

linux-media.vger.kernel.org archive mirror

search	help / color / mirror / Atom feed
--------	-----------------------------------

From: imv4bel@gmail.com
To: mchehab@kernel.org
Cc: Hyunwoo Kim <imv4bel@gmail.com>, kernel@tuxforce.de, linux-media@vger.kernel.org, linux-usb@vger.kernel.org, cai.huoqing@linux.dev, tiwai@suse.de
Subject: [PATCH 1/4] media: dvb-core: Fix use-after-free due to race condition occurring in dvb_frontend_release
Date: Tue, 15 Nov 2022 05:18:19 -0800 [thread overview]
Message-ID: <20221115131822.6640-2-imv4bel@gmail.com> (raw)
In-Reply-To: <20221115131822.6640-1-imv4bel@gmail.com>

From: Hyunwoo Kim <imv4bel@gmail.com>

If the device node of dvb frontend is open() and the device is disconnected, many kinds of UAFs may occur when calling close() on the device node.

The root cause of this is that wake_up() for dvbdev->wait_queue is implemented in the dvb_frontend_release() function, but wait_event() is not implemented in the dvb_frontend_stop() function.

So, implement wait_event() function in dvb_frontend_stop() and add 'remove_mutex' which prevents race condition for 'fe->exit'.

Signed-off-by: Hyunwoo Kim <imv4bel@gmail.com>

```
---
drivers/media/dvb-core/dvb_frontend.c | 39 ++++++
include/media/dvb_frontend.h         | 6 +++
2 files changed, 39 insertions(+), 6 deletions(-)

diff --git a/drivers/media/dvb-core/dvb_frontend.c b/drivers/media/dvb-core/dvb_frontend.c
index 48e735cdbe6b..b3556e3580c6 100644
--- a/drivers/media/dvb-core/dvb_frontend.c
+++ b/drivers/media/dvb-core/dvb_frontend.c
@@ -809,6 +809,8 @@ static void dvb_frontend_stop(struct dvb_frontend *fe)
{
    dev_dbg(fe->dvb->device, "%s:\n", __func__);

+    mutex_lock(&fe->remove_mutex);
+
    if (fe->exit != DVB_FE_DEVICE_REMOVED)
        fe->exit = DVB_FE_NORMAL_EXIT;
    mb();
@@ -818,6 +820,13 @@ static void dvb_frontend_stop(struct dvb_frontend *fe)
{
    kthread_stop(fepriv->thread);

+    mutex_unlock(&fe->remove_mutex);
+
+    if (fepriv->dvbdev->users < -1) {
+        wait_event(fepriv->dvbdev->wait_queue,
+            fepriv->dvbdev->users == -1);
+    }
+
    sema_init(&fepriv->sem, 1);
    fepriv->state = FESTATE_IDLE;
@@ -2750,9 +2759,13 @@ static int dvb_frontend_open(struct inode *inode, struct file *file)
{
    struct dvb_adapter *adapter = fe->dvb;
    int ret;

+    mutex_lock(&fe->remove_mutex);
+
    dev_dbg(fe->dvb->device, "%s:\n", __func__);
    if (fe->exit == DVB_FE_DEVICE_REMOVED)
+    if (fe->exit == DVB_FE_DEVICE_REMOVED) {
+        mutex_unlock(&fe->remove_mutex);
+        return -ENODEV;
+    }

    if (adapter->mfe_shared) {
        mutex_lock(&adapter->mfe_lock);
@@ -2773,8 +2786,10 @@ static int dvb_frontend_open(struct inode *inode, struct file *file)
        while (mferetry-- && (mfepriv->users != -1 ||
            mfepriv->thread)) {
            if (msleep_interruptible(500)) {
-                if (signal_pending(current))
+                if (signal_pending(current)) {
+                    mutex_unlock(&fe->remove_mutex);
+                    return -EINTR;
+                }
            }
        }
    }
@@ -2786,6 +2801,7 @@ static int dvb_frontend_open(struct inode *inode, struct file *file)
    if (mfepriv->users != -1 ||
        mfepriv->thread) {
        mutex_unlock(&adapter->mfe_lock);
+        mutex_unlock(&fe->remove_mutex);
        return -EBUSY;
    }
    adapter->mfe_dvbdev = dvbdev;
@@ -2845,6 +2861,8 @@ static int dvb_frontend_open(struct inode *inode, struct file *file)
{
    if (adapter->mfe_shared)
        mutex_unlock(&adapter->mfe_lock);
+    mutex_unlock(&fe->remove_mutex);
    return ret;
}

err3:
@@ -2866,6 +2884,8 @@ static int dvb_frontend_open(struct inode *inode, struct file *file)
err0:
    if (adapter->mfe_shared)
        mutex_unlock(&adapter->mfe_lock);
+    mutex_unlock(&fe->remove_mutex);
    return ret;
}

@@ -2876,6 +2896,8 @@ static int dvb_frontend_release(struct inode *inode, struct file *file)
{
    struct dvb_frontend_private *fepriv = fe->frontend_priv;
    int ret;

+    mutex_lock(&fe->remove_mutex);
+
    dev_dbg(fe->dvb->device, "%s:\n", __func__);

    if ((file->f_flags & O_ACCMODE) != O_RDONLY) {
@@ -2897,11 +2919,17 @@ static int dvb_frontend_release(struct inode *inode, struct file *file)
    }
    mutex_unlock(&fe->dvb->mdev_lock);
}

-#endif
-
+    if (fe->exit != DVB_FE_NO_EXIT)
+        wake_up(&dvbdev->wait_queue);
+    if (fe->ops.ts_bus_ctrl)
+        fe->ops.ts_bus_ctrl(fe, 0);
+
+    if (fe->exit != DVB_FE_NO_EXIT) {
+        mutex_unlock(&fe->remove_mutex);
+        wake_up(&dvbdev->wait_queue);
+    } else
+        mutex_unlock(&fe->remove_mutex);
+
+    } else
+        mutex_unlock(&fe->remove_mutex);
}

```

```

        dvb_frontend_put(fe);

@@ -3000,6 +3028,7 @@ int dvb_register_frontend(struct dvb_adapter *dvb,
        fepriv = fe->frontend_priv;

        kref_init(&fe->refcount);
        mutex_init(&fe->remove_mutex);

/*
 * After initialization, there need to be two references: one
diff --git a/include/media/dvb_frontend.h b/include/media/dvb_frontend.h
index e7c44870f20d..411ec32cd8df 100644
--- a/include/media/dvb_frontend.h
+++ b/include/media/dvb_frontend.h
@@ -686,7 +686,10 @@ struct dtv_frontend_properties {
        * @id: Frontend ID
        * @exit: Used to inform the DVB core that the frontend
        * thread should exit (usually, means that the hardware
        * got disconnected.
        * got disconnected.)
+ * @remove_mutex: mutex that avoids a race condition between a callback
+ * called when the hardware is disconnected and the
+ * file_operations of dvb_frontend
 */

struct dvb_frontend {
@@ -704,6 +707,7 @@ struct dvb_frontend {
        int (*callback)(void *adapter_priv, int component, int cmd, int arg);
        int id;
        unsigned int exit;
+ struct mutex remove_mutex;
};

/**
--
2.25.1

```

next prev parent reply other threads:[~2022-11-15 13:19 UTC|newest]

Thread overview: 6+ messages / expand[flat|nested] mbox.gz Atom feed top
2022-11-15 13:18 [PATCH 0/4] Fix multiple race condition vulnerabilities in dvb-core and device driver imv4bel
2022-11-15 13:18 ` [imv4bel \[this message\]](#)
2022-11-15 13:18 ` [PATCH 2/4] media: dvb-core: Fix use-after-free due to race condition occurring in dvb_net imv4bel
2022-11-15 13:18 ` [PATCH 3/4] media: dvb-core: Fix use-after-free due to race condition occurring in dvb_register_device() imv4bel
2022-11-17 4:16 ` Dan Carpenter
2022-11-15 13:18 ` [PATCH 4/4] media: ttusb-dec: Fix memory leak in ttusb_dec_exit_dvb() imv4bel

find likely ancestor, descendant, or conflicting patches for this message:

dfblob:48e735cdbe6	dfblob:e7c44870f20	dfblob:b3556e3580c
dfblob:411ec32cd8d		

search	(help)
--------	--------

Reply instructions:

You may reply publicly to [this message](#) via plain-text email using any one of the following methods:

- * Save the following mbox file, import it into your mail client, and reply-to-all from there: [mbox](#)

Avoid top-posting and favor interleaved quoting:
https://en.wikipedia.org/wiki/Posting_style#interleaved_style

- * Reply using the `--to`, `--cc`, and `--in-reply-to` switches of `git-send-email(1)`:

```

git send-email \
--in-reply-to=20221115131822.6640-2-imv4bel@gmail.com \
--to=imv4bel@gmail.com \
--cc=cai.huoging@linux.dev \
--cc=kernel@tuxforce.de \
--cc=linux-media@vger.kernel.org \
--cc=linux-usb@vger.kernel.org \
--cc=mchehab@kernel.org \
--cc=tiwai@suse.de \
/path/to/YOUR_REPLY

```

<https://kernel.org/pub/software/scm/git/docs/git-send-email.html>

- * If your mail client supports setting the **In-Reply-To** header via `mailto:` links, try the `mailto:` [link](#)

Be sure your reply has a **Subject:** header at the top and a blank line before the message body.

This is a public inbox, see [mirroring instructions](#) for how to clone and mirror all data and code used for this inbox; as well as URLs for NNTP newsgroup(s).