NAME

guestfs−security − security of libguestfs

DESCRIPTION

This manual page discusses security implications of using libguestfs, particularly with untrusted or malicious guests or disk images.

REPORTING SECURITY PROBLEMS

If you wish to privately report a security issue, please follow the Red Hat security procedure at https://access.redhat.com/security/team/contact

If the security problem is not so serious, you can simply file a bug (see ‘‘BUGS’’ below), or send an email to our mailing list (https://www.redhat.com/mailman/listinfo/libguestfs). You do not need to subscribe to the mailing list to send email, but there will be a delay while the message is moderated.

GENERAL ISSUES

Security of mounting filesystems

You should never mount an untrusted guest filesystem directly on your host kernel (eg. using loopback or kpartx).

When you mount a filesystem, mistakes in the kernel filesystem (VFS) can be escalated into exploits by attackers creating a malicious filesystem. These exploits are very severe for two reasons. Firstly there are very many filesystem drivers in the kernel, and many of them are infrequently used and not much developer attention has been paid to the code. Linux userspace helps potential crackers by detecting the filesystem type and automatically choosing the right VFS driver, even if that filesystem type is unexpected. Secondly, a kernel-level exploit is like a local root exploit (worse in some ways), giving immediate and total access to the system right down to the hardware level.

These exploits can be present in the kernel for a very long time (https://lwn.net/Articles/538898/).

Libguestfs provides a layered approach to protecting you from exploits:

```
  untrusted filesystem
 ------------------------------------
  appliance kernel
 ------------------------------------
  qemu process running as non-root
 ------------------------------------
  sVirt [if using libvirt + SELinux]
 ------------------------------------
  host kernel
```

We run a Linux kernel inside a qemu virtual machine, usually running as a non-root user. The attacker would need to write a filesystem which first exploited the kernel, and then exploited either qemu virtualization (eg. a faulty qemu driver) or the libguestfs protocol, and finally to be as serious as the host kernel exploit it would need to escalate its privileges to root. Additionally if you use the libvirt back end and SELinux, sVirt is used to confine the qemu process. This multi-step escalation, performed by a static piece of data, is thought to be extremely hard to do, although we never say ‘never’ about security issues.

Callers can also reduce the attack surface by forcing the filesystem type when mounting (use ‘‘guestfs_mount_vfs’’ in **guestfs** (3)).

General security considerations

Be careful with any files or data that you download from a guest (by ‘‘download’’ we mean not just the ‘‘guestfs_download’’ in **guestfs** (3) command but any command that reads files, filenames, directories or anything else from a disk image). An attacker could manipulate the data to fool your program into doing the wrong thing. Consider cases such as:

- the data (file etc) not being present
- being present but empty

- being much larger than normal
- containing arbitrary 8 bit data
- being in an unexpected character encoding
- containing homoglyphs.

Protocol security

The protocol is designed to be secure, being based on RFC 4506 (XDR) with a defined upper message size. However a program that uses libguestfs must also take care − for example you can write a program that downloads a binary from a disk image and executes it locally, and no amount of protocol security will save you from the consequences.

Inspection security

Parts of the inspection API (see ‘‘INSPECTION’’ in **guestfs** (3)) return untrusted strings directly from the guest, and these could contain any 8 bit data. Callers should be careful to escape these before printing them to a structured file (for example, use HTML escaping if creating a web page).

Guest configuration may be altered in unusual ways by the administrator of the virtual machine, and may not reflect reality (particularly for untrusted or actively malicious guests). For example we parse the hostname from configuration files like */etc/sysconfig/network* that we find in the guest, but the guest administrator can easily manipulate these files to provide the wrong hostname.

The inspection API parses guest configuration using two external libraries: Augeas (Linux configuration) and hivex (Windows Registry). Both are designed to be robust in the face of malicious data, although denial of service attacks are still possible, for example with oversized configuration files.

Running untrusted guest commands

Be very cautious about running commands from the guest. By running a command in the guest, you are giving CPU time to a binary that you do not control, under the same user account as the library, albeit wrapped in qemu virtualization. More information and alternatives can be found in ‘‘RUNNING COMMANDS’’ in **guestfs** (3).

HISTORICAL SECURITY ISSUES IN LIBGUESTFS

CVE−2010−3851

https://bugzilla.redhat.com/642934

This security bug concerns the automatic disk format detection that qemu does on disk images.

A raw disk image is just the raw bytes, there is no header. Other disk images like qcow2 contain a special header. Qemu deals with this by looking for one of the known headers, and if none is found then assuming the disk image must be raw.

This allows a guest which has been given a raw disk image to write some other header. At next boot (or when the disk image is accessed by libguestfs) qemu would do autodetection and think the disk image format was, say, qcow2 based on the header written by the guest.

This in itself would not be a problem, but qcow2 offers many features, one of which is to allow a disk image to refer to another image (called the ‘‘backing disk’’). It does this by placing the path to the backing disk into the qcow2 header. This path is not validated and could point to any host file (eg. ‘‘/etc/passwd’’). The backing disk is then exposed through ‘‘holes’’ in the qcow2 disk image, which of course is completely under the control of the attacker.

In libguestfs this is rather hard to exploit except under two circumstances:

1. You have enabled the network or have opened the disk in write mode.
2. You are also running untrusted code from the guest (see ‘‘RUNNING COMMANDS’’ in **guestfs** (3)).

The way to avoid this is to specify the expected disk format when adding disks (the optional `format` option to ‘‘guestfs_add_drive_opts’’ in **guestfs** (3)). You should always do this if the disk is raw format, and it’s a good idea for other cases too. (See also ‘‘DISK IMAGE FORMATS’’ in **guestfs** (3)).

For disks added from libvirt using calls like ‘‘guestfs_add_domain’’ in **guestfs** (3), the format is fetched

from libvirt and passed through.

For libguestfs tools, use the *−−format* command line parameter as appropriate.

CVE−2011−4127

https://bugzilla.redhat.com/752375

This is a bug in the kernel which allowed guests to overwrite parts of the host's drives which they should not normally have access to.

It is sufficient to update libguestfs to any version ≥ 1.16 which contains a change that mitigates the problem.

CVE−2012−2690

https://bugzilla.redhat.com/831117

Old versions of both virt-edit and the guestfish `edit` command created a new file containing the changes but did not set the permissions, etc of the new file to match the old one. The result of this was that if you edited a security sensitive file such as */etc/shadow* then it would be left world-readable after the edit.

It is sufficient to update libguestfs to any version ≥ 1.16.

CVE−2013−2124

https://bugzilla.redhat.com/968306

This security bug was a flaw in inspection where an untrusted guest using a specially crafted file in the guest OS could cause a double-free in the C library (denial of service).

It is sufficient to update libguestfs to a version that is not vulnerable: libguestfs ≥ 1.20.8, ≥ 1.22.2 or ≥ 1.23.2.

CVE−2013−4419

https://bugzilla.redhat.com/1016960

When using the **guestfish** (1) *−−remote* or guestfish *−−listen* options, guestfish would create a socket in a known location (*/tmp/.guestfish−$UID/socket−$PID*).

The location has to be a known one in order for both ends to communicate. However no checking was done that the containing directory (*/tmp/.guestfish−$UID*) is owned by the user. Thus another user could create this directory and potentially hijack sockets owned by another user's guestfish client or server.

It is sufficient to update libguestfs to a version that is not vulnerable: libguestfs ≥ 1.20.12, ≥ 1.22.7 or ≥ 1.24.

Denial of service when inspecting disk images with corrupt btrfs volumes

It was possible to crash libguestfs (and programs that use libguestfs as a library) by presenting a disk image containing a corrupt btrfs volume.

This was caused by a NULL pointer dereference causing a denial of service, and is not thought to be exploitable any further.

See commit d70ceb4cbea165c960710576efac5a5716055486 for the fix. This fix is included in libguestfs stable branches ≥1.26.0, ≥1.24.6 and ≥1.22.8, and also in RHEL ≥7.0. Earlier versions of libguestfs are not vulnerable.

CVE−2014−0191

Libguestfs previously used unsafe libxml2 APIs for parsing libvirt XML. These APIs defaulted to allowing network connections to be made when certain XML documents were presented. Using a malformed XML document it was also possible to exhaust all CPU, memory or file descriptors on the machine.

Since the libvirt XML comes from a trusted source (the libvirt daemon) it is not thought that this could have been exploitable.

This was fixed in libguestfs ≥ 1.27.9 and the fix was backported to stable versions ≥ 1.26.2, ≥ 1.24.9, ≥ 1.22.10 and ≥ 1.20.13.

Shellshock (bash CVE−2014−6271)

This bash bug indirectly affects libguestfs. For more information see: https://www.redhat.com/archives/libguestfs/2014−September/msg00252.html

CVE−2014−8484

CVE−2014−8485

These two bugs in binutils affect the GNU **strings**(1) program, and thus the ‘‘guestfs_strings’’ in **guestfs** (3) and ‘‘guestfs_strings_e’’ in **guestfs** (3) APIs in libguestfs. Running strings on an untrusted file could cause arbitrary code execution (confined to the libguestfs appliance).

In libguestfs ≥ 1.29.5 and ≥ 1.28.3, libguestfs uses the `strings −a` option to avoid BFD parsing on the file.

CVE−2015−5745

https://bugzilla.redhat.com/show_bug.cgi?id=1251157

This is not a vulnerability in libguestfs, but because we always give a virtio-serial port to each guest (since that is how guest-host communication happens), an escalation from the appliance to the host qemu process is possible. This could affect you if:

- your libguestfs program runs untrusted programs out of the guest (using ‘‘guestfs_sh’’ in **guestfs** (3) etc), or
- another exploit was found in (for example) kernel filesystem code that allowed a malformed filesystem to take over the appliance.

If you use sVirt to confine qemu, that would thwart some attacks.

Permissions of *.ssh* **and** *.ssh/authorized_keys*

https://bugzilla.redhat.com/1260778

The tools **virt−customize** (1), **virt−sysprep** (1) and **virt−builder** (1) have an *−−ssh−inject* option for injecting an SSH key into virtual machine disk images. They may create a *~user/.ssh* directory and *~user/.ssh/authorized_keys* file in the guest to do this.

In libguestfs < 1.31.5 and libguestfs < 1.30.2, the new directory and file would get mode `0755` and mode `0644` respectively. However these permissions (especially for *~user/.ssh*) are wider than the permissions that OpenSSH uses. In current libguestfs, the directory and file are created with mode `0700` and mode `0600`.

CVE−2015−8869

https://bugzilla.redhat.com/CVE−2015−8869

This vulnerability in OCaml might affect virt tools written in the OCaml programming language. It affects only 64 bit platforms. Because this bug affects code generation it is difficult to predict which precise software could be affected, and therefore our recommendation is that you recompile libguestfs using a version of the OCaml compiler where this bug has been fixed (or ask your Linux distro to do the same).

CVE−2017−5208, CVE−2017−5331, CVE−2017−5332, CVE−2017−5333, CVE−2017−6009, CVE−2017−6010, CVE−2017−6011

Multiple vulnerabilities in the **wrestool** (1) program in the `icoutils` package can be exploited for local code execution on the host.

When libguestfs inspection (see ‘‘Inspection security’’ above) detects a Windows XP or Windows 7 guest and is asked to find an associated icon for the guest, it will download an untrusted file from the guest and run `wrestool −x` on that file. This can lead to local code execution on the host. Any disk image or guest can be crafted to look like a Windows guest to libguestfs inspection, so just because you do not have Windows guests does not help.

Any program calling the libguestfs API `guestfs_inspect_get_icon` could be vulnerable. This includes **virt−inspector** (1) and **virt−manager** (1).

The solution is to update to the non-vulnerable version of icoutils (at least 0.31.1).

CVE−2017−7244, CVE−2017−7245, CVE−2017−7246

Multiple vulnerabilities in PCRE could be exploited to crash libguestfs (ie. cause a denial of service) when performing inspection on an untrusted virtual machine.

The solution is to update to a version of PCRE with these bugs fixed (upstream version ≥ 8.41).

CVE−2018−11806

Vulnerabilities affecting qemu user networking (SLIRP) allow a malicious filesystem image to take control of qemu and from there attack the host.

This affects libguestfs when the backend is set to `direct` *and* networking is enabled.

The direct backend is the default upstream, but not in some downstream Linux distributions including Fedora, Red Hat Enterprise Linux and CentOS. It might also have been selected if you set the `LIBGUESTFS_BACKEND=direct` environment variable or called `guestfs_set_backend(g,"direct")`.

Networking is enabled automatically by some tools (eg. **virt−builder** (1)), or is enabled if your code called `guestfs_set_network(g,1)` (which is not the default).

The libvirt backend is not affected.

The solution is to update qemu to a version containing the fix (see https://lists.gnu.org/archive/html/qemu−devel/2018−06/msg01012.html).

SEE ALSO

guestfs (3), **guestfs−internals** (1), **guestfs−release−notes** (1), **guestfs−testing** (1), http://libguestfs.org/.

AUTHORS

Richard W.M. Jones (`rjones at redhat dot com`)

COPYRIGHT

LICENSE

BUGS

To get a list of bugs against libguestfs, use this link: https://bugzilla.redhat.com/buglist.cgi?component=libguestfs&product=Virtualization+Tools

To report a new bug against libguestfs, use this link: https://bugzilla.redhat.com/enter_bug.cgi?component=libguestfs&product=Virtualization+Tools

When reporting a bug, please supply:

- The version of libguestfs.
- Where you got libguestfs (eg. which Linux distro, compiled from source, etc)
- Describe the bug accurately and give a way to reproduce it.
- Run **libguestfs−test−tool** (1) and paste the **complete, unedited** output into the bug report.