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 21 décembre 2015
N° CERTFR-2015-ACT-051

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2015-ACT-051

1 - Intérêt des fichiers SuperFetch pour l'investigation numérique

Cet article s'inscrit dans la lignée des bulletins d'actualité CERTFR-2014-ACT-037 et CERTFR-2015-ACT-044, concernant l'identification des programmes exécutés dans le cadre de l'investigation numérique.

Un mot sur les fichiers Prefetch

Dans le cadre de ses analyses, le CERT-FR est régulièrement amené à analyser les fichiers Prefetch, si ceux-ci sont disponibles, afin de déterminer la date d'exécution d'un programme sur le système et éventuellement l'emplacement depuis lequel il a été exécuté.

Pour rappel, les fichiers Prefetch **.pf*, introduits sous Windows XP et localisés dans *%SystemRoot%\Prefetch*, sont utilisés par le système d'exploitation pour caractériser les applications exécutées par le système et l'utilisateur.

Cette fonctionnalité permet de déterminer les pages mémoires de code utilisées par un programme afin de les charger préalablement lors de l'exécution de ce dernier. L'objectif ainsi visé est d'éviter un maximum d'accès disque. Par défaut, le Prefetch est désactivé pour tous les programmes sur les environnements Windows Server.

Ce paramètre est stocké dans la valeur suivante :

```
HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\
PrefetchParameters\EnablePrefetcher
```

Windows Vista et SuperFetch

Introduit avec le noyau de Vista, SuperFetch est un procédé de gestion de la mémoire, à l'image du Prefetcher introduit sous XP. SuperFetch vise à améliorer les performances générales du système via un mécanisme de prédiction d'utilisation des pages mémoire de code en fonction de scénarios temporels (exécution en semaine ou week-end, utilisation entre 6 heures et midi, midi et 18h, 18h et minuit).

Prefetch ne se base que sur l'activité récente du système pour charger préalablement des données. Si une application utilise intensivement la mémoire, l'historique d'utilisation des pages sera faussé. SuperFetch tente d'optimiser ce modèle de gestion mémoire par le composant de rééquilibrage (rebalancer), qui permet de prioriser à nouveau la liste des pages mémoire en fonction de leur historique d'utilisation et des scénarios temporels établis.

Les fichiers *Ag*.db* constituent une base d'informations sur l'historique d'utilisation des programmes et de leurs pages mémoire de code. Par abus de langage, ils sont nommés fichiers SuperFetch, bien que ce terme englobe le procédé de gestion mémoire optimisé dans son ensemble. A l'instar des fichiers Prefetch, ils se trouvent dans le dossier *%SystemRoot%\Prefetch*.

Comme pour Prefetcher, le SuperFetch est désactivé par défaut pour tous les programmes sur les environnements Windows Server.

Le paramètre est contenu dans la valeur suivante :

`HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters\EnableSuperfetch`

Initialement, *enablePrefetcher* et *enableSuperfetch* étaient désactivée par défaut sur Windows 7 sur les systèmes équipés de disques SSD, afin d'améliorer la durée de vie des premiers modèles de disque. L'option *enablePrefetcher* a été réactivée par défaut à partir de Windows 8, mais ce n'est pas le cas pour SuperFetch.

Utilité des artefacts d'exécution de programme

Dans le cadre d'une investigation lors d'un incident de sécurité informatique, l'analyste sera intéressé de savoir si un programme a été exécuté (le nombre de fois, la date, la fréquence et l'emplacement). Cela peut mettre en avant un comportement malveillant si une application suspecte est utilisée, ou déterminer la légitimité d'une autre par une étude statistique.

Les fichiers Prefetch peuvent être décodés avec les outils suivants :

- Pf de TzWorks (payant) ;
- CrowdResponse de CrowdStrike (gratuit) ;
- Windows file analyzer de Mitec (gratuit) ;
- le greffon prefetch de RegRipper (libre et gratuit) ;
- Prefetch-parser de Airbus DS (libre et gratuit).

Les fichiers SuperFetch peuvent être partiellement décodés avec les outils suivants :

- CrowdResponse de CrowdStrike (gratuit) ;
- Superfetch-dumper de Rewolf (libre et gratuit).

Documentation

- http://cert.ssi.gouv.fr/site/CERTFR-2014-ACT-037
- http://cert.ssi.gouv.fr/site/CERTFR-2015-ACT-044
- https://technet.microsoft.com/en-us/magazine/2007.03.vistakernel.aspx
- https://www.crowdstrike.com/blog/crowdresponse-application-execution-modules-released
- M. Russinovitch, D. Solomon, A. Ionescu. *Windows Internals vol.2*. Microsoft Press. p.338
- https://github.com/libyal/libagdb/blob/master/documentation/Windows SuperFetch (DB) format.asciidoc

2 - Analyse de la vulnérabilité Joomla!

Contexte

Le 14/12/2015, le CERT-FR a émis une alerte sur une vulnérabilité critique Joomla! (corrigée à cette même date par l'éditeur). Celle-ci donne à un attaquant la possibilité d'exécuter des commandes sur le serveur vulnérable (avec les droits du processus PHP), et ce sans nécessiter une authentification préalable.

Cette vulnérabilité est d'autant plus critique que l'identification de serveurs vulnérables peut s'effectuer via des moteurs de recherche, permettant ainsi une automatisation de l'exploitation.

Marqueurs de détection

Une publication, provenant de la société de sécurité SUCURI, propose des éléments de détection de cette vulnérabilité. Les marqueurs publiés peuvent être recherchés sur les journaux des serveurs. Il s'agit des chaînes "JDatabaseDriverMysqli" ou "O:", au niveau de l'entête HTTP User-Agent.

Des tentatives d'exploitation ont été observées par des partenaires du CERT-FR, avant la publication de ce correctif. Les adresses IP suivantes auraient laissé des traces d'exploitation dans certains journaux :

- 37.61.232.173 ;
- 74.3.170.33 ;
- 93.179.68.167 ;
- 146.0.72.83 ;
- 185.15.185.17 ;
- 194.28.174.106 ;

– 199.182.234.132.

Enfin, depuis la publication du correctif, plusieurs codes d'exploitation ont été publiés sur Internet (notamment dans le cadriciel Metasploit).

Il est à noter que certains de ces codes utilisent d'autres entêtes HTTP que le user-agent, évitant ainsi d'être inscrits dans les journaux. Une détection par la méthode évoquée précédemment n'est ainsi pas possible.

Versions affectées

Afin d'être exploitée, cette vulnérabilité nécessite que la version de Joomla! soit inférieure ou égale à 3.4.5. De plus, elle ne peut être exploitée que si la version de PHP utilisée est inférieure ou égale à 5.6.12, elle-même exposée à la vulnérabilité CVE-2015-6835.

Cette vulnérabilité PHP permet de manipuler la désérialisation des éléments de session afin de charger des objets arbitraires. Le correctif associé concerne une vulnérabilité de type "utilisation après libération". Le durcissement du chargement des objets sérialisés n'est qu'un "effet de bord".

Description de la vulnérabilité Joomla!

La vulnérabilité Joomla! repose sur un défaut de validation des données provenant de l'utilisateur, dans le fichier "libraries/joomla/session/session.php". Cela permet à un attaquant de contrôler la valeur de l'entête "User-Agent" (1), par la suite enregistrée dans les variables de session PHP (2).

Cet enregistrement s'effectue lors de la création ou lors du redémarrage d'une session Joomla (avec appel de la fonction "_validate").

```
    protected function _validate($restart = false)
    {
        [...]

        // Check for clients browser
        if (in_array('fix_browser', $this->_security) &&
isset($_SERVER['HTTP_USER_AGENT']))
        {
            $browser = $this->get('session.client.browser');

            if ($browser === null)
            {
(1)                 $this->set('session.client.browser',
$_SERVER['HTTP_USER_AGENT']);
            }
            [...]
        }
    }

    public function set($name, $value = null, $namespace = 'default')
    {
        $namespace = '__' . $namespace;
        [...]
(2)     $_SESSION[$namespace][$name] = $value;
        [...]
    }
```

Lors de la création ou de la fermeture d'une session, le contenu de la variable globale $_SESSION est décodé ou encodé grâce à un appel aux fonctions natives "session_encode" ou "session_decode". Ces fonctionnalités d'encodage et de décodage sont similaires (mais pas identique) au principe de sérialisation de PHP.

De plus, Joomla! surcharge le mécanisme de stockage de session de PHP afin de définir différentes méthodes de stockage des données. L'une d'entre elles permet la lecture et l'insertion des données dans une base de données :

```
    public function write($id, $data)
    {
        $db = JFactory::getDbo();
```

```
        $data = str_replace(chr(0) . '*' . chr(0), '\0\0\0', $data);
        try
        {
            $query = $db->getQuery(true)
                ->update($db->quoteName('#__session'))
                ->set($db->quoteName('data') . ' = ' . $db->quote($data))
                ->set($db->quoteName('time') . ' = ' . $db->quote((int) time()))
                ->where($db->quoteName('session_id') . ' = ' .$db->quote($id));

            $db->setQuery($query);

            if (!$db->execute())
            [...]
            return true;
        }
    }

    public function read($id)
    {
        $db = JFactory::getDbo();
        try
        {
            // Get the session data from the database table.
            $query = $db->getQuery(true)
                ->select($db->quoteName('data'))
                ->from($db->quoteName('#__session'))
                ->where($db->quoteName('session_id') . ' = ' . $db->quote($id));

            $db->setQuery($query);

            $result = (string) $db->loadResult();

            $result = str_replace('\0\0\0', chr(0) . '*' . chr(0), $result);

            return $result;
        }
        [...]
    }
```

Le fait que les données provenant de l'entête "User-Agent" soient directement enregistrées en base présente deux avantages, du point de vue de l'attaquant :

- la table stockant les sessions dans la base de données MySQL utilise le jeu de caractères "utf8_general_ci". Ce jeu de caractères a pour effet de tronquer les données insérées en base dès qu'un caractère unicode invalide est détecté (utilisant plus de trois octets pour son stockage). Cela permet à l'attaquant de "couper" la chaîne de caractères représentant les données sérialisées afin que, lors de la désérialisation, l'objet forgé soit correctement chargé en mémoire ;
- lors de la récupération de données, la chaîne de caractères "\0\0\0" est convertie en "<NULL>*<NULL>". Cette suite de caractères sert à représenter les propriétés de type "protected" dans la chaîne de caractères sérialisée. En insérant "\0\\0", l'attaquant s'assure de pouvoir définir des paramètres "protected", ce qui lui aurait été impossible sans cette conversion car cela aurait terminé de façon prématurée sa charge utile.

Lors d'une deuxième connexion de l'attaquant avec la même session, les données sont chargées depuis la base de données et passées à la fonction "session_decode". Comme cette dernière interprète incorrectement les données sérialisées, l'attaquant est en mesure de charger en mémoire un objet dont il contrôle les propriétés. Il y a cependant quelques restrictions lors de l'instanciation des objets :

- l'attaquant ne peut contrôler que les propriétés, pas le code de l'objet ;
- l'objet doit avoir été préalablement défini afin que le moteur PHP puisse le sérialiser.

L'analyse des données stockées en base, après exploitation de la vulnérabilité, permet de récupérer la chaîne de caractères forgée et ainsi mieux comprendre le cheminement aboutissant à l'exécution de code arbitraire.

Dans un premier temps, viennent les valeurs "légitimes" correspondant à un tableau de 8 éléments stocké par Joomla!.

```
__default|a:8:
    {
        s:15:"session.counter";i:1;
        s:19:"session.timer.start";i:1446585738;
        s:18:"session.timer.last";i:1446585738;
        s:17:"session.timer.now";i:1446585738;
        s:22:"session.client.browser";s:615:
```

Puis, à partir de l'élément "session.client.browser", les données enregistrées ont été manipulées par l'attaquant :

```
"}__test|O:21:"JDatabaseDriverMysqli":3:{s:2:"fc";O:17:"JSimplepieFactory":0:{}
s:21:"\0\0\0disconnectHandlers";a:1:{i:0;a:2:{i:0;O:9:"SimplePie":5:{s:8:"sanitize"
;O:20:"JDatabaseDriverMysql":0:{}s:8:"feed_url";s:239:"eval(chr(115).chr(121).chr(115)
[..];exit";s:19:"cache_name_function";s:6:"assert";s:5:"cache";b:1;s:11:"cache_class";
O:20:"JDatabaseDriverMysql":0:{}}i:1;s:4:"init";}}s:13:"\0\0\0connection";b:1;}
```

Dans sa chaîne spécialement conçue, l'attaquant s'attache tout d'abord à "fermer" le tableau nommé "__default" puis a définir son propre objet appelé "__test".

La structure de l'objet "__test" peut se représenter schématiquement sous la forme suivante :

```
JDatabaseDriverMysqli
    - fc = JSimplepieFactory
    - disconnectHandlers = (object, function_name) (b)
    - connection = True (a)
```

La variable "disconnectHandlers" représentant un tableau dont le premier élément est lui même un tableau contenant :

```
    - object = SimplePie (c)
        - sanitize = JDatabaseDriverMysql (e)
        - feed_url = "eval(chr(115).chr(121).chr(115)[..];exit"
        - cache_name_function = "assert"
        - cache = True
        - cache_class = JDatabaseDriverMysql
    - function_name = "init" (d)
```

L'attaquant choisit d'instancier l'objet "JDatabaseDriverMysqli" car dans sa classe est définie une fonction spécifique : "__destruct". Cette dernière est appelée lors de la destruction de l'objet (quand plus aucune référence n'est faite sur ce dernier).

Lors de la destruction de l'objet, un appel est fait à la fonction "disconnect". Cette fonction vérifie l'état de la propriété "connection" (3) et parcourt le tableau de fonctions "disconnectHandlers" en appelant chaque fonction stockée avec l'appel natif "call_user_func_array" (4).

```
    public function __destruct()
    {
        $this->disconnect();
    }

    public function disconnect()
    {
        // Close the connection.
(3)        if ($this->connection)
        {
            foreach ($this->disconnectHandlers as $h)
            {
(4)                call_user_func_array($h, array( &$this));
            }

            mysqli_close($this->connection);
        }
```

```
        $this->connection = null;
    }
```

Comme l'attaquant contrôle les variables, il aura pris soin de définir "connection" à vrai (a) et d'enregistrer dans le tableau "disconnectHandlers" un tableau contenant un couple objet (c), fonction (d) permettant à "call_user_func_array" d'appeler la fonction "init" propre à l'objet "SimplePie".

Cette fonction "init" se charge tout d'abord d'effectuer des vérifications sans importance sur l'environnement PHP. Puis des appels sont effectués sur des fonctions de l'objet "sanitize" (précédemment défini comme instance de "JDatabaseDriverMysql"). Le flot d'instructions est ainsi guidé pour aboutir à un appel à "call_user_func" avec des paramètres contrôlés par l'attaquant (5). La variable "cache_name_function" a alors pour valeur "assert" et "feed_url" représente la charge utile. Cet appel provoque donc l'exécution de code arbitraire recherchée.

```
    function init()
    {
        // Check absolute bare minimum requirements.
        [...]
        // Pass whatever was set with config options over to the sanitizer.
        $this->sanitize->pass_cache_data($this->cache, $this->cache_location,
            $this->cache_name_function, $this->cache_class);
        $this->sanitize->pass_file_data($this->file_class, $this->timeout,
            $this->useragent, $this->force_fsockopen);

        if ($this->feed_url !== null || $this->raw_data !== null)
        {
            $this->data = array();
            $this->multifeed_objects = array();
            $cache = false;

            if ($this->feed_url !== null)
            {
                $parsed_feed_url = SimplePie_Misc::parse_url($this->feed_url);
                // Decide whether to enable caching
                if ($this->cache && $parsed_feed_url['scheme'] !== '')
                {
(5)                 $cache = call_user_func(array($this->cache_class,'create'),
                     $this->cache_location, call_user_func($this->cache_name_function,
                     $this->feed_url), 'spc');
                }
            [...]
```

Recommandations du CERT-FR

Devant la criticité de cette vulnérabilité, le CERT-FR recommande de procéder le plus vite possible à la mise à jour de PHP et de Joomla!. De plus, une grande vigilance doit être apportée aux comportements suspects sur des serveurs ayant hébergé des versions vulnérables de ces outils.

En effet, l'exploitation de la vulnérabilité avant la sortie d'un correctif, la disponibilité de scripts d'exploitation sur Internet (dont certains ne laissent pas de traces dans les journaux), l'existence d'autres objets pouvant être utilisés pour le détournement du flot d'exécution (par exemple, sur un serveur PostgreSQL, la classe "JDatabaseDriverPostgresql" est susceptible d'être utilisée) sont autant d'éléments qui justifient cette vigilance accrue.

Documentation

- Bulletin d'alerte du CERT-FR :
  http://cert.ssi.gouv.fr/site/CERTFR-2015-ALE-013/index.html
- Publication de la société Sucuri présentant les techniques de détection :
  https://blog.sucuri.net/2015/12/remote-command-execution-vulnerability-in-joomla.html
- Publication de la société Sucuri d'une analyse de la vulnérabilité :
  https://blog.sucuri.net/2015/12/joomla-remote-code-execution-the-details.html

3 - Rappel des avis émis

Dans la période du 14 au 20 décembre 2015, le CERT-FR a émis les publications suivantes :

- CERTFR-2015-ALE-013-001 : Vulnérabilité dans Joomla!
- CERTFR-2015-ALE-014 : Vulnérabilité dans Juniper ScreenOS
- CERTFR-2015-AVI-540 : Vulnérabilité dans PuTTY
- CERTFR-2015-AVI-541 : Multiples vulnérabilités dans Joomla!
- CERTFR-2015-AVI-542 : Multiples vulnérabilités dans Cisco
- CERTFR-2015-AVI-543 : Multiples vulnérabilités dans Mozilla
- CERTFR-2015-AVI-544 : Multiples vulnérabilités dans Google Chrome
- CERTFR-2015-AVI-545 : Multiples vulnérabilités dans Samba
- CERTFR-2015-AVI-546 : Vulnérabilité dans Cisco Unified Communications Manager
- CERTFR-2015-AVI-547 : Multiples vulnérabilités dans Citrix
- CERTFR-2015-AVI-548 : Vulnérabilité dans Cisco Application Policy Infrastructure Controller
- CERTFR-2015-AVI-549 : Multiples vulnérabilités dans le noyau Linux d'Ubuntu
- CERTFR-2015-AVI-550 : Multiples vulnérabilités dans Juniper ScreenOS
- CERTFR-2015-AVI-551 : Multiples vulnérabilités dans Xen
- CERTFR-2015-AVI-552 : Multiples vulnérabilités dans les produits Cisco
- CERTFR-2015-AVI-553 : Multiples vulnérabilités dans les produits Bluecoat

Gestion détaillée du document

21 décembre 2015 version initiale.