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 09 mai 2014
N° CERTFR-2014-ACT-019

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2014-ACT-019

1 - Présentation du mécanisme de sécurité de type bac à sable

Avec l'exploitation de plus en plus fréquente de vulnérabilités de type *0day* visant des logiciels tiers au système d'exploitation (par exemple les navigateurs Internet, les lecteurs de documents, les logiciels de traitement de texte, etc.), certains éditeurs ont pris le parti d'inclure un nouveau mécanisme de sécurité au sein de leurs produits : la protection dite « bac à sable » ou plus communément appelée *sandbox*.

Le principe de ce mécanisme est d'architecturer un logiciel autour de deux processus (ou plus). Un premier processus, dit *broker*, conserve les privilèges de l'utilisateur sur le système et est utilisé pour lancer de nouveaux processus fils tout en leur affectant des privilèges moins élevés (les processus protégés dits *sandboxés*).

Le processus « sandboxé » est alors utilisé pour l'interprétation de code provenant de sources inconnues (pages HTML, documents PDF, etc). Ses droits étant contrôlés, ce dernier est limité dans les actions dangereuses qu'il pourrait réaliser sur le système d'exploitation (par exemple il ne lui est pas possible de créer un fichier dans le dossier système ou d'écrire une clef registre de type *run*).

Cependant, ces processus peuvent avoir besoin, de manière occasionnelle et légitime, de plus de privilèges que ceux qui leur ont été accordés. Pour gérer ce genre de cas, un canal de communication existe entre le *broker* et la *sandbox* et permet au processus « sandboxé » de demander au *broker* de réaliser l'action à sa place.

Par exemple, lorsqu'un utilisateur souhaite enregistrer un document sur son disque, un message est envoyé au *broker* demandant la création du fichier. Ce dernier, disposant des droits nécessaires à la réalisation de cette opération, retourne un *handle* sur le fichier ouvert au processus « sandboxé ». Ce dernier peut ensuite compléter l'opération d'écriture du fichier sur le disque.

L'intérêt de ce mode de fonctionnement est de permettre à l'éditeur d'implémenter un jeu de règles acceptant (ou refusant) par défaut certaines opérations en se basant sur le type de demande ou les paramètres passés en argument (nom de fichier, clef registre, etc.). Une autre possibilité de ce jeu de règles est de demander confirmation à l'utilisateur s'il souhaite poursuivre ou non l'opération (par exemple après avoir cliqué sur un lien hypertexte dans un document et avant de lancer l'ouverture de ce lien dans un navigateur Internet).

Ce mécanisme permet donc de renforcer la sécurité du produit en augmentant la difficulté, pour l'attaquant, d'écrire un code d'exploitation lui permettant de s'installer durablement sur la machine. En effet, afin de voir son code s'exécuter avec les mêmes droits que l'utilisateur courant, l'attaquant doit exploiter deux vulnérabilités :

- une première visant la *sandbox*, afin d'exécuter du code dans un contexte restreint ;
- une deuxième visant le *broker*, afin d'exécuter du code avec les mêmes droits que l'utilisateur.

Il est à noter qu'en lieu et place d'une vulnérabilité visant le *broker*, l'attaquant peut aussi utiliser une vulnérabilité ayant pour cible le noyau du système d'exploitation afin d'exécuter du code avec les privilèges les plus élevés.

De plus en plus de produits « courants » intègrent dans leurs dernières versions ce type de protection. Il est possible de citer :

- Google Chrome (depuis les premières versions).
- Microsoft Internet Explorer (à partir de la version 7) ;
- Microsoft Office (à partir de la version 2003) ;
- Adobe Acrobat Reader (à partir de la version X) ;

Le CERT-FR recommande donc de privilégier l'utilisation des applications qui bénéficient de ce type de protection et met en garde ses utilisateurs contre de nombreux greffons (tels que Oracle Java) qui, même s'ils sont lancés depuis un produit intégrant le mécanisme de *sandbox*, contournent cette protection et s'exécutent avec les droits de l'utilisateur.

2 - Déploiement d'un mécanisme de contrôle d'accès obligatoire basé sur Tomoyo

Comme présenté dans le bulletin d'actualité du 25 avril 2014, plusieurs solutions de contrôle d'accès obligatoire sont intégrées dans le noyau Linux :

- SELinux
- Smack
- AppArmor
- Tomoyo

Les mécanismes de contrôle d'accès obligatoire basés sur les labels peuvent présenter des difficultés :

- nécessité de « marquer » toutes les arborescences de fichiers ;
- problèmes avec certains éditeurs de fichiers qui provoquent des pertes de labels ;
- etc.

Dans le cas où des fonctionnalités avancées, telles que celles proposées par SELinux, ne sont pas nécessaires (ex : support du multi-niveaux), une solution alternative comme Tomoyo, plus simple à mettre en œuvre, peut être envisagée.

Présentation de Tomoyo

Tomoyo est un mécanisme de contrôle d'accès obligatoire, intégré au noyau Linux. Il se base sur les chemins des fichiers exécutables ainsi que sur leur contexte d'exécution pour l'application de règles de contrôle d'accès aux ressources du système (fichiers, réseau, variables d'environnement, etc.). Tomoyo fonctionne par liste blanche, c'est-à-dire que tout ce qui n'est pas explicitement autorisé est interdit.

Un des objectifs des développeurs de Tomoyo est de proposer des outils et une syntaxe de configuration simples.

Tomoyo a été développé dans un premier temps comme un patch pour le noyau Linux, avant d'y être intégré à partir de la version 2.6.30 du noyau. À cette fin, le code de Tomoyo a été modifié afin d'utiliser l'interface Linux Security Modules (LSM).

En conséquence, il existe à ce jour deux grandes branches de Tomoyo :

- la version 1.x, sous forme de patch du noyau Linux ;
- la version 2.x, intégrée au noyau sous forme de LSM, activable en positionnant la directive de compilation noyau `CONFIG_SECURITY_TOMOYO=y`.

A noter qu'un certain nombre de fonctionnalités ne sont pas disponibles dans la version LSM (2.x) de Tomoyo, du fait des limitations de cette interface (restriction des capacités, restriction du *loader*, restriction des envois de signaux, etc.).

De nombreuses distributions sont livrées avec un noyau précompilé avec le support de Tomoyo. Afin de vérifier le support de ce dernier, il suffit de chercher la chaîne de caractères « tomoyo » dans les symboles exportés par le noyau :

```
# grep tomoyo /proc/kallsyms | head -1
XXXXXXXXXXXXXXXX T tomoyo_init_log
```

Les noyaux fournis avec la distribution Debian sont compilés avec le support de Tomoyo. Cete distribution sera prise à titre d'exemple dans le reste de l'article.

Fonctionnement de Tomoyo

Tomoyo introduit la notion de « domaine », qui permet de considérer nativement les restrictions d'utilisation d'un binaire selon son contexte d'exécution, c'est-à-dire selon l'arborescence des processus parents. Pour un domaine, la configuration de Tomoyo définit les ressources (fichiers, *sockets*, *fifo*, etc.) et les opérations (lecture, écriture, création, envoi de trames, etc.) autorisées lors de l'exécution du binaire.

Tomoyo offre une grande souplesse concernant les domaines : il est possible de définir un domaine « générique » applicable quelque soit le contexte d'exécution du binaire, comme d'appliquer des profils particuliers dépendant du contexte d'exécution.

Afin de faciliter la création de domaines, Tomoyo propose, entre autres, un mécanisme appelé profil d'apprentissage. Le domaine en apprentissage n'appliquera aucune restriction, mais journalisera les différents accès aux ressources du système dans un fichier. Ce fichier pourra être ensuite directement utilisé comme fichier de configuration.

Une fois un domaine renseigné (soit manuellement par l'administrateur du système, soit automatiquement par le profil d'apprentissage), il est possible d'appliquer la politique du domaine selon plusieurs modes :

- journalisation des accès ne respectant pas les restrictions du domaine ;
- application des restrictions du domaine ;
- application sélective des restrictions (par exemple n'appliquer que les restrictions correspondant à l'accès en lecture aux fichiers).

Mise en place de Tomoyo

De par son inclusion dans le code du noyau, le déploiement de Tomoyo sur un serveur ne nécessite que l'installation des outils « user-land » (package Debian tomoyo-tools) ainsi que son activation lors du lancement du noyau, au travers des actions suivantes :

- modification du fichier /etc/default/grub :
  `GRUB_CMDLINE_LINUX_DEFAULT="quiet security=tomoyo"`
- application de la modification de la ligne de démarrage Linux dans grub :
  `# update-grub`
- création d'une politique par défaut :
  `# tomoyo-editpolicy /etc/tomoyo`
- redémarrage du serveur.

Tomoyo fournit une suite d'outils permettant de gérer les différents éléments constituant son architecture. En particulier, l'outil `tomoyo-editpolicy` permet d'administrer les politiques de domaines, les profils appliqués (désactivé, apprentissage, journalisation, restreint) et de tracer les ressources autorisées lors de la phase d'apprentissage.

Les fichiers de configuration de Tomoyo sont des fichiers texte facilement compréhensibles et éditables manuellement. La syntaxe supporte un format d'expressions rationnelles permettant de limiter les conséquences d'une erreur de configuration. Par exemple, les expressions rationnelles *greedy* ne prennent jamais en considération le caractère barre oblique (indiquant une séparation de chemin sous Unix), ce qui permet d'éviter l'écriture non sollicitée dans un répertoire de niveau supérieur (remontée d'arborescence du type « ../../../../ »).

Les outils Tomoyo permettent la modification dynamique des restrictions et du profil de contrôle d'accès appliqués à un domaine, par simple modification de la politique de ce dernier (pas de redémarrage du serveur nécessaire).

Exemples d'application

La documentation officielle [1] du projet propose comme exemple le déploiement de Tomoyo pour la sécurisation d'un service Web porté par Apache. De même, une solution permettant de restreindre en lecture seule ou en lecture / écriture l'accès à un serveur SFTP est présenté en [2].

Un autre exemple d'utilisation peut être la restriction des commandes que peut exécuter un utilisateur connecté en SSH. Cette configuration permet d'illustrer la notion de domaine : l'utilisateur connecté en SSH ne pourra exécuter qu'un sous-ensemble de commandes, mais ne sera pas soumis à de telles restrictions s'il se connecte localement sur le système.

Dans un premier temps, il suffit de créer un profil limitant l'exécution de commandes aux seules déclarées dans le domaine et limitant les connections réseau :

```
cat >> /etc/tomoyo/profile.conf <<EOF
5-COMMENT=-----NoSocketAndRestrictExecute Mode-----
5-PREFERENCE={ max_audit_log=1024 max_learning_entry=2048 }
5-CONFIG::file::execute={ mode=enforcing grant_log=yes reject_log=yes }
5-CONFIG::network={ mode=enforcing grant_log=yes reject_log=yes }
5-CONFIG={ mode=disabled grant_log=no reject_log=yes }
EOF

# tomoyo-loadpolicy -p < /etc/tomoyo/profile.conf
```

Il faut ensuite définir un domaine à appliquer au *shell* bash lorsqu'il est lancé par le service SSHD :

```
# cat >> /etc/tomoyo/exception_policy.conf <<EOF
initialize_domain /usr/sbin/sshd from any
keep_domain <kernel> /usr/sbin/sshd /bin/bash
EOF
# tomoyo-loadpolicy -ef < /etc/tomoyo/exception_policy.conf
```

Enfin, il reste à identifier les binaires que l'on souhaite autoriser dans le contexte d'une connexion SSH :

```
# cat >> /etc/tomoyo/domain_policy.conf<<EOF
<kernel> /usr/sbin/sshd /bin/bash
use_profile 5
use_group 0

file execute /bin/basename
file execute /bin/cat
file execute /usr/bin/clear
file execute /usr/bin/clear_console
file execute /usr/bin/cut
file execute /usr/bin/dircolors
file execute /bin/egrep
file execute /bin/grep
file execute /bin/hostname
file execute /usr/bin/id
file execute /bin/ls
[...]
EOF
# tomoyo-loadpolicy -df < /etc/tomoyo/domain_policy.conf
```

La journalisation des accès refusés, effectuée par le service `tomoyo-auditd`, est consultable dans le fichier /var/log/tomoyo/reject_003.log.

Conclusion

Tomoyo est une solution pouvant être déployée à peu de frais, car intégrée dans la majorité des noyaux fournis avec les distributions GNU/Linux. Les outils en user-land permettent une souplesse d'administration, en proposant notamment la création simple et pertinente de domaines avec le mode d'apprentissage, et la mise en place de contrôle d'accès sans à avoir à redémarrer le serveur.

Une des critiques majeures portée à l'encontre de Tomoyo est l'utilisation du nom des binaires et non de leur label. Il est en effet possible de contourner une restriction en créant un lien dur vers le binaire visé, ou en faisant usage de points de montage arbitraires. Cependant, l'approche liste blanche permettrait, avec une configuration adaptée, de restreindre la création de tels liens et d'encadrer l'utilisation de la commande mount à l'échelle du système.

Documentation

1 http://tomoyo.sourceforge.jp/2.5/index.html.en
2 http://tomoyo.sourceforge.jp/2.5/sftp-protection-using-environment-variable.html.en

3 - Rappel des avis émis

Dans la période du 02 au 08 mai 2014, le CERT-FR a émis les publications suivantes :

- CERTFR-2014-AVI-210 : Vulnérabilité dans Microsoft Internet Explorer
- CERTFR-2014-AVI-211 : Multiples vulnérabilités dans plusieurs produits Cisco
- CERTFR-2014-AVI-212 : Multiples vulnérabilités dans les produits Huawei
- CERTFR-2014-AVI-213 : Vulnérabilité dans Microsoft Windows
- CERTFR-2014-AVI-214 : Vulnérabilité dans Apache Struts
- CERTFR-2014-AVI-215 : Vulnérabilité dans les produits F5

Durant la même période, les publications suivantes ont été mises à jour :

- CERTFR-2014-ALE-005-001 : Vulnérabilité dans Microsoft Internet Explorer (fermeture de l'alerte, suite à la diffusion du correctif par l'éditeur.)

Gestion détaillée du document

09 mai 2014 version initiale.