Closed Issue created 2 years ago by Kevin Backhouse

CVE-2020-12049: File descriptor leak in _dbus_read_socket_with_unix_fds

GitHub Security Lab (GHSL) Vulnerability Report: `GHSL-2020-057`

The GitHub Security Lab team has identified a potential security vulnerability in dbus.

We are committed to working with you to help resolve these issues. In this report you will find everything you need to effectively coordinate a resolution of these issues with the GHSL team.

If at any point you have concerns or questions about this process, please do not hesitate to reach out to us at `securitylab@github.com` (please include your `GHSL-2020-057` as a reference).

If you are *NOT* the correct point of contact for this report, please let us know!

Summary

D-Bus has a file descriptor leak, which can lead to denial of service when the dbus-daemon runs out of file descriptors. An unprivileged local attacker can use this to attack the system dbus-daemon, leading to denial of service for all users of the machine.

Product

D-Bus (dbus-daemon)

Tested Version

1.12.2-1ubuntu1.1 (tested on Ubuntu 18.04.4 LTS)

Details

Issue 1: File descriptor leak in `_dbus_read_socket_with_unix_fds`

The function `_dbus_read_socket_with_unix_fds` contains the following code at dbus-sysdeps-unix.c, line 438:

```
if (m.msg_flags & MSG_CTRUNC)
  {
    /* Hmm, apparently the control data was truncated. The bad
       thing is that we might have completely lost a couple of fds
       without chance to recover them. Hence let's treat this as a
       serious error. */

    errno = ENOSPC;
    _dbus_string_set_length (buffer, start);
    return -1;
  }
```

The intention of this code is to handle the case where too many file descriptors are sent over the unix socket, causing the control data to get truncated. That could be a deliberate attempt by an attacker to cause a denial of service. The problem with the code is that some file descriptors may still have been received, even though the message has been truncated. So we need to make sure that those file descriptors are closed. Otherwise an attacker can cause us to quickly run out of file descriptors.

In my opinion, the simplest solution is to just delete this block of code entirely. The loop below (starting at line 450) handles the file descriptors correctly, so it is better to ignore the MSG_CTRUNC flag and process the message as normal. File descriptors will be correctly closed if the message is invalid. I have confirmed that my proof-of-concept exploit does not work if this block of code is deleted. An alternative solution is to postpone checking the MSG_CTRUNC flag until after the loop at line 450 has finished, so that the file descriptors can be closed correctly.

Impact

This issue can lead to a local denial of service attack: an unprivileged local attacker can make the system unusable for all users. For example, on Ubuntu 18.04.4 LTS, my proof-of-concept exploit prevents all users from logging in, because the login screen needs to send a D-Bus message, but the dbus-daemon is no longer able to send or receive any messages because it cannot create any new file descriptors.

Remediation

As I mentioned above, my recommendation is to delete the block of code at dbus-sysdeps-unix.c, line 438 to 448.

Resources

I have attached the source code of my proof-of-concept exploit: `fd_dos.cpp`. Compile it and run it like this:

```
gcc fd_dos.cpp -o fd_dos
./fd_dos /var/run/dbus/system_bus_socket
```

Credit

This issue was discovered and reported by GHSL team member @kevinbackhouse (Kevin Backhouse).

Contact

You can contact the GHSL team at `securitylab@github.com`, please include the `GHSL-2020-057` in any communication regarding this issue.

Disclosure Policy

This report is subject to our coordinated disclosure policy.

Edited 2 years ago by Simon McVittie

Drag your designs here or click to upload.

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Link issues together to show that they're related. Learn more.

Related merge requests 1

Make more tests modular
!153

When this merge request is accepted, this issue will be closed automatically.

Activity

Kevin Backhouse changed the description 2 years ago

Simon McVittie @smcv · 2 years ago Owner

File descriptor passing, the gift that keeps on giving :'-(

@lennart, you wrote this; does the suggested code-deletion make sense to you?

Kevin Backhouse @kevinbackhouse · 2 years ago Author

The alternative solution would be to add some code like this *after* the loop:

```
if (m.msg_flags & MSG_CTRUNC)
  {
    size_t i;
    for (i = 0; i < *n_fds; i++)
      {
        close (fds[i]);
      }
    *n_fds = 0;

    errno = ENOSPC;
    _dbus_string_set_length (buffer, start);
    return -1;
  }
```

Simon McVittie @smcv · 2 years ago

Owner

> so it is better to ignore the MSG_CTRUNC flag and process the message as normal. File descriptors will be correctly closed if the message is invalid.

I don't think that's an ideal solution. If the kernel tells us that truncation has occurred, we ought to report that to the next layer up, rather than pretending that everything is fine. Otherwise, in the worst case we might associate fds with a wrongly-chosen message from the same sender.

For historical reasons, and due to some less-precise-than-would-be-ideal specification text and the fact that we're implementing a message-oriented protocol over a kernel-side transport that has trouble deciding whether it's a stream or a series of messages, D-Bus messages that carry fds contain redundant information:

- the message header contains a `UNIX_FDS` header, which tells us the number of fds that the sender *claims* it attached
- the message header and/or body have some number of fds attached out-of-band
  - in practice all implementations that I know about attach them to the same `sendmsg()` as the first byte of the header, but the specification doesn't actually guarantee that
  - in principle the fds can be attached to any `sendmsg()` that has at least one byte of this message in its in-band payload, and a pathological sender would be allowed to send a message with 4 fds by making 4 separate `sendmsg()` calls, each containing at least one byte of the message in-band, and each with one fd attached
  - the relationship between in-band payload bytes and attached fds depends on message size, how/whether the kernel splits large writes, how/whether the kernel coalesces small writes, and other implementation details
- in principle it's an error to send more or fewer fds than the `UNIX_FDS` header says, but in practice the dbus-daemon has no good way to detect this; so we have to assume that senders know what they're doing, and to avoid DoS, ensure that if they get it wrong they can only harm themselves
- the message header and/or body may contain "handles" (type-code `h`), which are indexes into the out-of-band fd list that act as a placeholder for the fd itself
  - in practice no current message headers contain handles, but it's a possibility for future development
  - it is valid (although odd) to send a message containing fds, but no references to them in the headers or body
  - some D-Bus APIs allow accessing the fds by index even if the message body does not refer to them!
  - it is an error to send type-code `h` with a value >= the `UNIX_FDS` header (and this, at least, is easy to detect)

> The alternative solution would be to add some code like this *after* the loop

I prefer that, conceptually.

Simon McVittie @smcv · 2 years ago

Owner

This can be reproduced with the existing unit tests, without requiring privileges, which will make it a lot easier to test a proposed fix. If I hack `test/fdpass.c` to list open file descriptors before and after it tries to exceed the per-message limit:

▶ A suitable hack

Then compile dbus with GLib installed and tests enabled, and run:

```
${builddir}/test/test-fdpass -p /odd-limit
```

Expected result

In each pair of **File descriptors before** and **File descriptors after**, I get the same fds before and after.

Actual result

In the test-cases `/odd-limit/minus1` and `/odd-limit/at`, where the limit is not exceeded, I get the expected result.

In the test-cases `/odd-limit/plus1` and `/odd-limit/plus2`, where the limit *is* exceeded, the **File descriptors after** contains several extra file descriptors opened to `/dev/null`. (These are the out-of-band fd payload of the message.)

Edited by Simon McVittie 2 years ago

Simon McVittie @smcv · 2 years ago

Owner

Proposed patch

Sorry, no merge request here, because Gitlab isn't very good at embargoed merge requests.

```
From 8099849e648a0cf61d14d735cab1d8204a65969e Mon Sep 17 00:00:00 2001
From: Simon McVittie <smcv@collabora.com>
Date: Thu, 16 Apr 2020 14:45:11 +0100
Subject: [PATCH] sysdeps-unix: On MSG_CTRUNC, close the fds we did receive

MSG_CTRUNC indicates that we have received fewer fds that we should
have done because the buffer was too small, but we were treating it
as though it indicated that we received *no* fds. If we received any,
we still have to make sure we close them, otherwise they will be leaked.

On the system bus, if an attacker can induce us to leak fds in this
way, that's a local denial of service via resource exhaustion.

Reported-by: Kevin Backhouse, GitHub Security Lab
Fixes: dbus#294
Fixes: CVE-2020-12049
Fixes: GHSL-2020-057
---
 dbus/dbus-sysdeps-unix.c | 32 ++++++++++++++++++++------------------
 1 file changed, 20 insertions(+), 12 deletions(-)

diff --git a/dbus/dbus-sysdeps-unix.c b/dbus/dbus-sysdeps-unix.c
index b5fc2466..f65a248e 100644
--- a/dbus/dbus-sysdeps-unix.c
+++ b/dbus/dbus-sysdeps-unix.c
@@ -435,18 +435,6 @@ _dbus_read_socket_with_unix_fds (DBusSocket        fd,
       struct cmsghdr *cm;
       dbus_bool_t found = FALSE;

-      if (m.msg_flags & MSG_CTRUNC)
-        {
-          /* Hmm, apparently the control data was truncated. The bad
-             thing is that we might have completely lost a couple of fds
-             without chance to recover them. Hence let's treat this as a
-             serious error. */
-
-          errno = ENOSPC;
-          _dbus_string_set_length (buffer, start);
-          return -1;
-        }
-
       for (cm = CMSG_FIRSTHDR(&m); cm; cm = CMSG_NXTHDR(&m, cm))
         if (cm->cmsg_level == SOL_SOCKET && cm->cmsg_type == SCM_RIGHTS)
           {
@@ -501,6 +489,26 @@ _dbus_read_socket_with_unix_fds (DBusSocket        fd,
       if (!found)
         *n_fds = 0;

+      if (m.msg_flags & MSG_CTRUNC)
+        {
+          unsigned int i;
+
+          /* Hmm, apparently the control data was truncated. The bad
+             thing is that we might have completely lost a couple of fds
+             without chance to recover them. Hence let's treat this as a
+             serious error. */
+
+          /* We still need to close whatever fds we *did* receive,
+           * otherwise they'll never get closed. (GHSL-2020-057) */
+          for (i = 0; i < *n_fds; i++)
+            close (fds[i]);
+
+          *n_fds = 0;
+          errno = ENOSPC;
+          _dbus_string_set_length (buffer, start);
+          return -1;
+        }
+
       /* put length back (doesn't actually realloc) */
       _dbus_string_set_length (buffer, start + bytes_read);

--
2.26.1
```

Edited by Simon McVittie 2 years ago

Simon McVittie @smcv · 2 years ago

Owner

Reproducer (for git master)

This assumes !153 (merged) has been applied first, although I could adapt it to not.

```
From edb0af7b28d3f8b8d13abdad422b5c375624a039 Mon Sep 17 00:00:00 2001
From: Simon McVittie <smcv@collabora.com>
Date: Thu, 16 Apr 2020 14:41:48 +0100
Subject: [PATCH] fdpass: Assert that we don't leak file descriptors

Reproduces: dbus#294
Reproduces: CVE-2020-12049
Reproduces: GHSL-2020-057
Signed-off-by: Simon McVittie <smcv@collabora.com>
---
 test/fdpass.c | 8 ++++++++
 1 file changed, 8 insertions(+)

diff --git a/test/fdpass.c b/test/fdpass.c
index d84b7fe9..abd41504 100644
--- a/test/fdpass.c
+++ b/test/fdpass.c
@@ -92,6 +92,7 @@ typedef struct {
     GQueue messages;

     int fd_before;
+    DBusInitialFDs *initial_fds;
 } Fixture;

 static void oom (const gchar *doing) G_GNUC_NORETURN;
@@ -176,6 +177,8 @@ test_connect (Fixture *f,
   if (f->skip)
     return;

+  f->initial_fds = _dbus_check_fdleaks_enter ();
+
   g_assert (f->left_server_conn == NULL);
   g_assert (f->right_server_conn == NULL);

@@ -875,6 +878,11 @@ teardown (Fixture *f,
   if (f->fd_before >= 0 && close (f->fd_before) < 0)
     g_error ("%s", g_strerror (errno));
 #endif
+
+  /* TODO: It would be nice if we could ask GLib which test-case
+   * we're currently in */
+  if (f->initial_fds != NULL)
+    _dbus_check_fdleaks_leave (f->initial_fds, "next test-case");
 }

 int
--
2.26.1
```

Edited by Simon McVittie 2 years ago

Simon McVittie @smcv · 2 years ago Owner

Reproducer (for dbus-1.12)

Because we can't rely on refactoring like !153 (merged) on the stable branch, this will only reproduce the bug if dbus was configured with `--enable-embedded-tests` . In a Debian/Ubuntu package to which this patch has been applied, you could install the `dbus-tests` package and run the version of `test-fdpass` that is installed in `/usr/lib/dbus-1.0/debug-build/libexec/installed-tests/dbus` or some similar path (paths may vary according to the age of the package).

```
From 754b44216e15f2c536bc8a5b1b831273fe57e662 Mon Sep 17 00:00:00 2001
From: Simon McVittie <smcv@collabora.com>
Date: Thu, 16 Apr 2020 14:41:48 +0100
Subject: [PATCH] fdpass: Assert that we don't leak file descriptors

This version is for the dbus-1.12 branch, and doesn't rely on dbus!153
or dbus!120.

Reproduces: dbus#294
Reproduces: CVE-2020-12049
Reproduces: GHSL-2020-057
Signed-off-by: Simon McVittie <smcv@collabora.com>
---
 test/fdpass.c | 14 ++++++++++++++
 1 file changed, 14 insertions(+)

diff --git a/test/fdpass.c b/test/fdpass.c
index 4a3edc4e..8bad675f 100644
--- a/test/fdpass.c
+++ b/test/fdpass.c
@@ -50,6 +50,14 @@

 #include "test-utils-glib.h"

+#ifdef DBUS_ENABLE_EMBEDDED_TESTS
+#include <dbus/dbus-message-internal.h>
+#else
+typedef struct _DBusInitialFDs DBusInitialFDs;
+#define _dbus_check_fdleaks_enter() NULL
+#define _dbus_check_fdleaks_leave(fds) do {} while (0)
+#endif
+
 /* Arbitrary; included here to avoid relying on the default */
 #define MAX_MESSAGE_UNIX_FDS 20
 /* This test won't work on Linux unless this is true. */
@@ -92,6 +100,7 @@ typedef struct {
     GQueue messages;

     int fd_before;
+    DBusInitialFDs *initial_fds;
 } Fixture;

 static void oom (const gchar *doing) G_GNUC_NORETURN;
@@ -176,6 +185,8 @@ test_connect (Fixture *f,
   if (f->skip)
     return;

+  f->initial_fds = _dbus_check_fdleaks_enter ();
+
   g_assert (f->left_server_conn == NULL);
   g_assert (f->right_server_conn == NULL);

@@ -871,6 +882,9 @@ teardown (Fixture *f,
   if (f->fd_before >= 0 && close (f->fd_before) < 0)
     g_error ("%s", g_strerror (errno));
 #endif
+
+  if (f->initial_fds != NULL)
+    _dbus_check_fdleaks_leave (f->initial_fds);
 }

 int
--
2.26.1
```

Edited by Simon McVittie 2 years ago

Simon McVittie @smcv · 2 years ago Owner

/cc @poettering

Lennart Poettering @poettering · 2 years ago Maintainer

In sd-bus we don#t check for CTRUNC. But it might make sense to change that, it's probably safer to do that, indeed. So I think Simon's patch looks OK. And yeah, if one gets a message which one then deems not good enough, the error path for that of course should close all fds again...

Simon McVittie @smcv · 2 years ago Owner

@kevinbackhouse, do you allocate CVE IDs or do we need to get one of those from elsewhere?

We currently do treat local denial of service via resource exhaustion as a security vulnerability.

(However, in the past I've considered changing our policy to treat instances of local DoS via resource exhaustion as "just a bug", since local users can often cause DoS via resource exhaustion *anyway* on practical systems, and we've had at least one case where fixing an obscure attack of this type led to bugs that broke startup of systems that were *not* under active attack, which was arguably a worse DoS than the original problem.)

Kevin Backhouse @kevinbackhouse · 2 years ago Author

@smcv: I am happy to request a CVE for this from Mitre.

Simon McVittie @smcv · 2 years ago Owner

@kevinbackhouse Thanks for offering, but if you'd have to request a CVE from MITRE the same way anyone else can, I might as well do it myself, since I'll have a better overview of the release procedure than you do. I wasn't sure whether Github had its own CNA or similar side-channel.

Edited by Simon McVittie 2 years ago

Simon McVittie @smcv · 2 years ago Owner

CVE ID requested from MITRE.

Kevin Backhouse @kevinbackhouse · 2 years ago Author

@smvc: Thanks! I agree it makes sense for you to apply for the CVE yourself, if you're happy to do it. (I know it's tedious and probably not what you want to be spending your time doing.)

GitHub did recently become a CNA, but that's to make it easier for project maintainers to request CVEs for their own projects. We don't use it for bugs found by GitHub Security Lab because we think that would be a bit like marking our own homework. So we still request CVEs from Mitre.

Simon McVittie @smcv · 2 years ago Owner

This is now officially CVE-2020-12049.

I'm intending to unembargo this sometime in the next couple of weeks (with my downstream maintainer hat on, I need to talk to the Debian security team about whether they'll accept a new upstream release with non-security fixes in it). Because it's only a DoS, I'm not intending to do much coordination or pre-announcement to distros.

Simon McVittie changed title from **File descriptor leak in _dbus_read_socket_with_unix_fds** to **CVE-2020-12049: File descriptor leak in _dbus_read_socket_with_unix_fds** 2 years ago

Simon McVittie @smcv · 2 years ago Owner

Affected releases

dbus >= 1.3.0, except as noted below.

(In particular, dbus 1.10.28, 1.12.16 and 1.13.14, and older versions from each of those branches, are vulnerable.)

Fixed releases

- dbus 1.10.x >= 1.10.30 are intended to include a fix
  - I need to test this, but patches intended for 1.12.x probably apply OK
- dbus 1.12.x >= 1.12.18 are intended to include a fix
  - The patch above has been tested
- dbus 1.13.x >= 1.13.16 are intended to include a fix
  - The patch above has been tested

Unfixed end-of-life releases

dbus 1.8.x and all older branches have reached end-of-life and will not receive releases to address this. Distributions that still maintain versions derived from these branches are invited to use the upstream dbus git repository to share fixes (please contact the dbus maintainers to arrange this).

Simon McVittie @smcv · 2 years ago Owner

It's a while since we did our last round of releases, so I've done a 1.13.14 release *without* this change in it, to make sure we don't get any nasty surprises when trying to do a release *with* this fix later.

Simon McVittie mentioned in commit `fc0f2965` 2 years ago

Simon McVittie closed via commit `872b085f` 2 years ago

Simon McVittie mentioned in commit `272d4842` 2 years ago

Simon McVittie mentioned in commit `8bc13818` 2 years ago

Simon McVittie mentioned in commit `3418f4e5` 2 years ago

Simon McVittie mentioned in commit `7131a480` 2 years ago

Simon McVittie @smcv · 2 years ago Owner

Fixed in versions >= 1.13.16, 1.2.x >= 1.12.18, 1.10.x >= 1.10.30. Unembargoing.

Simon McVittie made the issue visible to everyone 2 years ago

Simon McVittie mentioned in issue #308 2 years ago

Please register or sign in to reply