

[Products](#)[Services](#)[Publications](#)[Resources](#)[What's new](#)

Follow [@Openwall](#) on Twitter for new release announcements and other news

[<prev](#)] [\[next>](#)] [\[day\]](#) [\[month\]](#) [\[year\]](#) [\[list\]](#)

Date: Thu, 14 Apr 2022 12:29:36 +0200
From: Matthias Gerstner <mgerstner@...e.de>
To: oss-security@...ts.openwall.com
Subject: Multiple vulnerabilities in swhkd hotkey helper for Wayland

Hello list,

swhkd [1] is a pair of programs that allow to configure arbitrary keyboard hotkey definitions for the Wayland graphics system, written in the Rust programming language. The RPM package integration submitted for openSUSE Tumbleweed contained an unusual Polkit rules definition file that required a review by the SUSE security team [2].

As a result of the review multiple security issues have been identified. The individual issues are described in the following detailed report.

1) Introduction =====

1.a) swhkd Design and Operation -----

swhkd consists of two components: `swhks` ("server") which runs in the context of the unprivileged user to execute hotkey actions and `swhkd` ("daemon") which runs with root privileges to interact with keyboard devices on uinput API level. The two components communicate with each other via a UNIX domain socket.

It is a bit unclear if `swhkd` allows other modes of operation. The function `permission_check()` is somewhat confusing in this regard. In the end the only way to get it running is doing it as root, and the README also points to `pkexec` for running it. So this is the only setup that I analyzed for this report.

1.b) Scope of Review -----

The target of this review was version 1.1.5 of swhkd.

1.c) Polkit pkexec Usage =====

In version 1.1.5 of swhkd a Polkit rules file
"/usr/share/polkit-1/rules.d/swhkd.rules" is used containing the following rule snippet:

```
...
polkit.addRule(function(action, subject) {
    if (action.lookup("program") == "/usr/bin/swhkd") {
        return polkit.Result.YES;
    }
})
...
```

This means that any user in the system independently of a session context is able to execute /usr/bin/swhkd as root via pkexec passing arbitrary parameters e.g. also a compromised 'nobody' user account could do it.

According to my recommendation upstream switched to using a Polkit policy file instead that only allows local users in an active session to run swhkd as root.

2) Security Issues =====

2.a) Use of Fixed Temporary File in /tmp/swkbd.pid (CVE-2022-27815)

The daemon running as root uses this path to record its own PID for instance monitoring. This fixed path has the following issues:

- local application DoS: if an attacker places the PID of an existing process there (e.g. PID 1), other users cannot start the daemon any more.
- local information leak if the kernel's symlink protection is turned off. The daemon logs the **full** content of the PID file to stdout. Example output of a run with a symlink pointing to a private root owned file containing the string "secret":

```
[2022-03-21T15:44:28Z DEBUG swkbd] Previous PID: secret
```

- local system DoS if kernel symlink protection is off. The daemon will overwrite the target file with its own PID. This could also create a world-readable file in interesting locations that allow for further attack vectors in other programs.
- functional issue: such a setup is not suitable for multiple users using the software in parallel.

Bugfix

The issue has been fixed upstream via the following commit:

<https://github.com/waycrate/swkbd/pull/93/commits/ee3534b401cba71fddf378f7e1cb18ada0e6fa75>

The PID file is now placed into /etc, which fixes the issue, albeit is an incorrect location. I suggested upstream to place this file in /run instead.

2.b) The `-c` Daemon Command Line Parameter Allows for Arbitrary File Existence Tests (CVE-2022-27814)

Example exploitation:

```
$ pkexec /usr/bin/swkbd -d -c /root/.somefile
[2022-03-22T12:32:25Z ERROR swkbd] "/root/.somefile" doesn't exist
```

```
$ pkexec /usr/bin/swkbd -d -c /root/.bash_history
[...] (daemon starts "normal" operation)
```

Bugfix

Upstream tries to fix this via the following commit:

<https://github.com/waycrate/swkbd/pull/105/commits/3c111663f2dcf8f28214ff1907169aa669fcf791>

This commit now uses the external `cat` program to read in the configuration file. Actually this is just a workaround, because the fix for issue 2.c) is incomplete (only the root UID is dropped, not the root GID, see [3]).

2.c) The `-c` Daemon Command Line Parameter Allows to Parse Arbitrary Files (CVE-2022-27819)

The file passed to `swkbd` via `-c` will be completely read in by `swkbd`. Any privileged file can thus also be processed. The daemon only outputs the contents if something that looks like a hotkey definition is found in the file, however. Since this syntax is pretty complex the involved information leak is rather hard to exploit. Something like

```
$ pkexec /usr/bin/swkbd -d -c /dev/sda
```

causes the daemon to "parse" the complete block device, exhausting memory and causing high I/O load.

Bugfix

Upstream addressed this by dropping privileges to the invoking user via

the following commit:

<https://github.com/waycrate/swkbd/commit/8ddd40fdc0fla356816671d6a91f067c112e4724>

The privilege drop is still incomplete, because the root group ID privilege is not dropped.

The security stance of swkbd can still be improved by dropping privileges to the invoking user by default and only raising them for privileged operations. I recommended this to upstream for follow up changes of the codebase.

2.d) The Daemon Connects to a Fixed UNIX Domain Socket in /tmp/swkbd.sock (CVE-2022-27818)

For connecting to the unprivileged swbks sibling process the swkbd daemon connects to the fixed domain socket path /tmp/swkbd.sock. This causes the following issues:

- local DoS: If an attacker pre-creates this pathname then the daemon cannot send out hotkey events and the unprivileged server component cannot start up successfully.
- local information leak: If an attacker places its *own* UNIX domain socket there, then the attacker will receive hotkey events instead of the legitimate user. The information contains the commands to be executed which is depending on the actual hotkey setup. In some cases this might even contain sensitive data.

Bugfix

Upstream addressed this by placing the socket into the unprivileged user's private /run/user/\$UID directory. This is done via the following commit:

<https://github.com/waycrate/swkbd/commit/3187d7fc75fe5833f903b342941eebcla56e5979>

2.e) Input Events are Consumed For all Keyboard Input Devices in all Sessions (CVE-2022-27817)

The daemon listens for input events on uinput device level. This means even other users in other Wayland sessions or on the text mode consoles will be affected by this. In theory this fact could be used to log passwords and other sensitive information from other users. However, recognized hotkey events will be discarded by the daemon i.e. if regular key presses are configured as hotkeys then the keys seem to work no longer. Therefore it is more like a local DoS for other users.

Bugfix

This issue has not been addressed by upstream yet. My suggested fix is as follows:

Establish a systemd Session Context: I think it is possible to determine the current session the unprivileged user is in via systemd. Then the daemon should pause its operation as soon as the active session is changed to another one, and reactivate operation once the original user session becomes active again.

2.f) The Unprivileged Server Process Uses a Fixed Temporary File in /tmp/swbks.pid (CVE-2022-27816)

This issue is similar to 2.a). The consequences are:

- local application DoS: if an attacker places the PID of an existing process there (e.g. PID 1), other users cannot start the server any more.
- local user file corruption: if kernel symlink protection is off, then the PID file can be a symlink to a private file in the user's home directory which will then be overwritten with the PID information.

Bugfix

Upstream addressed this issue by placing the PID file in the respective user's private /run/user/\$UID directory. This is done in the following commit:

<https://github.com/waycrate/swhkd/commit/4b8442fef512441c9155186956c767a120c12974>

2.g) The Unprivileged Server Process Receives Commands via /tmp/swhkd.sock

The socket used by the daemon (see 2.d) is created by the unprivileged server. The daemon sends commands to be executed to this socket and the server will execute them.

The UNIX socket `bind()` call only succeeds if the target file does not exist yet and it also doesn't follow symlinks. Therefore an attacker cannot pre-create this socket, without the server process failing to `bind()`. For `connect()`ing to the socket the caller needs to have write permissions on the socket. Therefore arbitrary other users cannot send commands to the server process if a `_sane_umask` is configured.

Should the user starting the server process **not** have a sane umask, however, then the socket could become writable for other users and therefore other users could execute arbitrary code in the context of the unprivileged user:

```
$ umask 0
$ swhks &
$ ls -lh /tmp/swhkd.sock
srwxrwxrwx 1 mgerstner users 0 Mär 22 13:46 /tmp/swhkd.sock
```

Since an overly open umask is an issue in itself I did not request a CVE for this issue.

Bugfix

To protect against this scenario the socket file is now placed in the respective user's private `/run/user/$UID` directory. Furthermore for hardening purposes a sane umask is applied. The following upstream commit contains this change:

<https://github.com/waycrate/swhkd/commit/707ea9915c9cf3d68b4a6729ac212a46a486e3fd>

3) Timeline

=====

- 2022-03-22: I reported these findings to upstream privately. I offered formal coordinated disclosure, no defined embargo and publication date have been established though.
- 2022-03-23: Upstream started quickly to work on bugfixes which have been published on GitHub right away. Therefore increasing parts of the information has become public over time.
- 2022-03-24: I received CVEs for the issues from Mitre and communicated them to upstream.
- 2022-04-13: I asked upstream for permission to publish the full report, since most CVEs have already public fixes, some of them still incomplete though.

References

=====

- [1]: <https://github.com/waycrate/swhkd>
- [2]: https://bugzilla.suse.com/show_bug.cgi?id=1196890
- [3]: <https://github.com/waycrate/swhkd/pull/102#issuecomment-1088417911>

Best Regards

Matthias

--

Matthias Gerstner <matthias.gerstner@...e.de>
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: Ivo Totev

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](#).

