Analyse d’une vulnérabilité de type Cross Site Scripting (CWE-79) CVE 2017-10991

Robin Stephane

PRESENTATION

La CVE 2017-10991 concerne l’extension de WordPress WP-Statistics antérieure à la version 12.0.10, et s’exécute uniquement avec le navigateur Mozilla Firefox.

WordPress est un système de gestion de contenu libre, gratuit et open-source utilisé par plus de 30 % des sites Web dans le monde. Ses fonctionnalités lui permettent de créer et gérer différents types de sites : blogs, vente en ligne, portfolio. Ecrit en PHP, il repose sur une base de données MySQL et est distribué selon les termes de la licence GNU GPL version 2. *(source Wikipedia)*

WP-Statistics est une extension de Wordpress permettant de répertorier et de représenter les données statistiques d’un site Internet. L’intérêt de cette application réside essentiellement dans le fait qu’il n’est pas nécessaire d’envoyer les données des utilisateurs à un serveur tiers pour obtenir ces informations. WP-Statistics est conforme au Règlement Général sur la Protection des Données. *(source WordPress.org)*

Mozilla Firefox est un navigateur Web libre, gratuit et open-source, distribué sous licence MPL. Ce logiciel multi-plateformes est compatible avec les systèmes d’exploitation Windows, MacOS, Linux, Android et iOS. Il est parmi les navigateurs les plus utilisés au monde en 2019. *(source Wikipedia)*

TYPE DE CVE

La CVE 2017-10991 est de type Cross-Site Scripting (XSS) non-persistant. Le Cross-Site Scripting est une attaque par injection qui exploite les interactions entre une application Web et ses utilisateurs. Cette faille de sécurité permet d’injecter du contenu malveillant dans une page, à travers un formulaire ou encore une URL. Le langage utilisé doit être pris en charge par le navigateur Web afin de provoquer des actions de celui-ci lorsqu’il visite la page concernée

par l'attaque. Ces actions peuvent être immédiates ou différées, et dans ce cas la donnée malicieuse est stockée dans le serveur. La CVE 2017-10991 est non-persistante car elle s'exécute immédiatement dans la page suite à la soumission des données malicieuses.

HISTORIQUE DE LA CVE

07 juillet 2017 : 1ere publication de la vulnérabilité
19 juillet 2017 : enregistrement et publication de la CVE
31 août 2018 : dernière modification par le vendeur

DESCRIPTION DE LA CVE

Jusqu'à la version 12.0.9 incluse, l'extension WP-Statistics était vulnérable au Cross-Site Scripting non-persistant. Dans la page `wps_referrers_page` les paramètres `rangestart` et `rangeend` de la méthode `GET` ne sont ni cryptés ni sanitisés, ce qui est favorable à une attaque XSS. Ainsi, l'administrateur du site qui serait resté connecté après s'être authentifié, pourrait être incité par un hacker à cliquer sur un lien malicieux, et voir ainsi une page de son site modifiée à l'aide d'un script écrit en JavaScript.

La CVE 2017-10991 permet donc d'exécuter du code JavaScript dans le contexte d'un site Web.

CHOIX DE LA CVE

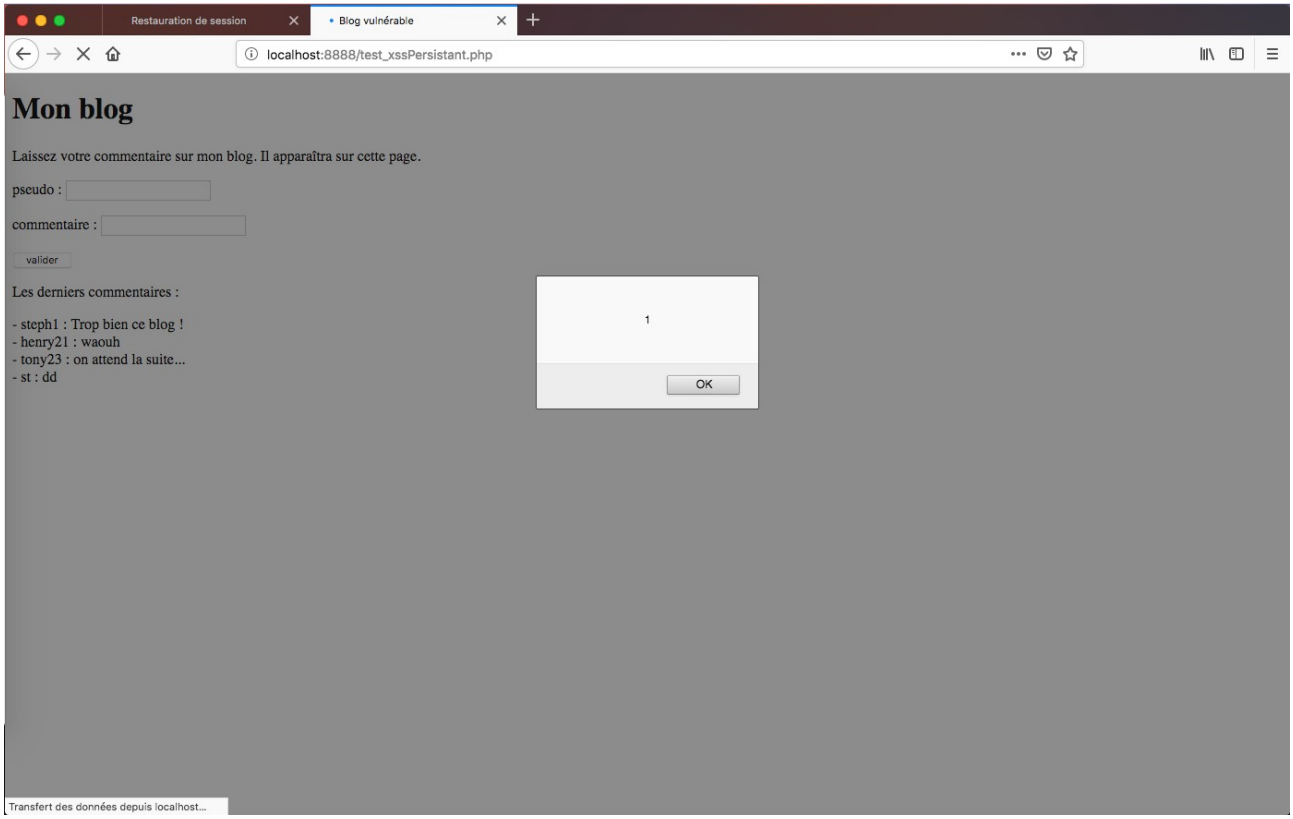
Nous voulions travailler sur une vulnérabilité de type XSS non-persistant. Pour mieux en comprendre le fonctionnement, nous avons créé une mini-page de blog en PHP contenant deux formulaires pseudo et commentaire. Le code est présenté ci-dessous :

```php
<!-- traitement de la saisie utilisateur avant la creation de page -->
<?php
$msgError = "";
if(isset($_POST['submit'])){
  if($_POST['pseudo'] !="" and $_POST['commentaire'] != ""){
    //$_POST['name'] = filter_var($_POST['name'], FILTER_SANITIZE_STRING);
    //$_POST['password'] = filter_var($_POST['password'], FILTER_SANITIZE_STRING);
    insererDonnees($_POST['pseudo'], $_POST['commentaire']);
  }
  else {
    $msgError = "Veuillez laisser un commentaire.";
  }
}
?>
```

```php
<!-- corps du document html -->
<!DOCTYPE html>
<html>
  <head>
    <title> Blog vulnérable </title>
  </head>
  <body>

    <!-- creation des commentaires du blog -->
    <h1> Mon blog </h1>
    <p> Laissez votre commentaire sur mon blog. Il apparaîtra sur cette page. </p>
    <form action="test_xssPersistant.php" method="post">
      <p> pseudo : <input class="input" name="pseudo" type="text" value=""> </p>
      <p> commentaire : <input class="input" name="commentaire" type="text" value=""> </p>
      <?php echo '<p>'.$msgError.'</p>';?>
      <input class="submit" name="submit" type="submit" value="valider">
    </form>

    <!-- affichage des commentaires du blog -->
    <?php afficherBdd() ?>
```

```php
  <!-- fonctions support du document -->
  <?php
  // affichage de la bdd
  function afficherBdd(){
    $conn = connectMaBase();
    $sql = 'SELECT * FROM tableCommentaires';
    $req = mysqli_query($conn, $sql);
    echo '<p> Les derniers commentaires : </p>';
    while($data = mysqli_fetch_array($req,MYSQLI_ASSOC)){
      echo "- ".$data['pseudo']." : ".$data['commentaire']."<br/>";
    }
    mysqli_free_result($req);
    mysqli_close($conn);
  }

  // connexion a la bdd
  // la base de donnees s'appelle databaseTest, la table s'appelle tableCommentaires,
  // les champs id, pseudo et commentaire, le formulaire alimente cette base
  function connectMaBase(){
    return mysqli_connect('localhost', 'root', 'root', 'databaseTest');
  }

  // insertion des donnees utilisateur dans la bdd
  function insererDonnees($name, $comment){
    $conn = connectMaBase();
    $sql = 'INSERT INTO tableCommentaires (pseudo, commentaire) VALUES("'.$name.'", "'.$comment.'")';
    mysqli_query($conn, $sql);
    mysqli_close($conn);
  }
  ?>
</body>
</html>
```

Après avoir testé le bon fonctionnement du code, nous avons introduit l'instruction JavaScript `<script>alert(1)</script>` dans le formulaire de commentaire. Seul le navigateur Mozilla Firefox a laissé le script s'exécuter pour obtenir la page suivante :



S'agissant de ce que nous voulions obtenir, nous avons alors cherché une CVE qui pourrait reproduire un résultat similaire, en utilisant un plug-in de WordPress et Mozilla Firefox qui semblait plus vulnérable, ce qui nous a conduit à la CVE 2017-10991.

ANALYSE DES RISQUES

Cette CVE n'entraîne aucun impact sur le contrôle d'accès, la confidentialité et la disponibilité du système.

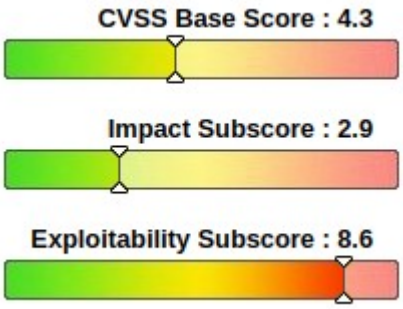
Concernant l'intégrité du système, l'impact est partiel car les modifications de quelques fichiers ou de certaines informations sont possibles. Toutefois, l'attaquant n'a pas de contrôle réel de ce qui peut être modifié et la portée de ses modifications est également limitée.

La complexité d'accès relative à la CVE est moyenne. En effet, les conditions d'accès sont assez spécifiques et doivent être respectées : l'administrateur doit être authentifié, connecté et doit interagir avec un mécanisme d'attaque en cliquant sur un lien.

La complexité de l'attaque est faible, car il s'agit essentiellement de créer un lien malveillant.

La CVE ne permet pas d'accéder à l'interface administrateur du site et l'authentification n'est pas requise de la part du hacker. En revanche, la vulnérabilité profite de l'authentification active de l'administrateur pour s'exécuter.

Le classement global de la vulnérabilité étudiée (Common Vulnerability Scoring System) est de 4.3 sur une échelle de 0 à 10.



EXPLOITATION DE LA VULNERABILITE

Vue de la page Web

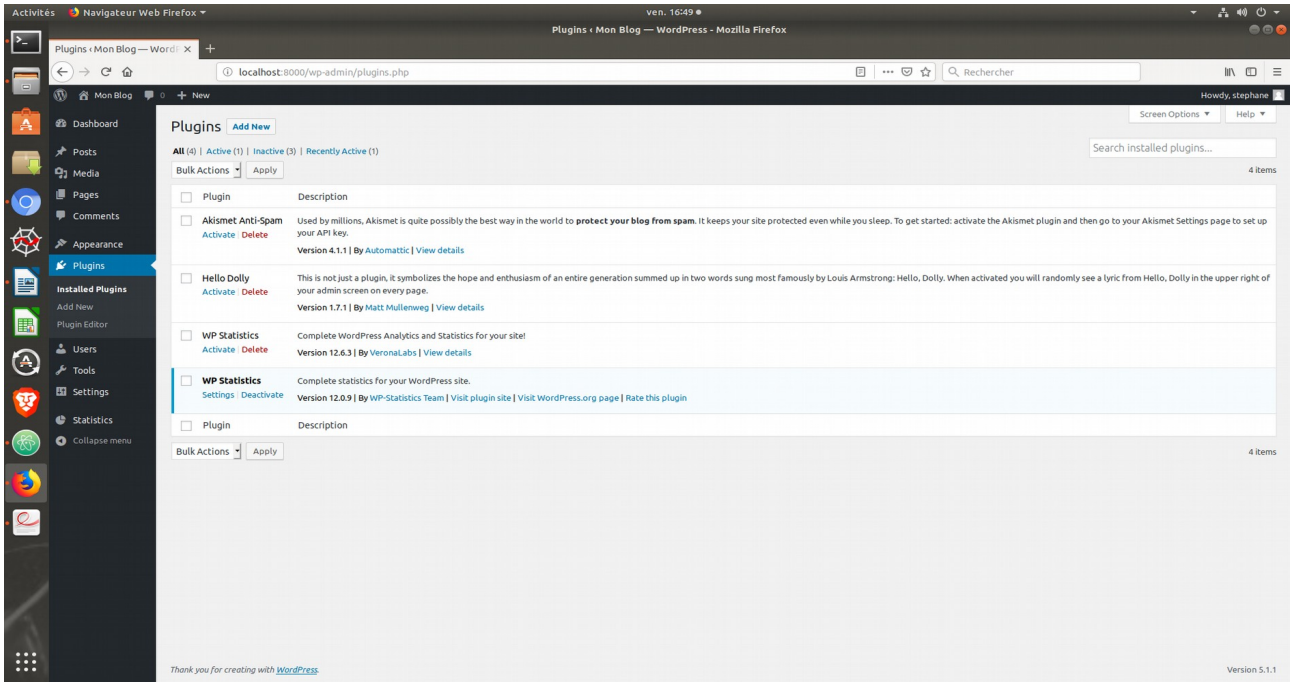
Mozilla Firefox était déjà présent sur notre ordinateur. Nous avons tout d'abord installé la dernière version de Wordpress en suivant les instructions de docker docs :

- création d'un fichier `docker-compose.yml`

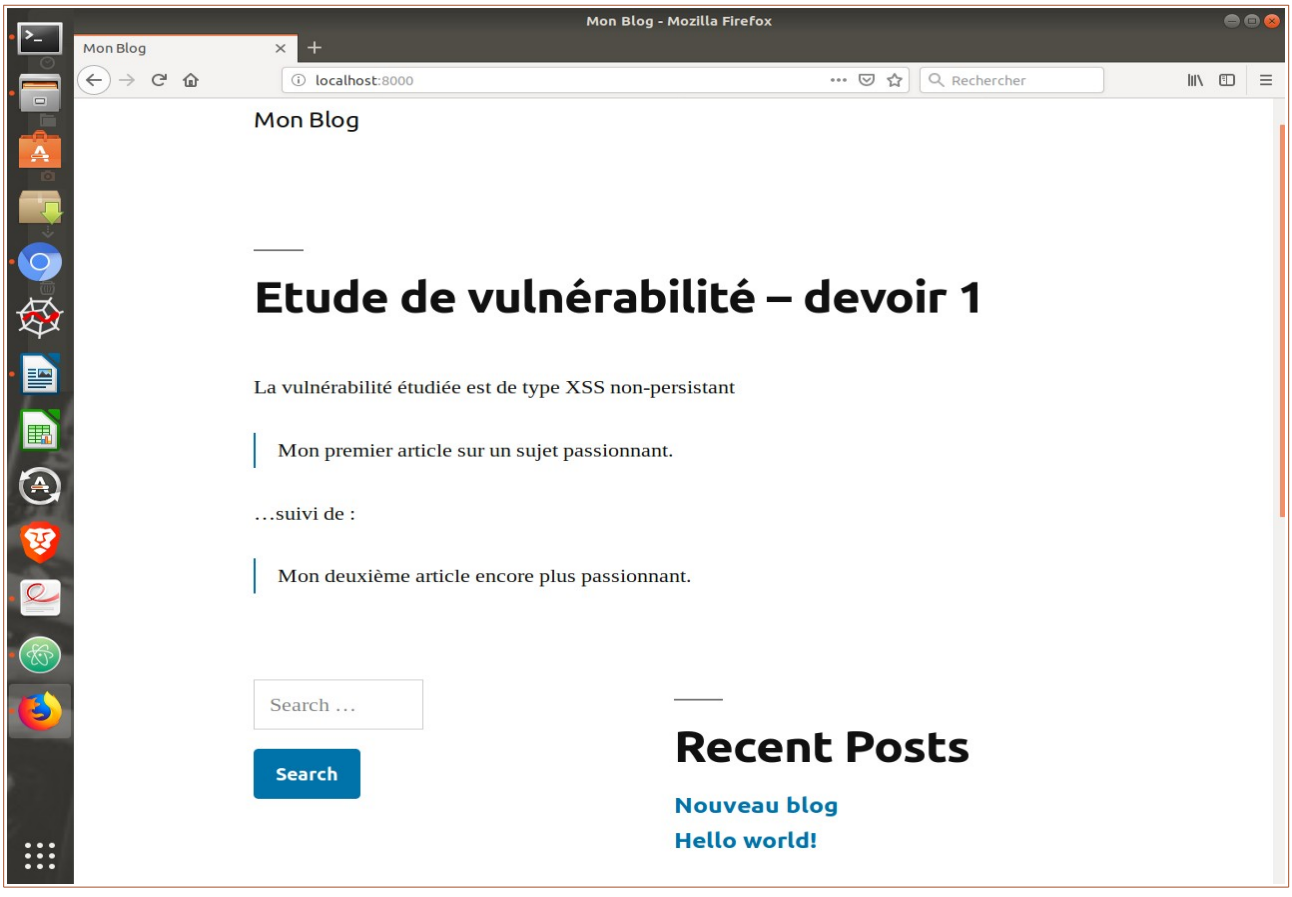
```yaml
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - ./db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     volumes:
       - ./wp_data:/var/www/html
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
```

- utilisation de l'instruction `docker-compose up -d` dans le Terminal.

Puis, nous avons téléchargé le plug-in WordPress vulnérable WP-Statistics 12.0.9 en utilisant la bibliothèque GitHub avec le Tag 12.0.9. Nous avons ensuite installé ce plug-in via l'interface administrateur du site WordPress, avant de l'activer.



Ce site est constitué sous forme de blog, présenté ci-dessous :



Détection de la vulnérabilité

Lorsque l'administrateur est authentifié et connecté à son site via le navigateur Mozilla Firefox, une attaque XSS se produit si il est amené à cliquer sur le lien malicieux suivant :

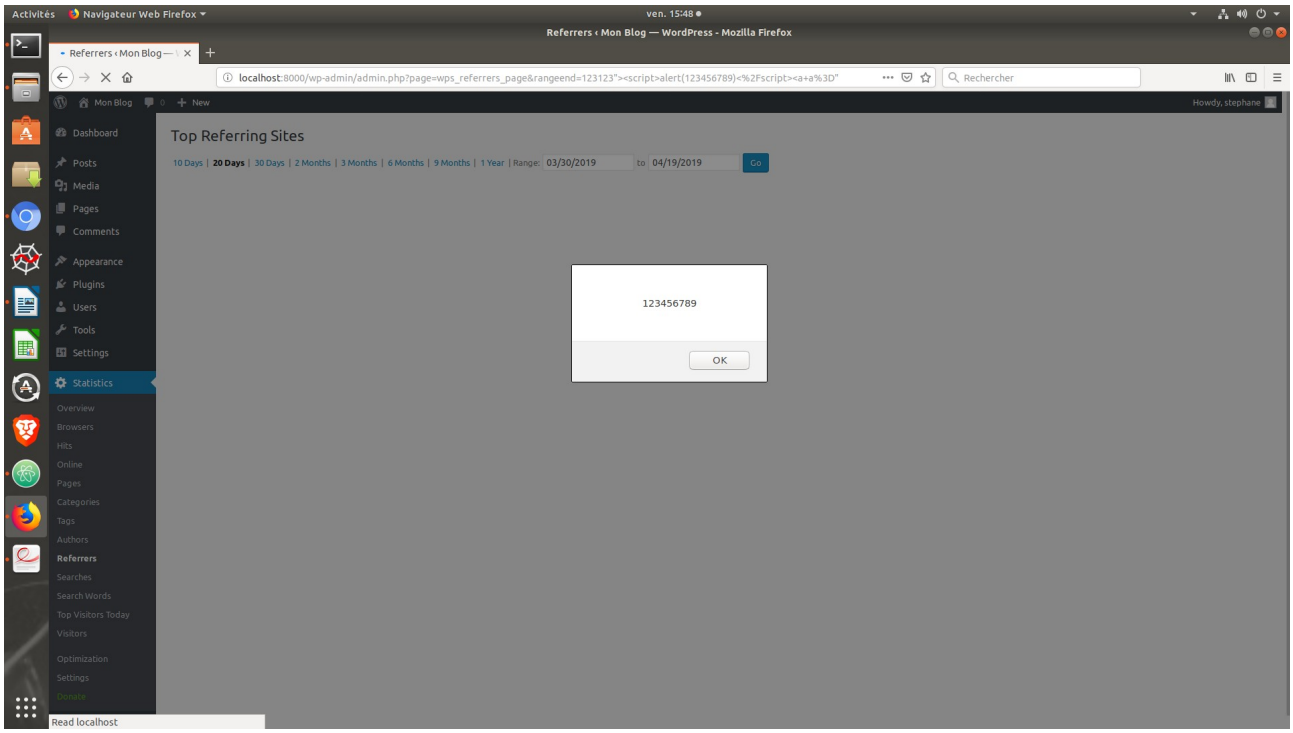
```
http://nomDuSiteVulnerable/wp-admin/admin.php?
page=wps_referrers_page&rangeend=123123"><script>alert(123456789)<
/script><a a="
```

Dans le cadre de notre travail, comme le site se trouve sur le serveur local, le lien à suivre sera :

http://localhost:8000/wp-admin/admin.php?page=wps_referrers_page&rangeend=123123"><script>alert(123456789)</script><a a="

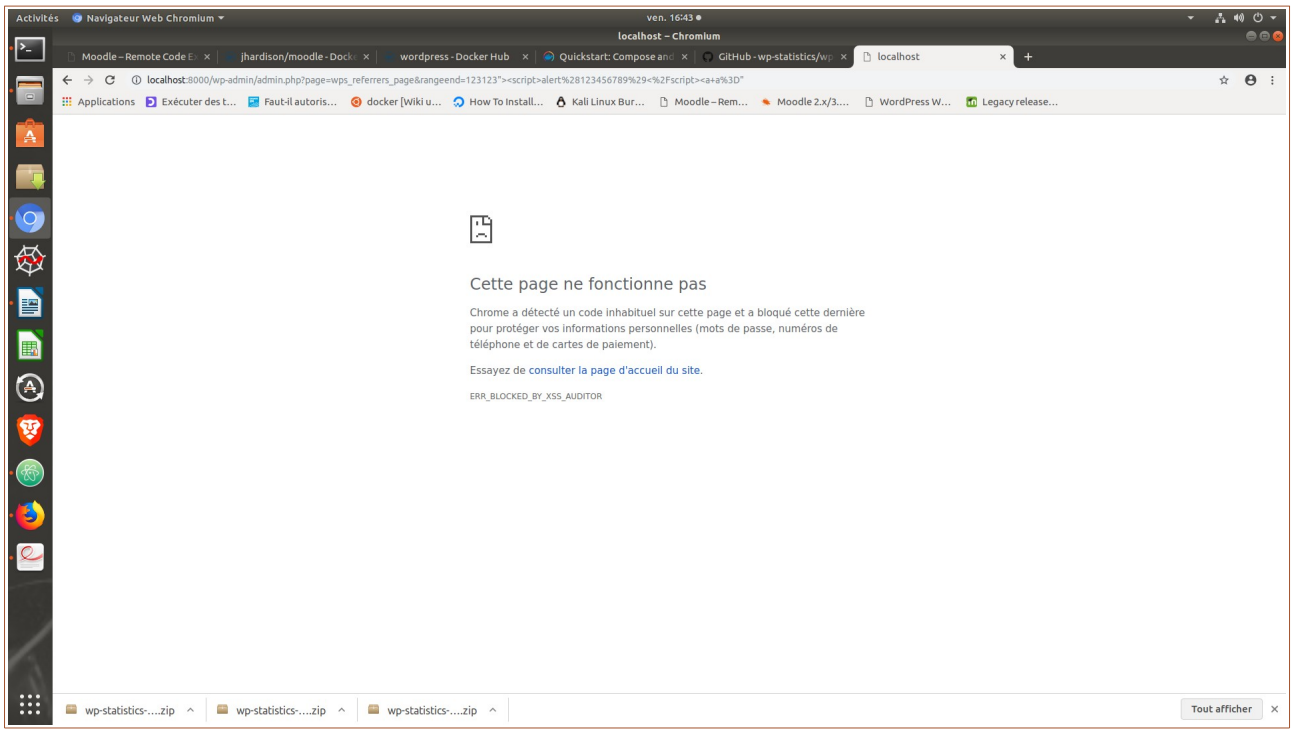
Il faut également s'assurer d'être bien connecté à l'interface administrateur à l'adresse `localhost:8000` en tant que `stephane` avec le mot de passe `chomolum1`, et que le navigateur par défaut soit Mozilla Firefox. En revanche, il n'est pas nécessaire d'être sur la page `wps_referrers_page` des statistiques du site pour que l'attaque se produise.

Celle-ci est démontrée ci-dessous avec l'apparition d'une fenêtre pop-up au sein du site Mon Blog :



Le point d'entrée utilisé par l'attaque est donc une méthode `GET` non sanitisée qu'on retrouve dans l'adresse du lien malicieux.

Nous vérifions par ailleurs que le script malicieux ne peut pas s'exécuter sous Chrome, Brave ou Safari. Dans chacun des cas, la page du site ne fonctionne plus. *(conf. impression écran ci-dessous)*



Correction de la vulnérabilité

La correction de la vulnérabilité a été faite à partir de la version 12.0.10 de WP-Statistics.

Le site WordPress.org présente la différence de codes sources entre les versions 12.0.9 et 12.0.10 du plug-in dans la rubrique Plugin Directory.

En rouge, apparaît ce qui a été retiré du code, en vert ce qui a été rajouté au code, le reste étant inchangé.

Unmodified Added Removed

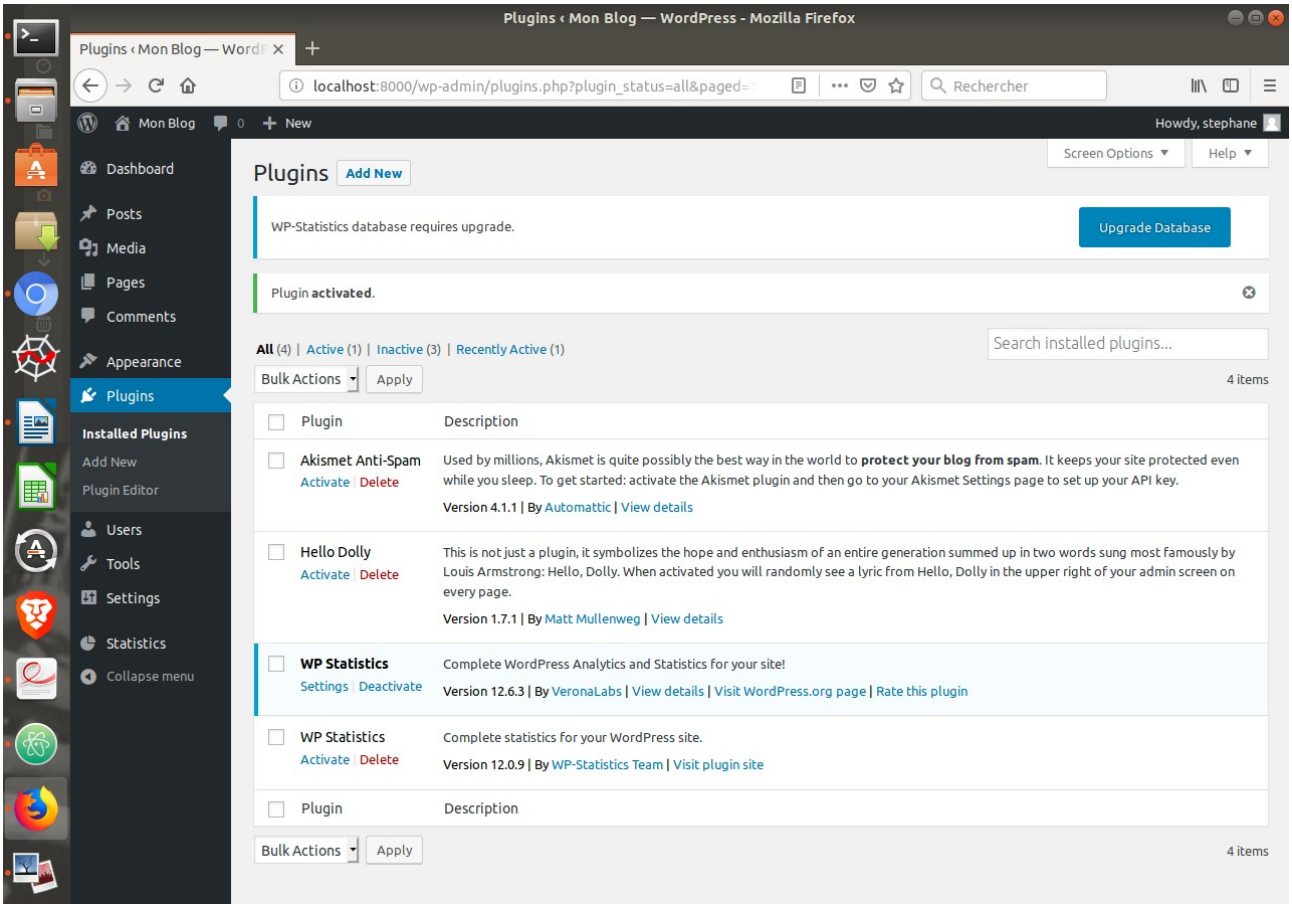
wp-statistics/tags/12.0.10/includes/log/top-referring.php — Tabular

r2068351 r2068351

```
12 12 $daysToDisplay = 20;
13 13 if ( array_key_exists( 'hitdays', $_GET ) ) {
14    -     $daysToDisplay = intval( $_GET['hitdays'] );
   14 +     $daysToDisplay = intval( esc_attr($_GET['hitdays']) );
15 15     $date_args     .= '&hitdays=' . $daysToDisplay;
16 16 }
17 17
18 18 if ( array_key_exists( 'rangestart', $_GET ) ) {
19    -     $rangestart = $_GET['rangestart'];
   19 +     $rangestart = esc_attr($_GET['rangestart']);
20 20     $date_args  .= '&rangestart=' . $rangestart;
21 21 } else {
... ...
24 24
25 25 if ( array_key_exists( 'rangeend', $_GET ) ) {
26    -     $rangeend  = $_GET['rangeend'];
   26 +     $rangeend  = esc_attr($_GET['rangeend']);
27 27     $date_args .= '&rangeend=' . $rangeend;
28 28 } else {
... ...
73 73 ?>
74 74 <div class="wrap">
75    -     <?php screen_icon( 'options-general' ); ?>
76 75     <h2><?php _e( 'Top Referring Sites', 'wp_statistics' ); ?></h2>
77 76
```

On remarque que la principale modification concerne le rajout de la fonction `esc_attr(string $text)` qui permet d'échapper les attributs html dans les formulaires et d'encoder les caractères spéciaux comme <, >, &, ″ , ‘. La description de cette fonction est donnée sur le site WordPress.org rubrique Code Reference.

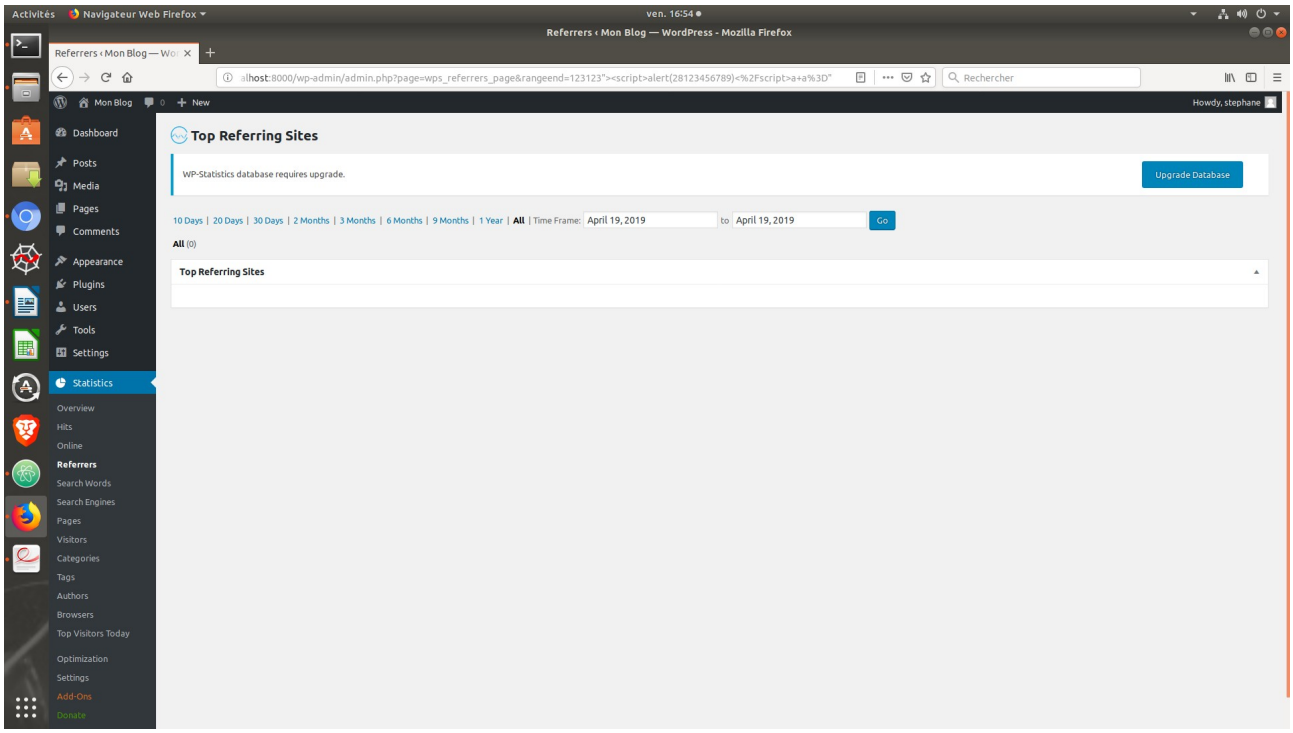
Pour vérifier le bon fonctionnement du plug-in corrigé, il suffit de charger de nouveau le plug-in à partir de la version 12.0.10. Nous avons choisi d'utiliser le plug-in WP-Statistics le plus récent car il est nativement proposé par l'interface administrateur. Il faut donc désactiver la version 12.0.9 et activer la dernière version (dans notre cas 12.6.3).



Une nouvelle tentative avec le lien malicieux

http://localhost:8000/wp-admin/admin.php?page=wps_referrers_page&rangeend=123123"><script>alert(123456789)</script><a a="

s'avère sans effet sur le navigateur Mozilla Firefox, montrant l'efficacité de la correction proposée. *(conf. impression écran ci-dessous)*



Reproduction de l'exploitation

La reproduction de cette exploitation peut se faire à l'aide de plusieurs images docker.

La première image contient l'application vulnérable.
Il s'agit de : `df.ctu.univ-fcomte.fr/srobin2-vuln1-img1-wp:unsafe`

La deuxième image contient la base de données de cette application vulnérable. Il s'agit de : `df.ctu.univ-fcomte.fr/srobin2-vuln1-img2-db:unsafe`

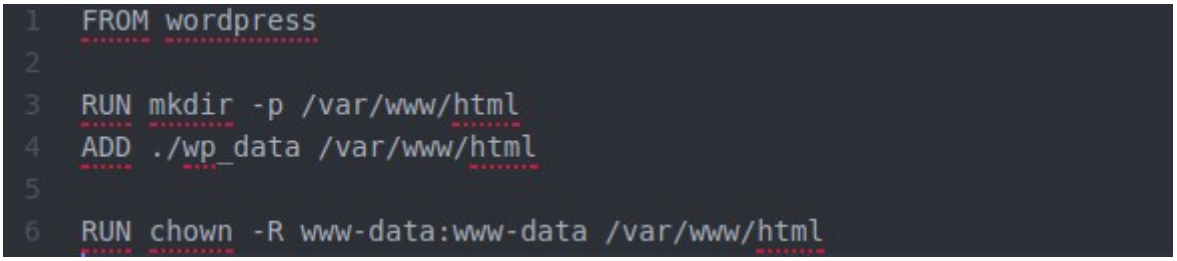
Le container docker permettant l'exploitation de la vulnérabilité peut s'exécuter à l'aide de l'instruction `docker-compose up -d`, à partir des deux images ci-dessus et du fichier `docker-compose.yml` suivant :

```
version: '2'

services:
   db:
     image: df.ctu.univ-fcomte.fr/srobin2-vuln1-img2-db:unsafe
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: df.ctu.univ-fcomte.fr/srobin2-vuln1-img1-wp:unsafe
     ports:
```

```
    - "8000:80"
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: wordpress
    WORDPRESS_DB_PASSWORD: wordpress
    WORDPRESS_DB_NAME: wordpress
```

On accède à l'application via l'URL http://localhost:8000 . Pour se connecter en tant qu'administrateur, il faut choisir comme identifiant `stephane` et comme mot de passe `chomolum1` .

Remarque sur la création de ces images : nous avons créé deux fichiers `Dockerfile_db` et `Dockerfile_wp` dans notre dossier `wordpress` afin de conserver la base de données construite. La première image a été obtenue à partir du fichier `Dockerfile_wp` et de l'instruction `sudo docker build -f Dockerfile_wp -t goyakla2015/srobin2-vuln1-img1-wp:unsafe .` :



```
FROM wordpress

RUN mkdir -p /var/www/html
ADD ./wp_data /var/www/html

RUN chown -R www-data:www-data /var/www/html
```

La deuxième image a été obtenue à partir du fichier `Dockerfile_db` et de l'instruction `sudo docker build -f Dockerfile_db -t goyakla2015/srobin2-vuln1-img2-db:unsafe .` :

```
FROM mysql

RUN mkdir -p /var/lib/mysql
ADD ./db_data /var/lib/mysql

RUN chown -R 999:root /var/lib/mysql
```

Les noms des images ont par la suite été modifiés par un `docker tag` pour être déposées dans le repository `df.ctu.univ-fcomte.fr` .

La troisième image contient l'application corrigée.
Il s'agit de : `df.ctu.univ-fcomte.fr/srobin2-vuln1-img1-wp:fixed`

La quatrième image contient la base de données de cette application corrigée.
Il s'agit de : `df.ctu.univ-fcomte.fr/srobin2-vuln1-img2-db:fixed`

Le container docker permettant de tester l'application corrigée peut s'exécuter à l'aide de l'instruction `docker-compose up -d`, à partir des deux images ci-dessus et du fichier `docker-compose.yml` suivant :

```
version: '2'
```

```
services:
   db:
     image: df.ctu.univ-fcomte.fr/srobin2-vuln1-img2-db:fixed
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: df.ctu.univ-fcomte.fr/srobin2-vuln1-img1-wp:fixed
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
```

En cas de problème pour extraire ces images, celles-ci existent également sur docker hub sous les noms de :
`goyakla2015/srobin2-vuln1-img1-wp:unsafe` pour la version vulnérable
`goyakla2015/srobin2-vuln1-img2-db:unsafe` pour la version vulnérable
`goyakla2015/srobin2-vuln1-img1-wp:fixed` pour la version corrigée
`goyakla2015/srobin2-vuln1-img2-db:fixed` pour la version corrigée

REFERENCES

CVE details,
https://www.cvedetails.com/cve/CVE-2017-10991/

CVE Mitre,
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-10991

National Vulnerability Database,
https://nvd.nist.gov/vuln/detail/CVE-2017-10991

Historique,
https://wpvulndb.com/vulnerabilities/8866

Plug-in WP-Statistics 12.0.9,
https://github.com/wp-statistics/wp-statistics/tree/12.0.9

Proof of Concept,
https://lorexxar.cn/2017/07/07/WordPress%20WP%20Statistics%20authenticated%20xss%20Vulnerability(WP%20Statistics%20-=12.0.9)/#Proof-of-Concept-PoC

Code source WP-Statistics 12.0.9,
https://plugins.trac.wordpress.org/browser/wp-statistics/tags/12.0.9/includes/log/top-referring.php

Code source WP-Statistics 12.1.0,
https://plugins.trac.wordpress.org/browser/wp-statistics/tags/12.1.0/includes/log/top-referring.php?rev=1737986

Différence de codes source, correction de la vulnérabilité,
https://plugins.trac.wordpress.org/changeset?old_path=%2Fwp-statistics%2Ftags%2F12.0.9%2Fincludes%2Flog%2Ftop-referring.php&old=2068351&new_path=%2Fwp-statistics%2Ftags%2F12.0.10%2Fincludes%2Flog%2Ftop-referring.php&new=2068351&sfp_email=&sfph_mail=

Fonction esc_attr(string $text),
https://developer.wordpress.org/reference/functions/esc_attr/

Wikipedia, WordPress,
https://fr.wikipedia.org/wiki/WordPress

WordPress, WP-Statistics,
https://wordpress.org/plugins/wp-statistics/

Installation de WordPress, suivant les instructions de docker docs,
https://docs.docker.com/compose/wordpress/

Wikipedia, Mozilla Firefox,
https://fr.wikipedia.org/wiki/Mozilla_Firefox