Ludovic Carré
Maxime Deloche
Vincent Lefoulon

Ensimag - 3A ISI
Janvier 2018

# Introduction à la sécurité des systèmes d'information
# Projet n° 2

## 1. *Origine de la vulnérabilité et exploitation*

WordPress est un logiciel de gestion de contenu (CMS) gratuit et open-source. Ecrit en PHP, il permet de facilement déployer et administrer un site de type blog sans écrire une ligne de code.

Dans la version 4.7, une API REST est mise à disposition pour interagir avec les éléments du site (pages, posts, utilisateurs, etc.) de façon programmatique.

Seulement, du fait d'une faille dans la vérification des permissions, il est devenu possible pour n'importe qui, via une requête API POST, de modifier un post quelconque, qu'il lui appartienne ou non.

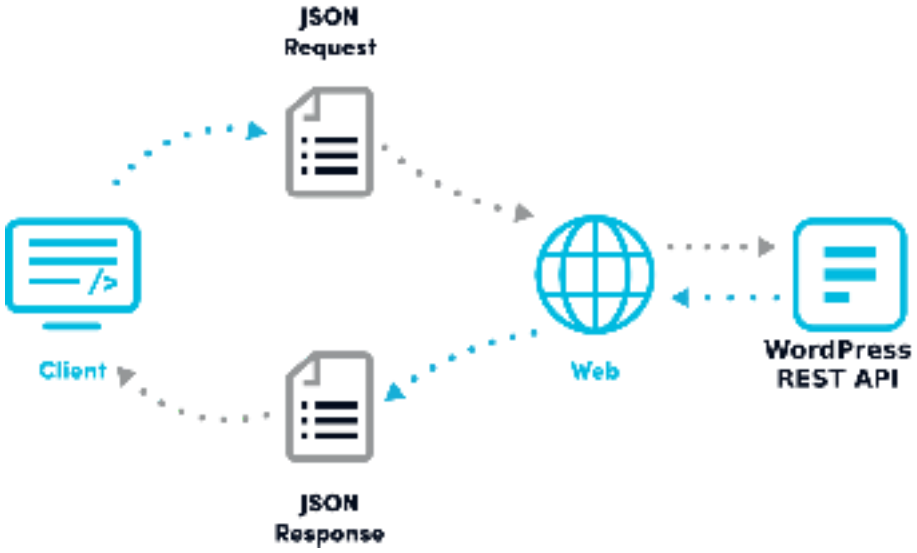
Les versions affectées sont les 4.7 et 4.7.1. Un patch a été introduit dans la version 4.7.2. La vulnérabilité est référencée par la **CVE-2017-1001000**.

Nous décrivons la faille en détail ici (fichier README du projet)..

## 2. *Scénario d'exploitation de la faille*

La faille est présente dans le logiciel WordPress, qui fait office de serveur. Plus précisément, elle apparaît dans le code traitant les requêtes de l'API REST.

L'architecture vulnérable est un simple serveur web (type Apache) transmettant les requêtes à WordPress, comme sur le schéma ci-dessous :




La faille se trouve au niveau de l'API REST de WordPress. Un utilisateur quelconque peut modifier le contenu d'un post de n'importe quel autre utilisateur en fournissant à la requête un id de post non numérique dans les paramètres. Cette procédure permet de passer outre la phase de vérification des permissions.

Exemple: POST /posts/111?id=123abc --data content="injected content"

L'utilisateur qui exécute cette requête va modifier le contenu de post 123 (et non du post 111) qu'il ait les droits ou non pour le faire.

## *3. Dispositions à prendre*

Pour limiter l'impact de cette faille, il est possible, voire nécessaire, de :

- Mettre à jour WordPress
- Garder un historique de la base pour éviter la suppression intempestive de données (comme sur Wikipédia)
- Mettre en place un système de vérification : envoyer un mail à l'auteur à chaque fois que le post est mis à jour pour qu'il soit averti si quelqu'un modifie un de ses posts

Il n'y avait de toute manière pas grand chose à faire parce que la faille était déjà corrigée au moment où elle a été rendue publique. Il suffisait donc simplement de mettre à jour WordPress.

Deux bonnes pratiques permettent de limiter les dégâts d'une telle faille : mettre à jour Wordpress dès que possible et effectuer des sauvegardes de sa base pour pouvoir la restaurer si elle est corrompue.

Il aurait été possible d'empêcher la faille durant la phase de développement en :

- Effectuant davantage de tests (?id=10ABC devrait lever une erreur)
- Mettant en place un système de peer-review : la fonction **`update_item_permission_check`** est très louche vu qu'elle ne lève pas d'erreur quand le post n'existe pas

## *4. Extrait de PSSI*

Cette faille présente un risque dès qu'une entreprise déploie un serveur web utilisant les versions 4.7 et 4.7.1 de WordPress : les posts du site pourraient alors être modifiés par un utilisateur malveillant et ainsi véhiculer de fausses informations, nuire à l'entreprise…

La PSSI (politique de sécurité du système d'information) est un document reflétant la vision d'une entreprise en matière de sécurité informatique, et détaille les enjeux, menaces et actions liés à ces problématiques de sécurité.

"Le service R&D de notre entreprise utilise WordPress pour communiquer sur les avancées technologiques et les projets sur lesquels travaillent nos ingénieurs.

Ces articles sont régulièrement consultés et cités par la communauté scientifique, et leur altération permettrait à des attaquant de nuire à la réputation de notre entreprise en diffusant des informations erronées sur nos travaux.

Afin de protéger au maximum nos données, la meilleure des pratiques est de tenir à jour les éléments de ce système et particulièrement WordPress, afin d'intégrer rapidement les dernières mises à jour de sécurité.

Les chercheurs veilleront également à enregistrer une sauvegarde de leurs publications de sorte à pouvoir restaurer leurs posts en cas d'exploitation de la faille."

## 5. *Expérimentation*

Note : en cas de problème, cloner ce dépôt et se référer à son README.

Installer **`docker-compose`**, un outil permettant de lancer et administrer plusieurs conteneurs Docker à partir d'un fichier de configuration.

Démarrer les services (MySQL et Wordpress) dans des conteneurs Docker :

```
docker-compose up
```

**Attention, un Wordpress vulnérable sera accessible sur le port `8080`.**

Dans un autre terminal, charger la base de données pré-remplie :

```
# Get the id of the mysql container
docker ps | grep dockerwordpresscontentinjection_db
./load_db.sh <mysql-container-id>
```

La base comporte deux auteurs. Récupérer les posts du premier :

```
./list_posts.sh author1
```

Modifier un post de cet auteur avec ce même auteur :

```
./update_post.sh 14 mysupercontent author1
```

On constate que le contenu a bien été mis à jour :

```
./get_post.sh 14
```

Lister les posts du deuxième auteur :

```
./list_posts.sh author2
```

Modifier de façon conventionnelle un post de l'auteur 2 avec l'auteur 1 :

```
./update_post.sh 17 mysupercontent author1
```

On obtient une erreur par manque de privilèges. Recommencer en exploitant la faille :

```
./exploit.sh 17 mysupercontent author1
```

Constater que le contenu a bien été modifié :

```
./get_post.sh 17
```

Arrêter les conteneurs :

```
docker-compose stop
docker-compose rm
```

# Glossaire

CMS : *un Content Management System est un logiciel, souvent une application web, permettant de créer et éditer dynamiquement des sites web depuis une interface graphique, sans écrire de code.*

API REST : *REST (Representational State Transfer) est un standard répandu d'API web. Une API REST permet d'interagir avec un service à partir de requêtes HTTP, donc notamment depuis un autre programme. Il est ainsi par exemple possible de poster un tweet depuis une application tierce, sans passer par l'interface graphique de Twitter.*

Query parameter : *paramètre passé lors du requête HTTP dans l'url. Par exemple, `id` est ici un query parameter : `GET /my/page?id=666`. Pour rendre les urls plus esthétiques, une pratique courante est d'intégrer les paramètres dans le corps de l'url : `GET /my/page/666`*

# Références


https://blog.sucuri.net/2017/02/content-injection-vulnerability-wordpress-rest-api.html

https://www.cvedetails.com/cve/CVE-2017-1001000/

Le dépôt de notre travail :
https://github.com/Vayel/docker-wordpress-content-injection