snapd vulnerable to Library Injection from CWD

Bug #1901572 reported by itszn on 2020-10-26

This bug affects 1 person 262

Affects	Status	Importance	Assigned to	Milestone
Snapcraft	Fix Released	Critical	Sergio Schvezov	
snapcraft (Ubuntu)	Fix Released	High	Emilia Torino	

Bug Description

I discovered that snapcraft will include the current directory as a search path for LD_LIBRARY_PATH when the home plug is enabled.
This means any snap package with the home plug, which are command line can be taken over if the user has a malicious file or directory in their CWD.

This was found when a snap package crashed due to libc.so.6 being in the CWD search path:

```
itszn@ubuntu:snap-escape$ ls tls/libc.so.6
-rw-rw-r-- 1 itszn itszn 2030544 Oct 25 11:46 tls/libc.so.6
itszn@ubuntu:snap-escape$ vlc # random snap package
Segmentation fault (core dumped)
```

We can figure out why this is happening by checking strace:

```
  $ sudo strace -f -v -e execve -s 1000 sudo -u itszn vlc 2>&1 | egrep
'LD_LIBRARY_PATH=[^"]+"' -o | head -n 1
  LD_LIBRARY_PATH=/var/lib/snapd/lib/gl:/var/lib/snapd/lib/gl32:/var/
lib/snapd/void:/snap/vlc/1700/lib/x86_64-linux-gnu:/snap/vlc/1700/usr/lib/
x86_64-linux-gnu::/snap/vlc/1700/lib:/snap/vlc/1700/usr/lib:/snap/vlc/
1700/lib/x86_64-linux-gnu:/snap/vlc/1700/usr/lib/x86_64-linux-gnu"
```

In the above variable, we see a :: this is the result of an empty env variable being used. This leads to strange behavior in ld where it includes cwd in the search path (which I might also personally consider buggy).

The extra : is probably due to an empty path passed to either https://github.com/snapcore/snapcraft/blob/master/snapcraft/internal/project_loader/_env.py#L43 or https://github.com/snapcore/snapcraft/blob/master/snapcraft/internal/elf.py#L64

This bug means that any library not provided by the first search paths will be loaded from CWD, including libc.so.6

A malicious libc.so.6 could be crafted to get code exec within the container when loaded. Additionally this file could be hidden in a subdirectory named either tls or x86_64.

This code execution can then abuse other plugs, such as the x11 plug, to interact with the rest of the system and run a script by sending key presses to an open terminal.

I have included a poc which will work with snaps using core-18 with home and x11 plugs. (for example VLC and Chromium)

The impact of this is large as it effects the most installed snaps (based on https://snapcraft.io/blog/popular-snaps-per-distro) are vulnerable and include apps specifically made to open files from the cli. Additionally chromium is now shipping as a snap on ubuntu 20.04, meaning users who normally would not use snap are now becoming vulnerable to this.

POC:
Example running vlc in the POC directory:

```
--------------------------------------
itszn@ubuntu:~$ tar xfvz snap-escape.tar.gz
itszn@ubuntu:~$ cd snap-escape
itszn@ubuntu:snap-escape$ ls
total 8
-rw-rw-r-- 1 itszn itszn 0 Oct 25 11:04 amazing-movie.mp4
-rw-rw-r-- 1 itszn itszn 0 Oct 25 11:28 cool-page.html
-rw-rw-r-- 1 itszn itszn 2193 Oct 25 11:45 README.txt
drwxrwxr-x 3 itszn itszn 4096 Oct 25 11:28 tls
itszn@ubuntu:snap-escape$ vlc
Got code execution running as itszn inside snap container!

We can read/write any non-hidden (non-dot) file in
+ echo 'Hello from snap code exec' > /home/itszn/pwned
+ cat /home/itszn/pwned
Hello from snap code exec

However we are still restricted by the container

We cannot access dotfiles
+ echo 'echo PWNED' >> /home/itszn/.bashrc
./tls/s: 20: ./tls/s: cannot create /home/itszn/.bashrc: Permission denied

Or other non-home files
+ cat /etc/issue
cat: /etc/issue: Permission denied

Luckily, this snap has the x11 plug
We can use this escape the container!
Starting container escape...

Escape Success!

We are now running code outside of snap container, we now have full privs
of itszn

For example we now can read /etc/issue:
+ cat /etc/issue
Ubuntu 18.04.4 LTS \n \l
```

```
Or modify dotfiles
+ echo 'echo PWNED' >> /home/itszn/.bashrc
+ tail -n 1 /home/itszn/.bashrc
echo PWNED

Full escape and code execution~!
---------------------------------

Attached is the poc
```

CVE References

2020-27348

itszn (itszn) wrote on 2020-10-26: #2

```
Additionally here is the script I used to generate the malicious libc.so.6
```

itszn (itszn) wrote on 2020-10-29: #3

```
Note: This was on Ubuntu 20.04 that it was tested. I have not tested on
other distros.
```

Jamie Strandboge (jdstrand) wrote on 2020-10-29: #4

```
I agree that the LD_LIBRARY_PATH should be cleaned up.

In terms of the snap application, it is always going to be confined via
the sandbox and will be as open or closed as this allows (eg, the x11 plug
necessarily gives a lot of access due to the design of X, which is why
things like xdotool allows you to do so much). Therefore, from the POV of
the *snap*, this is not a security vulnerability (it doesn't need the
shared lib from the host, it could just do that (ie, use xdotool)).

In terms of the *user*, the user doesn't need to hop through a snap
application to break out of confinement, the user is already unconfined.
In other words, the user doesn't need to put a shared lib on the system,
then run a snap to have access to /etc/issue or run xdotool. The user can
just perform those actions directly.

Where things get more interesting though is that a malicious snap with the
home interface connected could target other snaps by writing out a crafted
library then wait for another snap to be executed from this directory.
That crafted library could exfiltrate data or run other commands within
the context of the other snap. As such, I think this does constitute a
security vulnerability.

IME the bug is in snapcraft, but we should look at snapd to explore if we
should harden its 'environment' handling to remove the current directory
from various env vars.
```

itszn (itszn) wrote on 2020-10-29: #5

```
First I don't totally know where the LD_LIBRARY_PATH is being messed up so
I included snapcraft and snapd. Feel free to remove one or the other if
not relevant.

Second, the attack vector here is that the user downloads something from
the internet (let's say fantastic_4.tar.gz) then uses a snap inside the
downloaded directory. The malicious directory/libc could then gain code
execution. Normally running a program in an attacker controlled directory
should not result in attacker code execution and would be considered a bad
security vulnerability (ie exploiting vlc on its own).

I agree that the code shipped with the snap and the user doing this to
themselves is not a security concern, but most people download files and
that's where the danger is.

There are a lot of scenarios you could picture: vlc in a downloaded movie
archive, code editor in a malicious source repo, chromium run by something
like npm source watch, ghex being run on a suspicious binary to decide if
it is malicious, docker building a docker image, etc.

In each example the user is running a snap in a potential downloaded
directory which results in code execution which normally would not have
happened if they had installed the app without snap.

This is not a great look for snap if this is not fixed
```

itszn (itszn) wrote on 2020-10-29: #6

```
Additional yes I also think using this as a escalation from on snap to
anouther a potential attack vector for this bug as well.
```

Sergio Schvezov (sergiusens) on 2020-10-30

```
Changed in snapcraft:
    status:New → In Progress
importance:Undecided → Critical
  assignee:nobody → Sergio Schvezov (sergiusens)
```

Steve Beattie (sbeattie) wrote on 2020-11-03: #7

```
Please use CVE-2020-27348 for this issue. Thanks!
```

Emilia Torino (emitorino) wrote on 2020-11-10: #8

```
Hello itszn,

Thank you for the report. The security team and the snapcraft team are
coordinating a security update to fix this issue. We will inform further
as soon as we have more details.

I am removing snapd from this bug as is not affected by this
vulnerability.
```

no longer affects:snapd
Changed in snapcraft (Ubuntu):
assignee:nobody → Emilia Torino (emitorino)
importance:Undecided → High
status:New → In Progress

itszn (itszn) wrote on 2020-11-25: #9

Hi, have there been any updates on this?

Alex Murray (alexmurray) wrote on 2020-11-25: #10

Yes this is being actively worked on - emitorino is coordinating this from
the security team and will provide more details soon.

Launchpad Janitor (janitor) wrote on 2020-12-03: #11

This bug was fixed in the package snapcraft - 2.43.1+18.04.1

---

snapcraft (2.43.1+18.04.1) bionic-security; urgency=medium

[Sergio Schvezov]
* SECURITY UPDATE: library injection vulnerability on strict mode
snaps built with snapcraft via misconfigured LD_LIBRARY_PATH
- project_loader: do not export empty environment
- meta: do not export empty environment. Warn on empty environment.
- CVE-2020-27348
- LP: #1901572

-- Emilia Torino <email address hidden> Tue, 01 Dec 2020 09:10:42 -0300

Changed in snapcraft (Ubuntu):
status:In Progress → Fix Released

Sergio Schvezov (sergiusens) wrote on 2020-12-03: #12

Snapcraft 4.4.4

Changed in snapcraft:
status:In Progress → Fix Released

Emilia Torino (emitorino) wrote on 2020-12-03: #13

The security update to fix this issue has been released. Please see
https://discourse.ubuntu.com/t/usn-4661-1-snapcraft-vulnerability/19640,
https://ubuntu.com/security/notices/USN-4661-1 and https://forum.
snapcraft.io/t/ann-snapcraft-4-4-4-library-injection-vulnerability-on-
built-snaps/21465.

Thanks again itszn for your report and for helping us make Snapcraft
better! We have decided to keep this bug private for one more week given
the sensitiveness of the issue and the attached PoC you kindly provided
us.

itszn: let us know if you have any further question.

Alex Murray (alexmurray) wrote on 2020-12-04: #14

Deleted PoC etc before marking this public.

information type:Private Security → Public Security

See full activity log