PHP TYPE JUGGLING

EXPLOITATION DE L'API REST DE WORDPRESS

LE LANGAGE PHP

- Langage interprété, **à typage dynamique**
- Massivement utilisé **côté serveur**
- Rendu populaire grâce au plugin Apache « **mod_php** »
- Utilisé par de nombreuses **applications web** et **CMS**
- De nombreuses failles **propres à PHP** (faille include ...)

Nous allons nous intéresser à la gestion du typage en PHP d'un point de vue sécurité...

“

Le typage dynamique consiste à laisser l'ordinateur réaliser l’opération de typage « à la volée », lors de l'exécution du code, contrairement à certains langages statiquement typés qui demandent au programmeur de déclarer expressément, pour chaque variable qu'il introduit dans son code, son typage.

LE TYPAGE DYNAMIQUE DE PHP

- Les variables sont effectivement typées, mais le langage déduit leur type de leur contenu
- Lorsque deux variables de types différents sont comparées ou lorsqu'une variable est castée, PHP se débrouille pour convertir la valeur
- Certaines de ces conversions sont un peu obscures et peuvent amener le développeur à introduire des failles dans le script

QUELQUES EXEMPLES

Comparaison string/nombre

```
"0000" == int(0) → TRUE
"0e12" == int(0) → TRUE
"1abc" == int(1) → TRUE
"0abc" == int(0) → TRUE
"abc"  == int(0) → TRUE
```

QUELQUES EXEMPLES

Comparaison string/string

```
"0e12345" == "0e54321" → TRUE
"0e12345" <= "1"       → TRUE
"0e12345" == "0"       → TRUE
"0xF" == "15"          → TRUE
```

L’ EXPLOITATION

→ De son doux nom : CVE-2017-1001000
→ Exploitation de l’API REST de Wordpress 4.7.0, 4.7.1 et 4.7.2
→ Exploite du PHP Type Juggling pour bypasser l’authentification et modifier le contenu
→ Permet un défacement des pages et une exécution de code dans certaines conditions

POST http://lesitewp.com/wp-json/wp/v2/posts/1
id=1&title=Article1&content=Bonjour

Vérification des permissions
update_item_permissions_check()

Mise à jour de l’article
update_items()

```php
public function update_item_permissions_check( $request ) {

        $post = get_post( $request['id'] );
        $post_type = get_post_type_object( $this->post_type );

        // Vérification : L'utilisateur a-t-il le droit de modifier l'article ?
        // Si NON -> Erreur
        // Vérification : L'utilisateur a-t-il le droit de modifier cet article, dont il n'est pas l'auteur ?
        // Si NON -> Erreur
        // Vérification : L'utilisateur a-t-il le droit d'épingler cet article ?
        // Si NON -> Erreur
        // Vérification : L'utilisateur a-t-il le droit d'assigner des mots clés à l'article ?
        // Si NON -> Erreur

        // Si aucune des conditions n'est vérifiée, on retourne True
        return true;
}
```

Premier problème :

Structure en liste noire : si aucune des conditions n'est vérifiée, on autorise

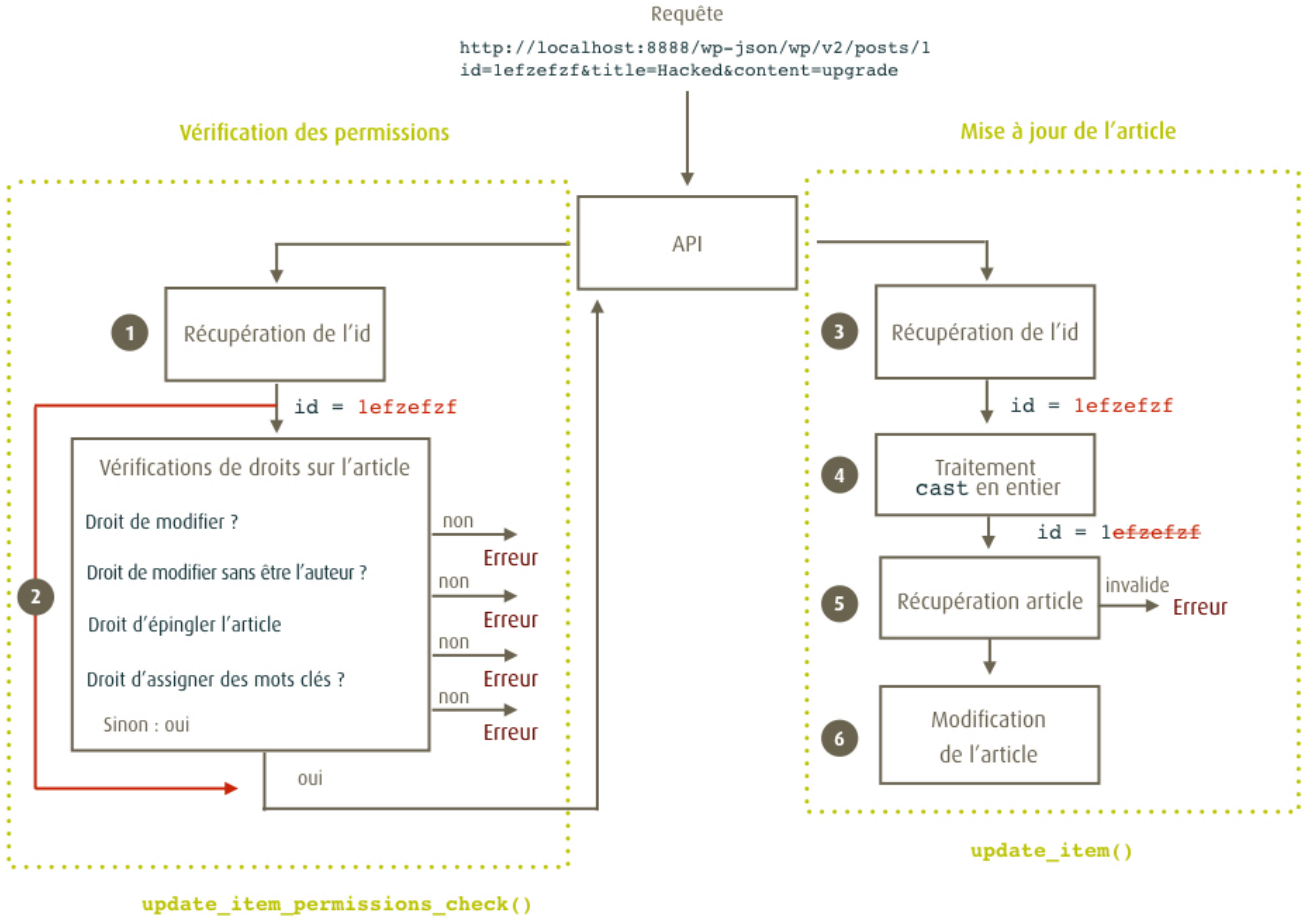
Si l'article n'existe pas, la fonction renvoie True !

```php
public function update_item( $request ) {
    $id   = (int) $request['id'];
    $post = get_post( $id );
    // Modification du post
    // ...
}
```

Second problème :

L’id est casté en entier avant d’être récupéré, et la fonction présuppose que l’utilisateur a le droit de modifier l’article

Modification d'un article (scénario d'exploitation)



1.4 à 1.8 millions
de pages web défacées

4 campagnes
de défacement massif

3 versions
touchées par la vulnérabilité

Bibliographie

11 questions pour comprendre la dernière vulnérabilité de l’API REST Wordpress [FR]
https://blog.xmco.fr/11-questions-pour-comprendre-la-derniere-vulnerabilite-de-lapi-rest-wordpress/

Content Injection Vulnerability in Wordpress [EN]
https://blog.sucuri.net/2017/02/content-injection-vulnerability-wordpress-rest-api.html

CVE-2017-1001000 sur CVEdetails.com [EN]
http://www.cvedetails.com/cve/CVE-2017-1001000

Proof Of Concept en Python sur le Github de leonjza [EN]
https://gist.github.com/leonjza/2244eb15510a0687ed93160c623762ab