RAP cookie") est une valeur aléatoire, stockée dans un registre réservé. Il peut éventuellement être modifié dans certaines sous-fonctions. Son existence réduit le risque qu'un attaquant soit en mesure de modifier à la fois l'adresse de retour et la valeur du registre témoin.

Pour limiter l'impact sur les performances sans perdre de sécurité, la protection RAP peut être élidée des fonctions ne présentant aucun risque de corruption mémoire.

ICTP

Le second mécanisme de protection proposé est dénommé ICTP ("IndirectControl Transfer Protection") et se subdivise en deux parties : la validation des appels de fonctions indirects (incluant les appels de méthodes pour les langages à objets), et la validation des retours de fonctions.

La pierre angulaire du mécanisme ICTP est le marquage des arêtes du graphe de flot de contrôle par le type de la fonction appelée. Avant chaque transfert de contrôle indirect (appel ou retour), le marquage placé au site d'appel ou de retour est validé pour s'assurer que les types à la source et à la destination du transfert concordent.

Le marquage des sites d'appels et de retour repose sur un condensat de 64 bits, dérivé du type de la fonction appelée. Chaque fonction possède des éléments associés, tels que son nom, son arité, les types de ses paramètres et de son retour. Selon les cas, un sous-ensemble de ces éléments est pris en compte dans le calcul du condensat caractéristique du type de la fonction.

Ainsi, une fonction C dont le prototype est `int f(int, char*)` (fonction acceptant un entier et un pointeur vers un caractère comme paramètres, et retournant un entier) se verra attribuer un condensat portant sur les types de ses paramètres et son type de retour.

En effet, tout contexte d'appel où cette fonction pourrait être invoquée indirectement, à travers un pointeur, ne peut légitimement utiliser que des pointeurs dont le type est exactement le même.

En revanche, une méthode virtuelle C++ dont le prototype est `A* A::m(int)` (méthode d'un objet A ou dérivé, acceptant un entier, et retournant un pointeur vers un objet A ou dérivé) se verra attribuer un condensat ne portant que sur son paramètre d'entrée.

En effet, en supposant une classe B dérivée de A, il serait légitime de trouver un pointeur vers un objet B dans un contexte d'appel invoquant la méthode m. Le condensat ne doit donc pas porter sur le type du paramètre caché `this`, ni sur le type de retour de la méthode.

En pratique les condensats sont embarqués dans le code compilé. Dans le cas des fonctions, le condensat précède immédiatement le point d'entrée de la fonction. Il est donc facilement accessible par les appelants. Avant d'invoquer un pointeur de fonction, les appelants le valident en le comparant à celui du type associé au site d'appel.

En d'autres termes, le type dynamique constaté à l'exécution doit être compatible avec le type statique déterminé à la compilation.

En cas de différence entre les condensats, indiquant que le pointeur de fonction a été compromis, l'exécution du programme est avortée.

Dans le cas des retours de fonctions, un condensat apparaît 2 octets après l'adresse de retour (au site d'appel, l'instruction `call` est suivie d'un saut inconditionnel sur 2 octets, qui saute par dessus le condensat). La plage de valeur des condensats associés aux retours de fonctions est distincte de celle associée aux appels.

Avant chaque retour, l'épilogue de fonction lit la valeur du condensat au site d'appel, et le valide en le comparant à la valeur de référence dérivée du type de la fonction.

Vulnérabilités mitigées par ICTP

Ce double-mécanisme de validation des appels indirects et des retours limite l'impact de vulnérabilités et de techniques d'exploitation courantes, dont les suivantes :

Les vulnérabilités de type dépassement de tampon sur le tas ou de type utilisation-après-libération, permettent parfois à un attaquant de contrôler la valeur d'un pointeur de fonction. La validation des appels indirects restreint le choix des valeurs du pointeur usurpé aux fonctions du bon type.

De surcroît, les fonctions dont l'adresse n'est jamais prise dans le programme (celles qui sont toujours appelées directement, et pas par le biais de pointeurs) recoivent un condensat dans une plage de valeur encore différente de celles susceptibles d'être appelées indirectement. Il est donc impossible de diriger un pointeur usurpé vers une telle fonction, même si elle est du bon type.

L'enchaînement de retours caractéristique d'une chaîne ROP est également rendu plus difficile, pour deux raisons.

D'une part, si l'attaquant souhaite retourner à un point d'entrée de fonction, il doit fournir une adresse de retour pointant vers un site d'appel contenant le bon condensat. Or il est difficile de forger un tel site d'appel en l'absence de mémoire à la fois exécutable et inscriptible.

D'autre part, l'altération des épilogues de fonctions réduit le catalogue de gadgets ROP disponibles.

Conclusion

Les concepteurs de ces protections indiquent que leur impact sur les performances d'applications cibles, telles que le noyau de Linux ou le navigateur Chromium, semble relativement faible (de l'ordre de 5%). Il est donc envisageable de les voir se généraliser sur les plates-formes en source ouverte.

Cependant, notons que la vérification des sites d'appels lors des retours de fonctions exclut toute rétro-compatibilité binaire. Il est donc peu probable que des systèmes tels que Windows, où la compatibilité avec une énorme base d'applications existantes est essentielle, adoptent ces protections, du moins dans une forme pure et parfaite.

Documentation

https://pax.grsecurity.net/docs/PaXTeam-H2HC15-RAP-RIP-ROP.pdf

2 - Acquisition de mémoire vive

Le CERT-FR procède fréquemment à l'acquisition de la mémoire vive des ordinateurs devant faire l'objet d'investigation numérique. Il est donc important de posséder des outils maîtrisés ainsi qu'une méthodologie détaillée facilitant les opérations le jour de l'opération.

Acquisition sous Windows

Sous Windows, il existe plusieurs logiciels permettant de faire l'acquisition de la mémoire, parmi les outils gratuits il existe notamment :

- Winpmem ;
- FTK Imager ;
- Memoryze ;

Des outils commerciaux payants existent, proposant parfois une version gratuite avec des limitations lors de l'acquisition (certaines versions de Windows supportées, architecture 32bits uniquement, etc.), on peut citer notamment :

- Kntdd, de la suite Knttools ;
- FastDump, de la société HBGary ;
- DumpIt, de la société Moonsols ;
- Imager, de la suite F-Response.

L'acquisition nécessite les droits du groupe d'administration de la machine sur laquelle il est lancé, car le logiciel doit charger un pilote signé. Il est préférable de déposer le fichier directement sur un disque externe ou une clé USB de capacité suffisante (toujours d'une taille supérieure à la quantité de mémoire vive installée sur la machine). De cette manière, le risque d'écrasement des données résidentes sur le disque dur est limité. Par ailleurs, il convient d'utiliser sur le périphérique accueillant l'image de la mémoire, un système de fichiers supportant des fichiers d'une taille importante (par exemple, FAT32 ne supporte pas les fichiers de taille supérieure à 4Go). L'opération ne nécessite pas de redémarrage de l'ordinateur. Au contraire, un redémarrage éliminerait de la mémoire la majorité des éléments que l'on souhaite récupérer. La commande à lancer pour procéder à l'acquisition avec Winpmem est :

```
Winpmem.exe F:\image.raw
```

Acquisition sous Linux

Il existe de multiples sources de données permettant l'acquisition de la mémoire, mais celles-ci dépendent de la version du noyau et de la distribution utilisée : */dev/mem*, */dev/kmem*, */proc/kcore*, */dev/fmem*.

Chacune présente des avantages et des inconvénients en plus de n'être utilisable que sur certaines versions du noyau Linux. Le module noyau LiME permet de réaliser l'acquisition d'une image de la mémoire en faisant abstraction du périphérique utilisable sur la distribution employée. La compilation du module noyau nécessite cependant le téléchargement et l'installation des sources et les en-têtes du noyau. Une fois compilé, il faut insérer le module en mémoire (nécessitant des droits de super utilisateur) ; encore une fois il est conseillé d'écrire l'image sur un support externe afin de limiter l'impact sur le disque dur de la machine concernée. Comme sous windows, il est recommandé de ne pas redémarrer l'ordinateur lors de l'opération.

Exemple de ligne de commande permettant de récupérer l'image mémoire directement sur un support USB monté :

```
insmod ./lime.ko "path=/mnt/cle_usb/image.raw format=raw"
```

Enfin, pour faciliter l'analyse sous Linux, il est nécessaire de collecter le fichier *System.map* correspondant à la version du noyau utilisé, et de compiler les vtypes sur le système dont la mémoire est acquise. Cette étape est essentielle à la création d'un profil qui permettra d'identifier les structures des objets présents dans l'image mémoire nouvellement acquise. Le CERT-FR recommande de s'assurer au préalable que les équipes devant réaliser une opération d'acquisition de la mémoire disposent des droits d'administration. En outre, l'outil et la procédure d'acquisition doivent être régulièrement testé sur un environnement prévu à cet effet.

Liens

- Winpmem
  https://github.com/google/rekall/releases

- FTK Imager
  http://accessdata.com/support/adownloads#FTKImager
- Memoryze
  https://www.fireeye.com/services/freeware/memoryze.html
- LiME
  https://github.com/504ensicslabs/lime

3 - Rappel des avis émis

Dans la période du 16 au 22 novembre 2015, le CERT-FR a émis les publications suivantes :

- CERTFR-2015-AVI-488 : Multiples vulnérabilités dans libpng
- CERTFR-2015-AVI-489 : Multiples vulnérabilités dans les produits Cisco
- CERTFR-2015-AVI-490 : Vulnérabilité dans Drupal
- CERTFR-2015-AVI-491 : Vulnérabilité dans dovecot
- CERTFR-2015-AVI-492 : Multiples vulnérabilités dans les produits Cisco
- CERTFR-2015-AVI-493 : Multiples vulnérabilités dans Adobe ColdFusion
- CERTFR-2015-AVI-494 : Multiples vulnérabilités dans les produits Cisco
- CERTFR-2015-AVI-495 : Multiples vulnérabilités dans les pilotes Nvidia
- CERTFR-2015-AVI-496 : Vulnérabilité dans Vmware

Gestion détaillée du document

24 novembre 2015 version initiale.