Date: Wed, 4 Nov 2020 10:17:05 +0100
From: Matthias Gerstner <mgerstner@...e.de>
To: oss-security@...ts.openwall.com
Subject: Security Issues in the spice-vdagentd daemon

Hello list,

please find below a security report regarding the spice-vdagentd [1].

Attached to this mail is a tarball containing the final set of patches
that upstream developed to address the issues. These patches are by now
already published in the upstream git repository. Furthermore attached
is a tarball containing scripts and source code to reproduce some of the
issues discussed in the report below. CVEs have been assigned by
upstream for the individual issues, please find them in the respective
sections of the report.

[1]: https://gitlab.freedesktop.org/spice/linux/vd_agent

1) Introduction

The SUSE security team got a request to review the `spice-vdagentd` daemon.
It has been brought to our attention that this daemon can start
indirectly via udev (`70-spice-vdagentd.rules`) even if the service was
not explicitly activated by the Administrator. Thus it might introduce
additional security attack surface in default SUSE installations.

The `spice-vdagentd` is typically used in Qemu virtual machines to provide
additional features like:

- a shared clipboard between host and guest system to allow seamless
  copy/paste
- file transfers from the host to the guest machine
- sharing information about the virtual display size and monitor
  configuration
- synchronizing audio device volume between host and guest

`spice-vdagentd` is running as *root* as a socket activated systemd
service.

The following security report is based on the spice-vdagent version
0.20.0, which was the latest upstream release available at the time of
its writing.

2) The `vdagentd` Communication Channels

This section gives a short introduction about the communication channels
used by `vdagentd` for readers that are not familiar with its design.

a) The com.redhat.spice.0 Virtio Serial Device

In Qemu virtual machine instances, that have the spice protocol enabled,
a virtual serial device will be emulated, typically enabled by passing
switches like these to qemu:

```
-spice disable-ticketing,unix,addr=/path/to/spice_socket, -device virtio-serial-pci -device virtserialport,chardev=spicechannel0,name=com.redhat.spice.0 -chardev
spicevmc,id=spicechannel0,name=vdagent
```

Within the virtual machine `spice-vdagentd` talks to this serial device
(`/dev/virtio-ports/com.redhat.spice.0`) to exchange information with
the host machine and implement the spice features. The SPICE protocol is
used on this communication channel.

This serial device is of lesser importance in the context of this
security report.

b) The spice-vdagent UNIX Domain Socket

The `spice-vdagentd` creates a listening UNIX domain socket of type
`SOCK_STREAM` in `/run/spice-vdagentd/spice-vdagent-sock`. This socket
path is accessible to all users in the system (file mode 0666). It is
used by the counterpart of `spice-vdagentd`, the `spice-vdagent`, which
runs in the context of a graphical user session with user privileges.
This role will also be called the "user agent" for the rest of this
document.

The `spice-vdagent` e.g. informs the `spice-vdagentd` about changes in
clipboard contents in the graphical VM session, which in turn forwards
this information via the serial device to the host machine. Similarly
requests from the host machine to e.g. initiate a file transfer will be
forwarded from `spice-vdagentd` to the `spice-vdagent`, which will then
store the file data in the context of the user session.

Multiple user agents can connect to `spice-vdagentd` at the same time.
At most one user agent may be present in the currently active session,
however. Only a user agent running in the currently active session can
access most of the features of `spice-vdagentd`. `spice-vdagentd`
determines the session a user agent is part of by querying systemd
facilities (e.g. `sd_seat_get_active()`, `sd_pid_get_session()`). The
user agent (if any) associated with the currently active session will
also be the target of any forwarded requests received from the host on
the virtual serial device.

3) Security Issues

This report is accompanied by two Python scripts (`vdagent.py`,
`create_vdagent_conns.py`) and a C++ source file
(`socket_pid_attack.cxx`) which can be used to reproduce the findings
that follow. These reproducers will be mentioned in the individual
sections. All reproducers need to be executed in the context of a host
system running a SPICE enabled Qemu virtual machine (see section 2.a for
the required Qemu switches, or use something like livbirtd and its Qemu
backend with appropriate settings).

a) CVE-2020-25650: Memory DoS via Arbitrary Entries in `active_xfers` Hash Table

The `spice-vdagentd` maintains a hash map named `active_xfers` that maps
`task_ids` to UNIX domain socket connections they belong to. These
`task_ids` refer to ongoing file transfers from the host to the virtual
machine.

An arbitrary client connected to `spice-vdagentd` via a UNIX domain
socket can trigger an entry into this hash map, without the requirement
that the client is associated with the currently active graphical
session (function `do_agent_file_xfer_status`, specifically
`vdagentd.c:1025`). There is no limit on the maximum amount of file
transfers ongoing in parallel and there are no timeouts applied for a
file transfer to be finished.

Therefore any unprivileged local user with access to the
`/run/spice-vdagentd/spice-vdagent-sock` socket path can perform a
memory denial-of-service by entering a large amount of entries into this
hash map.

Impact

- an unprivileged local attacker inside a virtual machine can cause

  large memory allocation in a daemon running as *root*. These memory
  allocations will persist as long as the UNIX domain socket connection
  remains intact. After a longer while, depending on the available
  system memory and swap space, the system might enter an out of memory
  situation, causing a denial-of-service for `spice-vdagentd` or even
  other processes in the system (the Linux kernel OOM killer is known to
  punish innocent victims at times).
- after a large amount of entries have been made by an attacker this way
  and once the UNIX domain socket connection is terminated, the
  `spice-vdagentd` will remove all the hash map entries belonging to
  this connection (function `agent_disconnect`, specifically
  `vdagentd.c:955`). Each "cancelled" file transfer will be logged in
  the system (`vdagentd.c:910`, `vdagentd.c:342`). In my tests this
  also caused a high load in `systemd-journald` and high disk space
  consumption from the logs stored in `/var/log`.

Fixes

- Patch 02 changes the handling of entries into the hash map. User
  agents will no longer be able to trigger arbitrary entries into it.
- Patch 03 limits the maximum number of transfers allowed to be active
  at the same time.

Reproducer

Using the supplied Python program the DoS can be performed this way:

```
user $ ./vdagent.py --xfer-status-DoS
```

The program will infinitely create new entries in the `active_xfers`
hash map up until the maximum 64-bit (for `x86_64` architectures)
integer value. Memory consumption will grow rather slowly but reliably.
An exponential growth allocation scheme seems to be in place, because
the observed memory consumption (as seen in `top`) only happens in
jumps.

b) CVE-2020-25651: Possible File Transfer DoS and Information Leak via `active_xfers` Hash Map

The same basic problem as described in section 3.a can lead to a file
transfer information leak. The file transfer protocol roughly works like
this:

- The host will send a `VD_AGENT_FILE_XFER_START` message that is
  forwarded to the user agent (function `do_client_file_xfer()`,
  specifically `vdagentd.c:376`). This message contains a `task_id` that
  identifies the file transfer process in future messages.
- The `spice-vdagent` will check free disk space and allocate a file of
  the expected size in the file system. If all checks pass then it will
  reply with a `VDAGENTD_FILE_XFER_STATUS_CAN_SEND_DATA` message, which
  causes `spice-vdagentd` to associate the client connection with the
  ongoing file transfer.
- The host will now start sending out chunks of the file data with
  `VDAGENTD_FILE_XFER_DATA` messages (processed in function
  `do_client_file_xfer()`, specifically `vdagentd.c:386`).
  `spice-vdagentd` will forward each chunk to the client connection
  stored in the `active_xfers` hash map.

The host application (tested with `remote-viewer` from the virt-viewer
package [2]) chooses an incrementally growing `task_id` for file
exchanges which starts counting at 1. Thus the `task_id` is predictable.
Since any unauthenticated local client can replace the mapping of
`task_id` to client connection by its own client connection, there is a
possibility for an attacker to obtain parts of the transferred file
data.

The attacker needs to win a race condition here, because it needs to hit
the time window after the legitimate client sends out the
`VDAGENTD_FILE_XFER_STATUS_CAN_SEND_DATA` message and before the host
starts sending out file chunks via `VDAGENTD_FILE_XFER_DATA`. If the
attacker sends his own `VDAGENTD_FILE_XFER_STATUS_CAN_SEND_DATA` using
the correct `task_id` during this time window, then he can obtain the
complete file. At least for large file exchanges bigger parts of the
file are feasible to be obtained, even when the initial parts of the
file are transferred to the legitimate client.

The more difficult part for an attacker will be to identify when such a
file transfer will take place. The reproducer shows the basic attack
technique.

[2]: https://gitlab.com/virt-viewer/virt-viewer

Impact

File data from the host system can end up in full or in parts in the
client connection of an illegitimate local user in the VM system.
Exploitability will be difficult if there is not a suitable side channel
with information about file transfers going on.

In any case active file transfers from other users can also be
interrupted (DoS aspect).

Fixes

Additionally to the fixes in section 3.a:

- Patch 08 cleans all entries from the hash map when the active user
  agent disconnects.
- Patch 09 prevents existing entries in the hash map to be replaced.

Reproducer

This reproducer has the precondition that the victim's home directory is
world-readable (or at least readable by the attacker). The script will
wait (using the *inotify* API) for a new file to be created in the home
directory (assuming that this is the initiation of a file transfer).

For reproducing perform the following steps:

1. Open a fresh `remote-viewer` connection for the test VM. This will
   reset the `task_id` used for file transfers to start to 1.
2. Run the reproducer script in an attacker context as follows:

    ```
    # replace this path by the actual victim's home directory
    attacker$ VICTIM=/home/victim
    attacker$ ./vdagent.py --send-xfer-status 1 --wait-for-file-create $VICTIM
    ```

3. Log in as the *victim* user in a graphical session in the VM using
   the `remote-viewer` connection started in step 1.
4. Drag-and-drop a larger file (I tested using an 8 megabyte text file)
   into the `remote-viewer` window to initiate file transfer.

You should see that the attacker's `vdagent.py` script outputs large
parts of the file transfer to the terminal. The original file transfer
will indicate an error condition without specific details about the
problem.

Remember that for each subseqeuent attack attempt you will need to
increment the `task_id` (`--send-xfer-status` parameter) or open a fresh
`remote-viewer` window for the VM, to reset the `task_id` used on the
host side.

c) CVE-2020-25652: Possibility to Exhaust File Descriptors in `vdagentd`

`spice-vdagentd` does not apply a limit to the amount of client
connections that can be established via the UNIX domain socket in
`/run/spice-vdagentd/spice-vdagent-sock`. Also existing connections
aren't subject to a timeout or any kind of preconditions for them to
stay alive. Thus it is easy to exhaust the file descriptor limit for

the `spice-vdagentd` process (typically 1024 file descriptors by
default, this limit is also imposed by system calls like `select()`).

Any local user in the virtual machine can open around ~1020 connections
to `spice-vdagentd` and simply keep them open without transmitting any
data. The `spice-vdagentd` will then become unable to open further
connections for legitimate clients or perform other tasks (like opening
the serial device, see section 2.a, or invoking systemd library calls
that require opening files).

Impact

By exhausting file descriptors in `spice-vdagentd` the following effects
can be achieved:

- The attack can prevent legitimate `spice-vdagent` instances from
  connecting to the `spice-vdagentd`. SPICE features won't be available
  to affected sessions.
- The attack can cause `vdagentd` to exit on error conditions if tuned
  carefully. For example, an attacker can exhaust all file descriptors
  in `spice-vdagentd` except for one and then wait for a legitimate
  client from an active session to connect. This connection attempt will
  succeed, but the subsequent attempt to open the serial device (see
  section 2.a) will fail, and `spice-vdagentd` will exit. This will then
  also cause the involved `spice-vdagent` to exit, because the
  connection to the system daemon is lost.
- `spice-vdagentd` will enter a 100 % CPU load infinite loop, because it
  tries to `accept()` the new connection, which is impossible, but also
  doesn't close the listening socket or abort execution.
- This attack vector makes security issue 3.d better exploitable, which
  will be explained there in more detail.

Fixes

- Patch 04 implements an upper limit of client connections accepted by
  the `spice-vdagentd`.
- Patch 07 implements a limit for client connections established from
  the same session.

Reproducer

Using the supplied script `create_vdagent_conns.py` the maximum number
of connections that `spice-vdagentd` can process, can be established.
The result will be that no new user agents can register with the system
daemon. A 100 % CPU loop will occur in `spice-vdagentd`.

d) CVE-2020-25653: UNIX Doman Socket Peer PID Retrieved via `SO_PEERCRED` is Subject to Race Condition

One major security property of `spice-vdagentd` is that it only allows
those clients access to most of the SPICE features (like clipboard, file
transfer) that are currently in an active session according to systemd
(see also section 2.b). It is possible for arbitrary local users (like
nobody) to connect to `spice-vdagentd` but these connections should
not be able to interact with the host machine, because they don't belong
to the active session.

The session check is performed after a new UNIX domain socket connection
is established in `agent_connect()` in `vdagentd.c:937`. The check
basically relies on these two source code lines:

```
pid = vdagent_connection_get_peer_pid(VDAGENT_CONNECTION(conn), &err);
agent_data->session = session_info_session_for_pid(session_info, pid);
```

The peer's PID is obtained via glib's `g_socket_get_credentials` which
boils down to the `SO_PEERCRED` socket option that is supported for UNIX
domain sockets (see `man 7 socket`, `man 7 unix`, `struct ucred`). The
man page says about this:

> The returned credentials are those that were in effect at the time of the call
> to connect(2) or socketpair(2).

This means that there is a race condition between the point in time when
a client performs the `connect()` call to establish a connection with
`spice-vdagentd` and the time `spice-vdagentd` retrieves and checks the
PID in its `agent_connect()` function. The PID in question can already
have been replaced by an unrelated process. Therefore the session that
`spice-vdagentd` associates with this PID might be a different one than
the actual peer process belonged to, when the `connect()` system call
was performed.

An attack to exploit the race condition requires the following steps:

1. an attacker can inherit a UNIX domain socket file descriptor to a
   child process that performs the `connect()` to `spice-vdagentd` and
   exits immediately again, thereby freeing the PID (let's call it the
   malicious PID) in the system as soon as the parent process performs a
   `wait()` on the exited child process. This malicious PID will now be
   associated in the kernel with the `SO_PEERCRED` data returned for the
   connected UNIX domain socket.
2. now the attacker needs to perform a PID cycle in the system (i.e.
   create many useless child processes to cause the maximum PID
   - typically 32768 - to be reached in the system and new processes get
   assigned small PIDs again). When the PIDs assigned by the kernel are
   getting close to the malicious PID, the attacker needs to stop
   creating child processes and wait for unrelated processes from other
   users to come into existence.
3. Once the malicious PID gets reassigned to an unrelated process and
   the `agent_connect()` function runs in `spice-vdagentd`, it will
   retrieve wrong session information for the existing connection. If
   the malicious PID gets reassigned to a process running in the active
   session, then the connection that the attacker uses will get access
   to the SPICE features and can communicate with the host, although the
   attacker would otherwise not have sufficient privileges to do so.

The described race condition is very hard to hit under normal
circumstances, because step 2., the PID cycle, is taking a long time and
the `agent_connect()` function in `spice-vdagentd` is very likely to run
before an unrelated process gets reassigned the malicious PID in
question. When combined with the file descriptor exhaustion security
issue described in section 3.c, however, then this attack will become
way more feasible.

This combined attack works like follows:

- Exhaust all file descriptors in `spice-vdagentd` as described in
  section 3.c.
- Perform the attack steps 1. and 2. as described previously. What
  happens now is that the attacker's UNIX domain socket `connect()` will
  succeed, because on kernel level this is still possible.
  `spice-vdagentd` won't be able to `accept()` this connection, though,
  because no more file descriptors are available to do so. The
  connection remains pending on the listening socket, however.
- Now for step 3., once the attacker notices that the malicious PID got
  assigned to an unrelated process, he can stop the file descriptor
  exhaustion put into place previously, thus making it possible for
  `spice-vdagentd` to `accept()` the malicious connection pending in the
  kernel. Only now will the `agent_connect()` function run, and it will
  more reliably determine the wrong session for the connection.

Impact

1. A compromised local account with little privileges inside the virtual
   machine like *nobody* can try to become the "active agent" for
   `spice-vdagentd` for the graphical session of a legitimate local
   user. If successful then the attacker can access the host's
   clipboard contents or send malicious clipboard content to the host.
   The attacker can also retrieve file data from the host (compare
   section 2.b) or send invalid screen resolution and display

   information to the host.
2. The combined attack using the file descriptor exhaustion and the
   `SO_PEERCRED` race condition is still not 100 % reliable but it can be
   repeated many times to increase chances of success. The only
   unpredictable ingredient is victim child processes appearing that get
   assigned the desired malicious PID and stay around for long enough for
   `spice-vdagentd` to pick up the wrong session information.
3. If the victim's graphical session already runs a legitimate
   `spice-vdagent` then a successful attack will trigger an information
   leak protection logic in `vdagentd.c:874`. This has the effect of a
   denial-of-service, because neither the attacker nor the legitimate user
   will be able to use the SPICE features anymore.
4. If the victim's graphical session is not running a `spice-vdagent`
   then the attacker can achieve all the effects described in 1.
5. If 3. applies (the victim's is already running `spice-vdagent`) then
   the attacker could try to crash the currently running `spice-vdagentd`
   (see section 3.c for a possible attack vector). systemd should then
   restart the `spice-vdagentd` while the victim's `spice-vdagent` should
   exit but not be restarted. After this situation 4) applies.

Fixes

- Patches 05 and 06 change the session check logic mainly by also taking
  into account the connected client's UID. If the UID of the
  determined session and the client don't match then the connection will
  be terminated.

Reproducer

This attack is quite complex and requires the combination of all
supplied programs to succeed. All programs must be placed in the same
directory, because they interact with each other. Perform the following
steps:

1. For reduced complexity it is a precondition that an active graphical
   session already exists in the VM, but no `spice-vdagent` is running in
   it. Therefore log in the "victim" user in a graphical session, kill any
   `spice-vdagent` that got started automatically and keep the session
   open.
2. In the context of an "attacker" user run `create_vdagent_conns.py`:

       attacker$ ./create_vdagent_conns.py
        Established 1015 active connections to vdagentd.
        Waiting for Ctrl-C. Or press ENTER to close 5 sockets.

   Wait until the program displays the message seen above. Keep the
   program running in this state.
3. Still in the context of the attacker compile and run the C++ program:

       attacker$ g++ -O2 socket_pid_attack.cxx -o socket_pid_attack
       attacker$ ./socket_pid_attack
        Target UDS PID = ????
        Cycled to PID ....
        [...]
        Closing in to target_pid ????: Got child PID ????
        Now waiting for ???? to get reassigned.

  This program creates a UNIX domain socket connected to `spice-vdagentd` but
  the `connect()` call is performed in a short-living child process,
  thus freeing the PID the kernel stored for `SO_PEERCRED`. The program
  then waits for the malicious PID to be assigned to an unrelated
  process before it continues. Keep the program running in this state.
4. In the context of the "victim" user create a number of long running
   new child processes in the graphical session. For example something
   like this:

       victim$ for i in `seq 20`; do sleep 1h & done

   This simulates new child processes being created in the victim's context to
   replace the malicious PID the attacker is waiting for.
5. If successful then you should see in the program from step 3. that
   something happened:

        PID ??? now exists, but can't read exe: Permission denied
        Running './vdagent.py --socket-fd 3 '
        Using existing connected socket file descriptor

   If this did not work right away then you need to repeat steps 3) and
   4) (rather quickly) until it succeeds.

   In the case of success, the `socket_pid_attack` program will have
   replaced itself by the `vdagent.py` Python program, which needs to be
   present in the same directory. The Python program will block, trying
   to receive data on the connected UNIX domain socket, because
   `spice-vdagentd` can't accept the connection.
   Now terminate the program still running from step 1. via Ctrl-C to
   free up the blocked file descriptors in `spice-vdagentd`. The
   malicious connection should now be accepted by `spice-vdagentd` and
   the attacker should have become the active agent for the victim's
   graphical session. The additional output of the program from step 3.
   should look similar to this:

   ```
   MessageType.VERSION arg1 = 0 arg2 = 0 bytes = 7
   b'302e32302e3000'
   sending resolution of 1024 768
   MessageType.GRAPHICS_DEVICE_INFO arg1 = 0 arg2 = 0 bytes = 4
   b'00000000'
   MessageType.CLIENT_DISCONNECTED arg1 = 0 arg2 = 0 bytes = 0
   MessageType.GRAPHICS_DEVICE_INFO arg1 = 0 arg2 = 0 bytes = 4
   b'00000000'
   MessageType.GRAPHICS_DEVICE_INFO arg1 = 0 arg2 = 0 bytes = 4
   b'00000000'
   MessageType.AUDIO_VOLUME_SYNC arg1 = 0 arg2 = 0 bytes = 7
   b'01eb0200000000'
   MessageType.AUDIO_VOLUME_SYNC arg1 = 0 arg2 = 0 bytes = 7
   b'00eb0200000000'
   MessageType.CLIPBOARD_RELEASE arg1 = 1 arg2 = 0 bytes = 0
   sending clipboard grab request for ClipboardType.SELECTION_PRIMARY
   ```

   If the attacker's connection is not considered to be part of the
   active session then the `AUDIO_VOLUME_SYNC` and
   `GRAPHICS_DEVICE_INFO` messages will not be received from the
   `spice-vdagentd`. This can happen if some other user in the system
   received the malicious PID or if the PID was not in existence for
   long enough. If this is the case repeat steps 3. to 5. until it
   succeeds.
6. If the previous steps all succeeded then you should be able to see
   for example the clipboard content from the host when a SPICE capable
   viewer is used to connect to the virtual machine. The attacker can
   also send back "malicious clipboard content" to the host.

4) Other Suggestions

a) fchmod() instead of chmod() for UNIX domain socket

I made a minor suggestion about the `chmod()` call in `vdagentd.c:1223`.
If possible this should be replaced by a call to `fchmod()`, if the file
descriptor can be obtained from the glib functions. If custom, unsafe
paths are used for the UNIX domain socket then symlink attacks might
become possible through the use of `chmod()`.

This was addressed by upstream in Patch 01.

5) Timeline

2020-09-18: I privately sent a version of this report and the
            reproducers to the upstream developer(s).
2020-10-16: Upstream shared their first version of the patches with me,

```
           I reviewed them and we discussed some details.
2020-10-26: I was asked to send this prenotification to the
           linux-distros mailing list to prepare the publication of the
           issues.

# 6) Further References

- SUSE bugzilla tracker bug [3]

[3]: https://bugzilla.suse.com/show_bug.cgi?id=1173749

Cheers

--
Matthias Gerstner <matthias.gerstner@...e.de>
Dipl.-Wirtsch.-Inf. (FH), Security Engineer
https://www.suse.com/security
Phone: +49 911 740 53 290
GPG Key ID: 0x14C405C971923553

SUSE Software Solutions Germany GmbH
HRB 36809, AG Nürnberg
Geschäftsführer: Felix Imendörffer
```

Download attachment "vdagent_reproducers.tar.gz**" of type "**application/octet-stream**" (7126 bytes)**

Download attachment "vdagent_security_patches_v5.tgz**" of type "**application/x-gtar**" (8360 bytes)**

Download attachment "signature.asc**" of type "**application/pgp-signature**" (834 bytes)**

Powered by blists - more mailing lists

Please check out the Open Source Software Security Wiki, which is counterpart to this mailing list.

Confused about mailing lists and their use? Read about mailing lists on Wikipedia and check out these guidelines on proper formatting of your messages.