USN-4746-1: xterm vulnerability

Tavis Ormandy discovered that xterm incorrectly handled certain character
sequences. A remote attacker could use this issue to cause xterm to crash,
resulting in a denial of service, or possibly execute arbitrary code.

USN-4698-2: Dnsmasq regression

USN-4698-1 fixed vulnerabilities in Dnsmasq. The updates introduced
regressions in certain environments related to issues with multiple
queries, and issues with retries. This update fixes the problem.
Original advisory details:

Moshe Kol and Shlomi Oberman discovered that Dnsmasq incorrectly handled
memory when sorting RRsets. A remote attacker could use this issue to cause
Dnsmasq to hang, resulting in a denial of service, or possibly execute
arbitrary code. (CVE-2020-25681, CVE-2020-25687)

Moshe Kol and Shlomi Oberman discovered that Dnsmasq incorrectly handled
extracting certain names. A remote attacker could use this issue to cause

Dnsmasq to hang, resulting in a denial of service, or possibly execute arbitrary code. (CVE-2020-25682, CVE-2020-25683)

Moshe Kol and Shlomi Oberman discovered that Dnsmasq incorrectly implemented address/port checks. A remote attacker could use this issue to perform a cache poisoning attack. (CVE-2020-25684)

Moshe Kol and Shlomi Oberman discovered that Dnsmasq incorrectly implemented query resource name checks. A remote attacker could use this issue to perform a cache poisoning attack. (CVE-2020-25685)

Moshe Kol and Shlomi Oberman discovered that Dnsmasq incorrectly handled multiple query requests for the same resource name. A remote attacker could use this issue to perform a cache poisoning attack. (CVE-2020-25686)

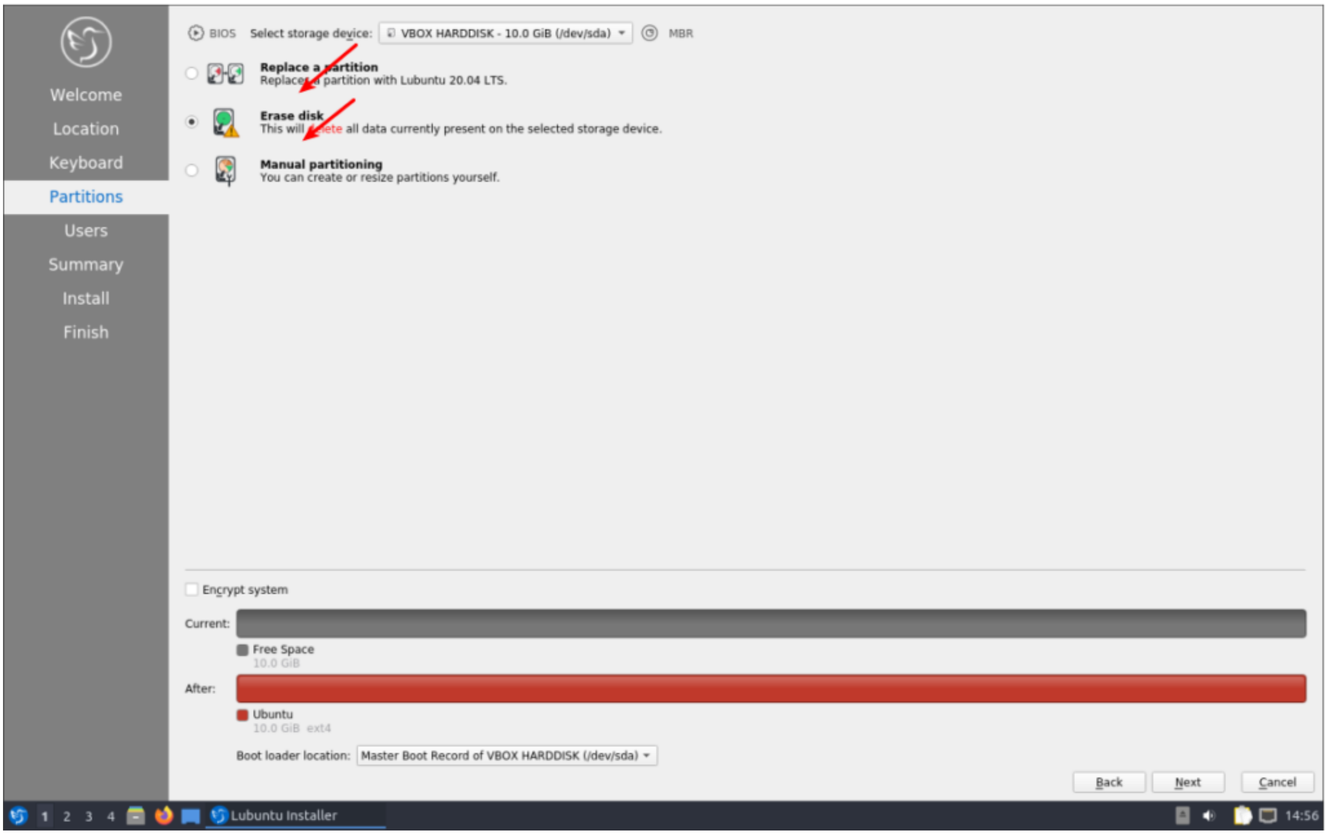
It was discovered that Dnsmasq incorrectly handled memory during DHCP response creation. A remote attacker could possibly use this issue to cause Dnsmasq to consume resources, leading to a denial of service. This issue only affected Ubuntu 16.04 LTS, Ubuntu 18.04 LTS, and Ubuntu 20.04 LTS. (CVE-2019-14834)

USN-4745-1: OpenSSL vulnerabilities

David Benjamin discovered that OpenSSL incorrectly handled comparing
certificates containing a EDIPartyName name type. A remote attacker could
possibly use this issue to cause OpenSSL to crash, resulting in a denial of
service. (CVE-2020-1971)
Tavis Ormandy discovered that OpenSSL incorrectly handled parsing issuer
fields. A remote attacker could possibly use this issue to cause OpenSSL to
crash, resulting in a denial of service. (CVE-2021-23841)

How To: Enable a Swapfile

Yeah, yeah… I have a modern, large SSD. I have more RAM than I'll ever possibly use. I still want/enable swap, in this case a swapfile. Imagine my dismay when I installed Lubuntu 20.04 and found there's no swap available during the basic installation? (It's there in 21.04.)



BIOS Select storage device: VBOX HARDDISK - 10.0 GiB (/dev/sda) MBR

○ **Replace a partition**
Replaces a partition with Lubuntu 20.04 LTS.

● **Erase disk**
This will delete all data currently present on the selected storage device.

○ **Manual partitioning**
You can create or resize partitions yourself.

□ Encrypt system

Current: Free Space 10.0 GiB

After: Ubuntu 10.0 GiB ext4

Boot loader location: Master Boot Record of VBOX HARDDISK (/dev/sda)

See? There's nothing there!

Sure, I could have made a swap partition during the installation, but that didn't seem like something I wanted to do – besides, I can always add it later. Which, of course, I did.

There are all sorts of views about whether swap is required in my situation, or in any situation, but I'm of the mind that disk space is cheap and my computer is faster than I am. So, if it has any chance of helping, it's all good. It should also be mentioned that swap is far more complicated than 'a place where the kernel sticks stuff when there's no more RAM left'. In fact, it's a lot more complicated than that. It's where the kernel pages content that's seldom used, and it'll happily use swap even when there's plenty of RAM available.

Since it's lacking, let's learn how to add a swapfile to Ubuntu (and official flavors) 20.04 and presumably other similarly aged variants. It's a pretty painless process.

Like normal, let's crack open your terminal emulator with CTRL

+ ALT + T.

Now, let’s check to see if you’ve already got some swap going on.

```
[code]swapon –show[/code]
```

If it shows nothing but a new line, you have no swap. If it says anything else, you’ve got swap enabled already and probably don’t need this article.

Just so you know, I personally just did this a couple of days ago, after upgrading to Lubuntu 20.04. So, I’m pulling this data from .bash_history.

Let’s make a swapfile.

```
[code]sudo fallocate -l 8G /swapfile[/code]
```

Why 8 gigabytes when I have ample RAM and an SSD? Because I never, ever want to worry about it again. I want to be able to open up every app I have and leave them open for a month. You do you and decide how big you want it to be!

Now, we need to set some permissions. We don’t want anyone writing to swap, we only want root writing to swap.

```
[code]sudo chmod 600 /swapfile[/code]
```

Next, we need to let the OS know that’s swap space.

```
[code]sudo mkswap /swapfile[/code]
```

And turn it on with:

```
[code]sudo swapon /swapfile[/code]
```

And you now have swap in the form of a swapfile and it’s turned on. I suppose we should make this permanent. To do this, we need to edit fstab and nano is a good tool for this.

```
[code]sudo nano /etc/fstab[/code]
```

And add this at the bottom of that document:

[code]/swapfile none swap sw 0 0[/code]

Those are the 0 digit, in case the font here makes it confusing. (I think I'll try messing with the fonts.)

Either way, you should now have a swapfile that gets loaded on reboot and is currently loaded and working. You can next edit the swappiness value. In Ubuntu, it is a default of 60. If you want to edit it, you'll have to wait for another article.

Like always, thanks for reading. It's missing at the moment, or not working, or I'd say subscribe and get notifications of new articles. However, I'll have to work on that. I just haven't made the time to do so.

USN-4467-3: QEMU regression

USN-4467-1 fixed vulnerabilities in QEMU. The fix for CVE-2020-13754
introduced a regression in certain environments. This update fixes the
problem.
We apologize for the inconvenience.

Original advisory details:

Ren Ding, Hanqing Zhao, Alexander Bulekov, and Anatoly Trosinenko
discovered that the QEMU incorrectly handled certain msi-x mmio operations.
An attacker inside a guest could possibly use this issue to cause QEMU to
crash, resulting in a denial of service. (CVE-2020-13754)