Quelques aspects de sécurité

felipe@IFT3225 H2020

Ressources lues (GET)

- pages HTML, feuilles de style (css), scripts javascript externes, etc: En lecture pour tout le monde (user, other, group).
- répertoires dans lesquels se trouvent des ressources: En exécution pour tout le monde.

```
[felipe@arcade06,/u/felipe/wift3225]ls -l
-rw-r--r--  1 felipe rali   816 Jan 14 10:44 contacts.html
-rw-r--r--  1 felipe rali 40175 Apr  7 02:17 cours.html
-rw-r-----  1 felipe rali   955 Apr  7 02:41 demo10.html
-rw-r--r--  1 felipe rali 15727 Jan 14 10:50 demo1.html
-rw-r--r--  1 felipe rali  2477 Jan 22 18:05 demo2.html
-rw-r--r--  1 felipe rali  1254 Feb  7 10:57 demo3.html
-rw-r--r--  1 felipe rali  2360 Feb 14 01:16 demo4.html
-rw-r--r--  1 felipe rali  2518 Feb 27 09:44 demo5.html
-rw-r--r--  1 felipe rali   603 Jan  6 23:44 demo6.html
-rw-r--r--  1 felipe rali  1669 Mar 13 10:09 demo7.html
-rw-r-----  1 felipe rali  1002 Mar 25 23:43 demo8.html
-rw-r--r--  1 felipe rali  1880 Apr  3 13:21 demo9.html
-rw-r--r--  1 felipe rali  6438 Feb 18 11:16 devoir1.html
-rw-r--r--  1 felipe rali  8924 Mar 31 21:35 devoir2.html
-rw-r--r--  1 felipe rali  3476 Feb 18 10:48 form-w3c.html
-rw-r--r--  1 felipe rali  2579 Apr  7 02:18 frontal.php
-rw-r--r--  1 felipe rali   223 Jan  6 23:44 horaires.html
drwx--xr-x  2 felipe rali  4096 Mar  9 23:36 images
-rw-r--r--  1 felipe rali   158 Jan  6 23:44 index.html
-rw-r--r--  1 felipe rali  1651 Jan  6 23:44 intro.html
-rw-r--r--  1 felipe rali  6972 Jan  6 23:44 liens.html
-rw-r--r--  1 felipe rali  2657 Jan  6 23:44 livres.html
-rw-r--r--  1 felipe rali   634 Jan  6 23:44 menu_page.html
-rw-r--r--  1 felipe rali  2640 Jan  6 23:44 notes.html
-rw-r--r--  1 felipe rali   614 Jan  6 23:44 notes-info.html
-rw-r--r--  1 felipe rali  4962 Jan  6 23:44 notes-main.html
-rw-r--r--  1 felipe rali  3987 Apr  7 02:34 nouvelles.html
-rw-r--r--  1 felipe rali  2348 Jan  6 23:46 perso.css
-rw-r--r--  1 felipe rali  1938 Jan  7 00:46 plan.html
drwxr-xr-x 18 felipe rali 12288 Apr  6 18:18 ressources
...
```

on peut spécifier les priorités à l'aide d'une représentation binaire où: read vaut 4, write vaut 2 et execute vaut 1, ou bien en utilisant la notation **ugo** (user, group, other)

pas accessible → chmod og+r demo10.html

chmod 644 demo10.html

en Unix, un utilisateur appartient à un groupe, et il y a aussi les autres. Les priorités (read, write, exécution) d'accès sont fixées pour chacun. Pour accéder à un répertoire, il faut avoir les droits x.

accessible, mais plus de permission qu'il n'en faut →

chmod g-r images
chmod og-r ressources

chmod 711 images
chmod 711 ressources

- read et write pour user
- read pour le groupe
- read pour les autres

-rw-r--r-- 1 felipe rali 2518 Feb 27 09:44 demo5.html

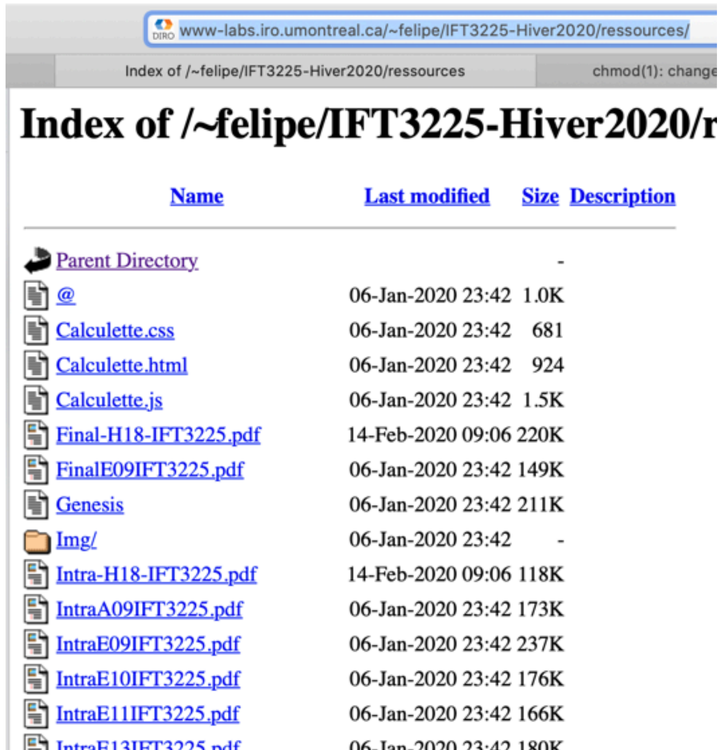
user groupe

Ressources lues (GET)

- pages HTML, feuilles de style (css), scripts javascript externes, etc: En lecture pour tout le monde (user, other, group).
- répertoires dans lesquels se trouvent des ressources: En exécution pour tout le monde.

[felipe@arcade06,/u/felipe/wift3225]ls -l

…
-rw-r--r-- 1 felipe rali 223 Jan 6 23:44 horaires.html
drwxr-xr-x 2 felipe rali 4096 Mar 9 23:36 images
drwxr-xr-x 18 felipe rali 12288 Apr 6 18:18 ressources
…



[felipe@arcade06,/u/felipe/wift3225]ls -l

…
-rw-r--r-- 1 felipe rali 223 Jan 6 23:44 horaires.html
drwx--x--x 2 felipe rali 4096 Mar 9 23:36 images
drwx--x--x 18 felipe rali 12288 Apr 6 18:18 ressources
…

un répertoire en exécution seulement n'est pas consultable en lecture: il faut connaitre le nom de la ressource (en lecture) pour y accéder.

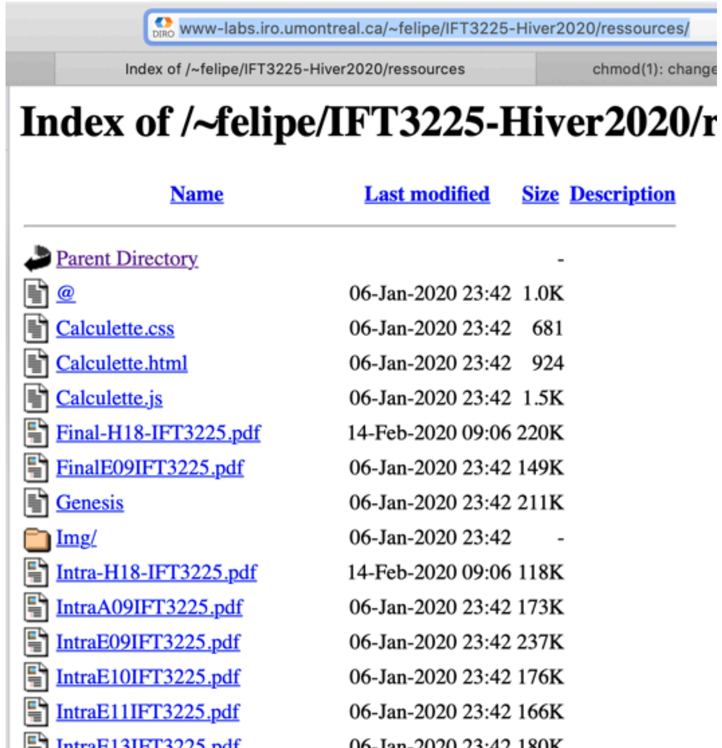
Ressources lues (GET)

- pages HTML, feuilles de style (css), scripts javascript externes, etc: En lecture pour tout le monde (user, other, group).
- répertoires dans lesquels se trouvent des ressources: En exécution pour tout le monde.

[felipe@arcade06,/u/felipe/wift3225]ls -l

...
-rw-r--r-- 1 felipe rali 223 Jan 6 23:44 horaires.html
drwxr-xr-x 2 felipe rali 4096 Mar 9 23:36 images
drwxr-xr-x 18 felipe rali 12288 Apr 6 18:18 ressources
...

si un répertoire ne contient pas de fichier index.html (configuration standard), et que le répertoire est ouvert en lecture (poru tout le monde), le navigateur crée un index des ressources lisibles (read) dans le répertoire demandé, ce qui n'est pas une bonne idée.



touch index.html
ls -l index.html
-rw-r----- 1 felipe rali 0 Apr 9 15:19 index.html

chmod og+r index.html
ls -l index.html
-rw-r—r-- 1 felipe rali 0 Apr 9 15:19 index.html

affiche une page vide (car ici index.html est une page vide)

Ressources lues (GET)

- pages HTML, feuilles de style (css), scripts javascript externes, etc: En lecture pour tout le monde (user, other, group).
- répertoires dans lesquels se trouvent des ressources: En exécution pour tout le monde.

[felipe@arcade06,/u/felipe/wift3225]cat index.html

```
<html>
<head>

<title>Redirection en html</title>
<meta http-equiv="refresh" content="0;URL=frontal.php?page=intro.html">
</head>

<body>
</body>

</html>
```

- mettre index.html (pour éviter l'index de lecture crée par le navigateur)
- en lecture pour tous (pour éviter l'affiche d'une erreur)
- et faire une redirection vers une page d'intérêt

[felipe@arcade06,/u/felipe/wift3225]ll | grep index
-rw-r--r-- 1 felipe rali   158 Apr 9 15:18 index.html

Les scripts PhP

sont en lecture (et écriture) seulement pour l'utilisateur.

```
-rw-------  1 felipe rali     779 Feb 24 10:02 cookies-send.php
-rw-r--r--  1 felipe rali   90238 Jan  6 23:42 cookies-utf8.pdf
-rw-r--r--  1 felipe rali    4996 Jan  6 23:42 cours-11-09-2009.html
-rw-------  1 felipe rali    2158 Feb 23 23:29 create-unigram-diro.php
```

Faille include

- Beaucoup de pages sont écrites en PhP, cela permet notamment de factoriser du code (par exemple avec `include` qui permet de charger une ressource particulière).
  - c'est par exemple le cas de la page de cours: frontal.php

bien sûr à ne pas faire en pratique …

```
<?php if (isset($_GET['source'])) die(highlight_file(__FILE__, 1)); ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf8" />
<meta name="Description" content="A free open source web design by Gen.  Free for anyone to use as long as credits are intact. " />
<meta name="Language" content="English" />

<title>Page d'accueil IFT 3225 </title>

<style type="text/css" title="layout" media="screen">
  @import url("style.css");
  @import url("perso.css");
</style>
</head>

<body>
<div id="wrapper">

<!-- titre de la page -->
<?php include("titre_page.html"); ?>

<!-- titre de la page -->
<?php include("menu_page.html"); ?>

<div id="content">

<!-- les nouvelles -->
<?php include("nouvelles.html"); ?>
…
```

les composants de la page

http://www-labs.iro.umontreal.ca/~felipe/IFT3225-Hiver2020/**frontal.php**?source

Faille include

Cool. Une fois un cadre de page bien défini, on peut vouloir offrir d'y mettre un contenu variable

```
include( $_GET['page'] );
```

frontal.php?page=cours.html

titre_page.html

menu_page.html

nouvelles.html

cours.html

frontal.php?page=devoir2.html

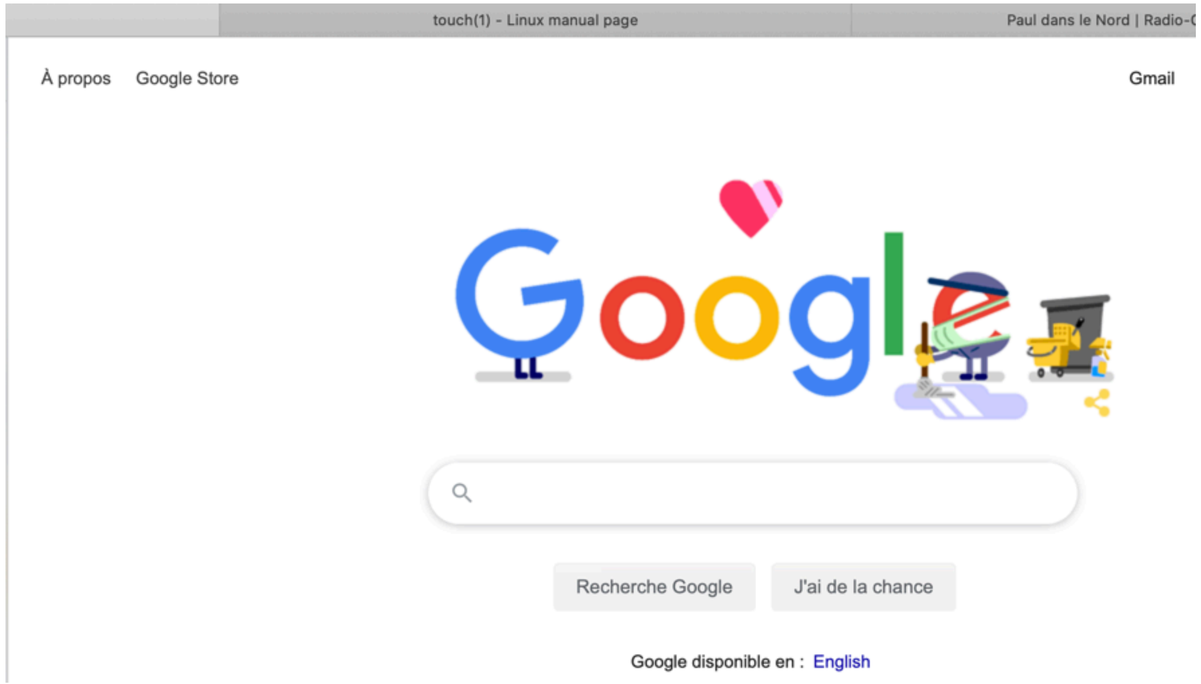
devoir2.html

Faille include

Problème: il s'agit d'un include calculé

```
include( $_GET['page'] );
```

http://www-labs.iro.umontreal.ca/~felipe/IFT3225-Hiver2020/frontal.php?page=togoogle.html




```
%more togoogle.html
<html>
<head>
<title>Redirection en html</title>
<meta http-equiv="refresh" content="0;URL=http://google.ca">
</head>
<body>
</body>
</html>
```

- il y a quelques années, cette URL effectuait directement la redirection sur les serveurs du DIRO:

```
http://www-labs.iro.umontreal.ca/~felipe/
IFT3225/frontal-withbug.php?page=http://
www.google.ca
```

- j'ai laissé ce script (avec faille include) en lecture pour illustrer le problème. Le serveur du DIRO a été attaqué et saturé dans la nuit…
  - utilisé pour faire plein de requêtes (venant de iro.umontreal.ca) à des serveurs dans le but de les ralentir.
- les "pros" sont à l'affût de failles includes
  - en repérant des paramètres particuliers dans l'URL et en générant des messages d'erreur lorsqu'on demande une ressource qui n'existe pas

Faille include

La parade est simple: vérifier la ressource demandée dans une liste

```php
<?php

    $pages = array("plan.html",
                   "livres.html",
                   "cours.html",
                   "travaux.html",
                   "horaires.html",
                   "devoir1.html",
                   "devoir2.html",
                   "demo1.html",
                   "demo2.html",
                   "demo3.html",
                   "demo4.html",
                   "demo5.html",
                   "demo6.html",
                   "demo7.html",
                   "demo8.html",
                   "demo9.html",
                   "demo10.html",
                   "notes.html",
                   "liens.html",
                   "tp1.html",
                   "tp2.html",
                   "contacts.html");

    $page = $_GET['page'];
    include(in_array($page,$pages)? $page : "cours.html");

?>
```

défaut

Injection de code (SQL)

Toujours le même problème: faire confiance à l'utilisateur !

Nous allons le voir dans les requêtes SQL calculées, mais faire une requête AJAX à un serveur via JSONP n'est pas moins dangereux.

On parle de cross-site-scripting (XSS)

```php
if (isset($_POST["login"]) && isset ($_POST["pass"])) {


  include('config-passwd.php');
  include('opendb-passwd.php');  // $conn is a my_sqli object

  // a buggy SQL request
  $sql = "SELECT login, secret FROM $db_table \
          WHERE login = '".$_POST["login"]."' AND secret = '".$_POST["pass"]."'";

  // just for debug
  echo "<table>";
  echo "<tr><td>requete: </td><td>".$sql."</td></tr>";
  echo "<tr><td>stripslashes: </td><td>".stripslashes($sql)."</td></tr>";
  echo "<tr><td>mysql_real: </td><td>".$conn->real_escape_string($sql)."</td></tr>";
  echo "<tr><td>mysql_real+strip: </td><td>".stripslashes($conn->real_escape_string($sql))."</td></tr>";
  echo "</table>";

  // requête a la base
  if ($res = $conn->query($sql)) {

    if (mysqli_num_rows($res) < 1)
      die ("Login/Pass incorrect !<br /><a href = 'injection-sql.php'>Retour</a>");
?>
```

logique claire: un formulaire envoie (POST) le login et le mot de passe saisi et les confronte à une base de données d'utilisateurs.

Injection de code (SQL)

```
$sql = "SELECT login, secret FROM $db_table \
        WHERE login = '".$_POST["login"]."' AND secret = '".$_POST["pass"]."'";
```

et soit l'input dans chaque champ: `a' OR 'a' = 'a`

la requête devient alors:

```
$sql = "SELECT login, secret FROM $db_table \
        WHERE login = '"."a' OR 'a' = 'a"."' AND secret = '"."a' OR 'a' = 'a"."'";
```

soit:

```
$sql = "SELECT login, secret FROM $db_table \
        WHERE login = 'a' OR 'a' = 'a' AND secret = 'a' OR 'a' = 'a'";
```

tjs vrai — tjs vrai

Parade

- vérifier les caractères spéciaux
  - anti-quote de certains caractères:
    - `mysql_real_escape_string`, `mysqli_real_escape_string`, `PDO::quote`
  - voir aussi:
    - `stripslashes`
- tester la présence de `<script>` (dans des attaques javascript)
- changer la logique de la BDD
  - ici, un login et un mot de passe devrait donner 0 (pas autorisé) ou 1 (autorisé) réponse. Mais la requête matche toutes les lignes (douteux !)

Injection de code (SQL)

```
$sql = "SELECT id_article, titre, contenu FROM $db_table \
        WHERE titre LIKE '".$_POST["search"]."%' \
           OR contenu LIKE '%" . $_POST["search"]."%'";
```

logique claire: afficher tous les produits dont le titre ou le contenu commence par la chaîne spécifiée par l'utilisateur (pourrait vous arriver dans le devoir #2).

mais avec l'input: `a' OR 'a%'='a`

on génère la requête:

```
$sql = "SELECT id_article, titre, contenu FROM $db_table \
        WHERE titre LIKE 'a' OR 'a%'='a%' \
           OR contenu LIKE '%a' OR 'a%'='a%'";
```

tjs vrai

toutes les lignes de la BDD *matchent.* *Anti-quoter l'input aurait prévenu l'attaque.*

Injection de code (SQL)

```
$sql = "SELECT id, titre FROM $db_table \
        WHERE id = '".$_POST["id"]."'";
```

logique claire: afficher le titre d'un article dont l'id est spécifié par un utilisateur

mais avec l'input: `0 or 1=1`

on génère la requête:

```
$sql = "SELECT id, titre FROM $db_table \
        WHERE id = '0 or 1=1'";
```

tjs vrai

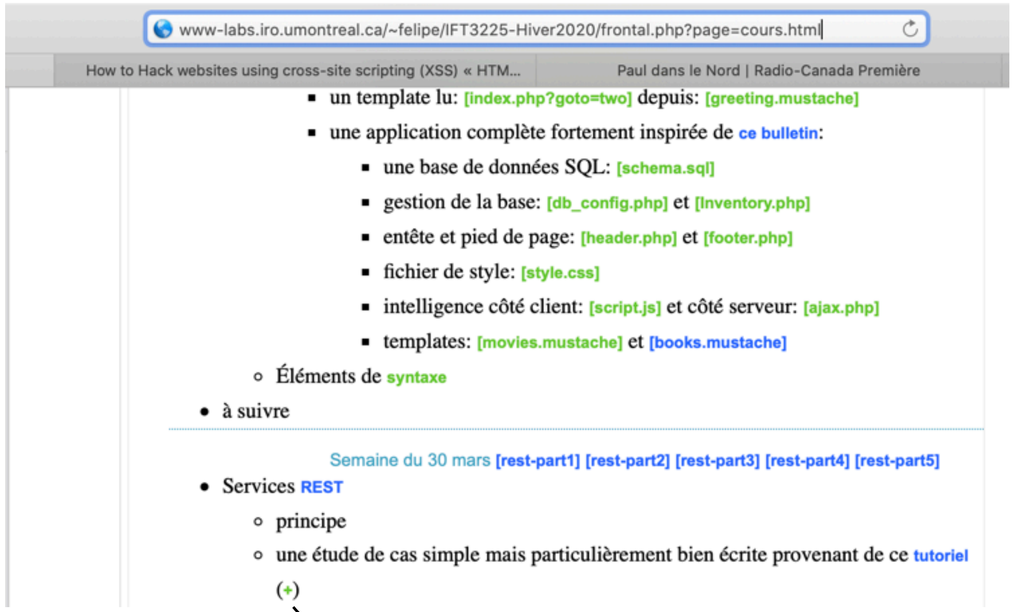
toutes les lignes de la BDD *matchent, sans guillemets ou autre caractère spécial. La logique SQL permet ici de parer l'attaque (ex: une seule ligne devrait être sélectionnée par requête)*

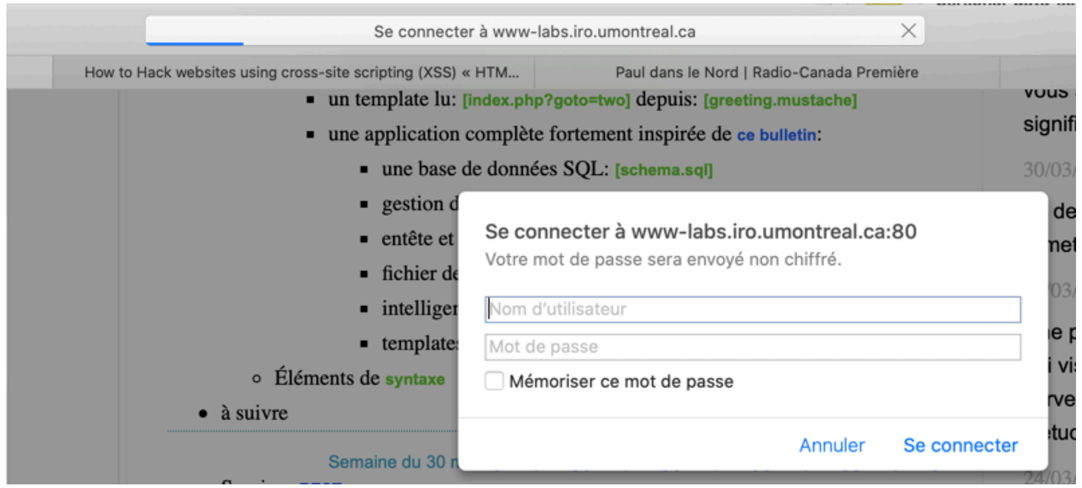
Les attaques XSS vous intéressent ? Regardez ce film

Contrôler l’accès à une ressource

Avec .htaccess (serveur Apache)

idée: lorsqu’un fichier .htaccess est présent dans un répertoire (ou sur le chemin) d’une ressource demandée, alors les vérifications demandées s’appliquent





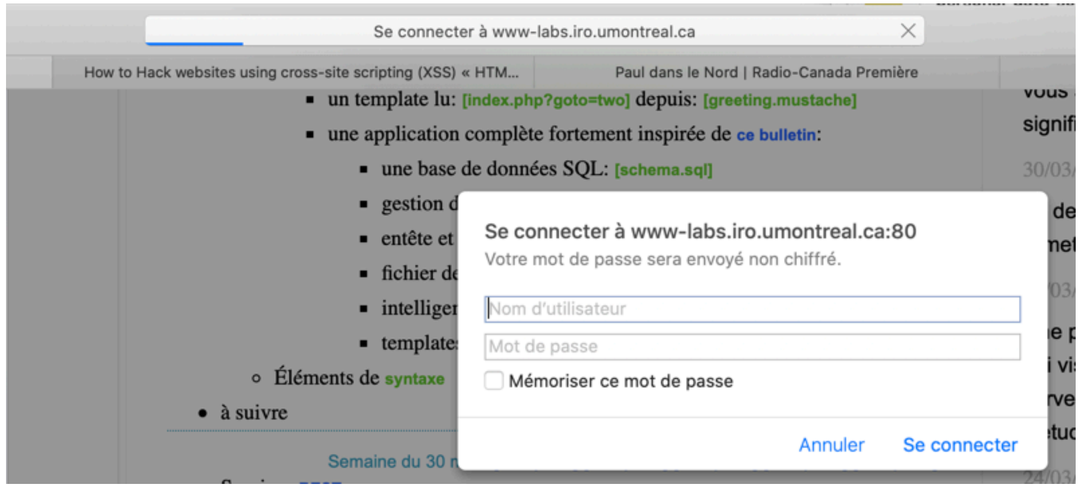
accès au code d’un tutoriel

http://www-labs.iro.umontreal.ca/~felipe/IFT3225-Hiver2020/ressources/__sekret/codeofaninja.tar.gz

Contrôler l’accès à une ressource

Avec .htaccess (serveur Apache)

ressources/__sekret/codeofaninja.tar.gz



le fichier de mots de passe ne devrait pas être accessible via HTTP (en dehors de *htdocs, public_html*)

```
[felipe@arcade06,/u/felipe/wift3225/ressources/__sekret]ls -l
-rw-r--r--  1 felipe rali 45330 Mar 30 23:50 codeofaninja.tar.gz
-rw-r--r--  1 felipe rali   155 Mar 30 23:43 .htaccess

[felipe@arcade06,/u/felipe/wift3225/ressources/__sekret]more .htaccess
AuthUserFile somewhere/.ift3225.pwd
AuthGroupFile /dev/null
AuthName TOC-TOC
AuthType Basic
<Limit  GET>
require user h20
</Limit>
```

endroit où se trouve le fichier de mots de passe

seul l’utilisateur h20 avec le bon mot de passe accèdera à la ressource

```
[felipe@arcade06]ll somewhere/.ift3225.pwd
-rw-r--r-- 1 felipe rali 84 Mar 30 23:43  ift3225.pwd
```

en lecture !

```
[felipe@arcade06]more somewhere/.ift3225.pwd
h18:$apr1$pHXfh7h1$lkSts5ZOCoXyUWMxTlawH
h20:$apr1$yG1fYcB7$SEsdDjcxyUE800R5omEiZ1
```

encrypté

Contrôler l’accès à une ressource

Créer un fichier de mots de passe avec:
`htpasswd -c <chemin> <user> <password>`

ex: chemin = somewhere/.ift3225.pwd

Ajouter un utilisateur:
`httpassword <chemin> user`

- **Note:** le fichier .htaccess doit être en lecture ugo.
  - En créer le moins possible, si possible dans la racine (htdocs, public_html)

- **Note**: l’usage de .htaccess ralentit l’accès à une ressource (puisqu’il faut lire les .htaccess sur le chemin de la ressource)

- Un guide compréhensible à .htaccess et des alternatives, un autre en français

- Lectures utiles:
  - https://httpd.apache.org/docs/2.4/fr/howto/htaccess.html
  - https://www.danielmorell.com/guides/htaccess-seo/basics/dont-use-htaccess-unless-you-must