

USN-4879-1: Linux kernel vulnerabilities

It was discovered that the Marvell WiFi-Ex device driver in the Linux kernel did not properly validate ad-hoc SSIDs. A local attacker could use this to cause a denial of service (system crash) or possibly execute arbitrary code. (CVE-2020-36158)

Loris Reiff discovered that the BPF implementation in the Linux kernel did not properly validate attributes in the getsockopt BPF hook. A local attacker could possibly use this to cause a denial of service (system crash). (CVE-2021-20194)

USN-4878-1: Linux kernel vulnerabilities

It was discovered that the Marvell WiFi-Ex device driver in the Linux kernel did not properly validate ad-hoc SSIDs. A local attacker could use this to cause a denial of service (system crash) or possibly execute arbitrary code. (CVE-2020-36158)

Ryota Shiga discovered that the sockopt BPF hooks in the Linux kernel could allow a user space program to probe for valid kernel addresses. A local

attacker could use this to ease exploitation of another kernel vulnerability. (CVE-2021-20239)

It was discovered that the priority inheritance futex implementation in the Linux kernel contained a race condition, leading to a use-after-free vulnerability. A local attacker could use this to cause a denial of service (system crash) or possibly execute arbitrary code. (CVE-2021-3347)

□□ discovered that the NFS implementation in the Linux kernel did not properly prevent access outside of an NFS export that is a subdirectory of a file system. An attacker could possibly use this to bypass NFS access restrictions. (CVE-2021-3178)

USN-4877-1: Linux kernel vulnerabilities

It was discovered that the Marvell WiFi-Ex device driver in the Linux kernel did not properly validate ad-hoc SSIDs. A local attacker could use this to cause a denial of service (system crash) or possibly execute arbitrary code. (CVE-2020-36158)

□□ discovered that the NFS implementation in the Linux kernel did not properly prevent access outside of an NFS export that is a

subdirectory of
a file system. An attacker could possibly use this to bypass
NFS access
restrictions. (CVE-2021-3178)

USN-4876-1: Linux kernel vulnerabilities

Olivier Benjamin and Pawel Wieczorkiewicz discovered a race condition the
Xen paravirt block backend in the Linux kernel, leading to a
use-after-free
vulnerability. An attacker in a guest VM could use this to
cause a denial
of service in the host OS. (CVE-2020-29569)
It was discovered that the Marvell WiFi-Ex device driver in
the Linux
kernel did not properly validate ad-hoc SSIDs. A local
attacker could use
this to cause a denial of service (system crash) or possibly
execute
arbitrary code. (CVE-2020-36158)

discovered that the NFS implementation in the Linux kernel
did not
properly prevent access outside of an NFS export that is a
subdirectory of
a file system. An attacker could possibly use this to bypass
NFS access
restrictions. (CVE-2021-3178)

Let's Spin up a Quick Python Server!

Today we're going to use a basic Python server to share files between multiple computers both quickly and easily, by using tools you likely have by default.

First, let's set the stage...

You're eight beers deep while sitting on your couch and playing with virtual machines. You have already downloaded the latest and greatest distro – but it's way over on your desktop and you're both lazy and not too sure about your walking ability!

What to do? What to do?

Sure, you could just download the file all over again – but you're aware of how much the project pays for bandwidth and you're inebriated enough so that if you don't do it right now then it just might not get done! So, you need that file and you need it with a quickness!

I'm going to make a huge assumption here. Someday, I'll write an article explaining how, but for now we're going to assume that you have SSH (secure shell) enabled on your desktop (in this scenario) and that you know how to use it. So, it's with a giant assumption and a leap of faith when I say that you've successfully used SSH to get to your desktop and you've already navigated to the directory where this latest and greatest distro image resides.

NOTE: If you don't have SSH enabled, surely you have access to a search engine. Go figure it out! Eventually I'll write an

article about it, but there are already hundreds of tutorials already out there.

Anyhow, sure... You've used SSH and now you could transfer the file with SCP (secure copy protocol) if you wanted. That's all well and good, but darn it we're aiming for the most contrived situation possible just so I can tell you how to spin up a server with Python! So, for whatever reason, you're hellbent on doing this in your browser. And do this in your browser you shall!

Yeah, there are a ton of ways to transfer files – many (perhaps all) of them better than this! Though you could also use this for testing your HTML and CSS skills. So, there's that! By the way, if my contrived scenario isn't good enough for you, you can make up your own reasons to do this. If your ideas are any good, you can even contribute to the site!

Anyhow, you're now connected to that desktop with SSH and you're in the directory where you store those important files. Now that you're there, run the following command:

```
[code]python -V[/code]
```

If you're using Python 1x or 2.x you should probably update, but the magic server command is this:

```
[code]python -m SimpleHTTPServer[/code]
```

If you're using Python 3.x then the command will be:

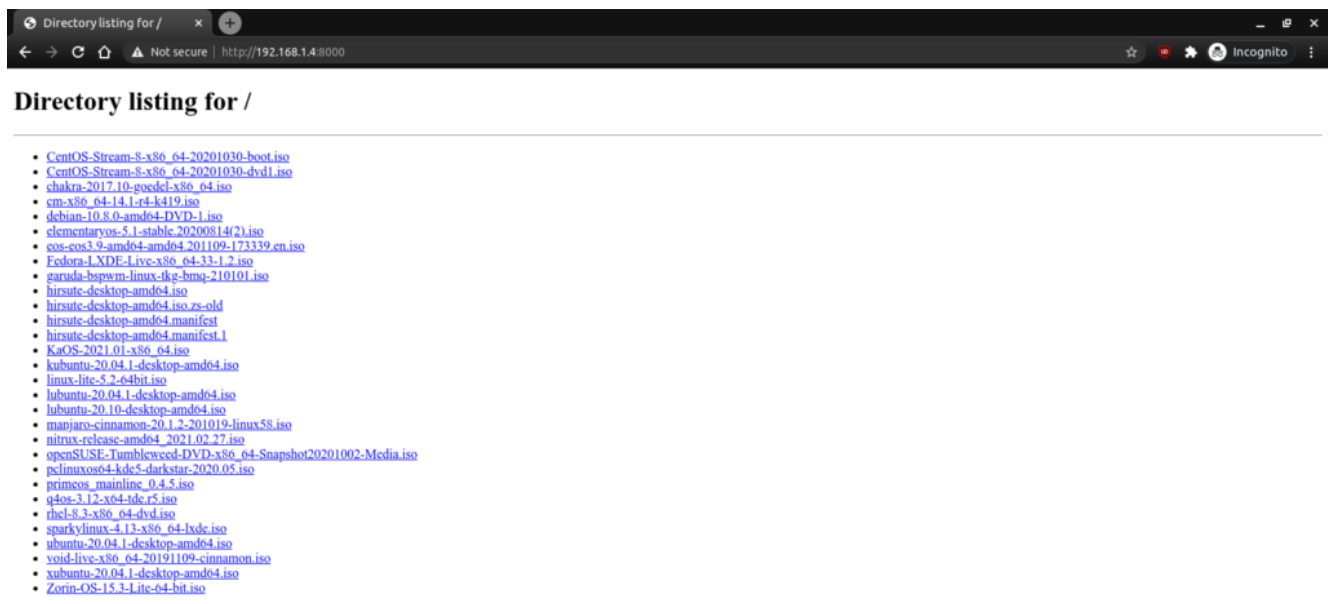
```
[code]python3 -m http.server[/code]
```

For the record, the '-m' is telling Python which module to load. Either way, you can now open your browser and enter this tidbit in the address bar:

```
[code]ip.address.of.desktop:8000[/code]
```

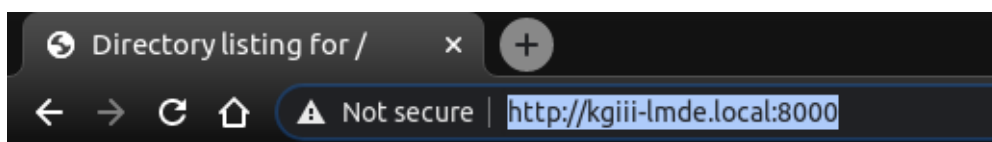
Obviously you should adjust it accordingly. If you've done it

right, it should look similar to this:



Obviously it will not look the same for you. It'll have your files!

If you want, you can also connect via the hostname. In the above example, the computer I was connected to is known as 'kgiii-lmde' and you need only add the .local for this to work. See the image below:



Directory listing for /

- [CentOS-Stream-8-x86_64-20201030-boot.iso](#)
- [CentOS-Stream-8-x86_64-20201030-dvd1.iso](#)
- [chakra-2017.10-goedel-x86_64.iso](#)
- [cm-x86_64-14.1-r4-k419.iso](#)
- [debian-10.8.0-amd64-DVD-1.iso](#)
- [elementaryos-5.1-stable.20200814\(2\).iso](#)
- [eos-eos3.9-amd64-amd64.201109-173339.en.iso](#)
- [Fedora-LXDE-Live-x86_64-33-1.2.iso](#)
- [garuda-bspwm-linux-tkg-bmq-210101.iso](#)
- [hirsute-desktop-amd64.iso](#)

See? No IP address required! You can also use

this for the above mentioned SSH!

Now, if you want to do so, you can also change the port number. This is the same for both commands. In both cases, just add your chosen port number at the end. Like so:

```
[code]python3 -m http.server 9000[/code]
```

And, again, it should look a bit like this:



Directory listing for /

- [CentOS-Stream-8-x86_64-20201030-boot.iso](#)
- [CentOS-Stream-8-x86_64-20201030-dvd1.iso](#)
- [chakra-2017.10-goedel-x86_64.iso](#)
- [cm-x86_64-14.1-r4-k419.iso](#)
- [debian-10.8.0-amd64-DVD-1.iso](#)
- [elementaryos-5.1-stable.20200814\(2\).iso](#)
- [eos-eos3.9-amd64-amd64.201109-173339.en.iso](#)
- [Fedora-LXDE-Live-x86_64-33-1.2.iso](#)
- [garuda-bspwm-linux-tkg-bmq-210101.iso](#)
- [hikuuta-desktop-amd64.iso](#)

Note the changed port number. You should probably avoid reserved ports.

Anyhow, you can use this for all sorts of things. You can now navigate the user's files with a web browser. If you're really insane, you can make this available over the internet by enabling port forwarding – but I'm gonna suggest you not do that. If you're going to do that, you should probably use something more robust and with much finer permission controls. It'd also take quite a bit of work to turn this into a server that'd do anything more than offer up static files. I suspect (but don't know) that it's about as secure as a screen door, so putting this on the public web would just be silly. Don't do that.

Though my written use-case was very contrived, I find myself

using this easy server fairly regularly. I actually prefer to use FTP to move files between computers, but I'll spin up a quick Python server so that I can grab things like config files and whatnot. Believe it or not, even wget works. In this instance, if I wanted to grab that Debian ISO, I could just do this:

```
[code]wget  
http://kgiii-lmde.local:9000/debian-10.8.0-amd64-DVD-1.iso[/code]
```

So long as the server is up and running, wget works. Granted, there's not too many contrived situations where I'd need to use wget on top of all this – but it's fun to play with and educational.

What sort of uses can you come up with? Feel free to leave them as comments below. Who knows? Maybe they'll end up being motivations for future articles?

As always, thanks for reading. Don't forget to sign up for the newsletter. Chances are good that I'll not always be on the forums and this is a way for you to keep track of what gets published. I also won't send you any spam, nor will I trade/sell/give your email address to anyone.