

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.huoping@linux.dev, tiwai@suse.de
 Subject: [PATCH 2/4] media: dvb-core: Fix use-after-free due to race condition occurring in dvb_net
 Date: Tue, 15 Nov 2022 05:18:20 -0800 [thread overview]
 Message-ID: <20221115131822.6640-3-imv4bel@gmail.com> (raw)
 In-Reply-To: <20221115131822.6640-1-imv4bel@gmail.com>

From: Hyunwoo Kim <imv4bel@gmail.com>

A race condition may occur between the .disconnect function, which is called when the device is disconnected, and the dvb_device_open() function, which is called when the device node is open()ed. This results in several types of UAFs.

The root cause of this is that you use the dvb_device_open() function, which does not implement a conditional statement that checks 'dvbnet->exit'.

So, add 'remove_mutex' to protect 'dvbnet->exit' and use locked_dvb_net_open() function to check 'dvbnet->exit'.

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

```
---
drivers/media/dvb-core/dvb_net.c | 37 ++++++
include/media/dvb_net.h          | 4 +++
2 files changed, 38 insertions(+), 3 deletions(-)

diff --git a/drivers/media/dvb-core/dvb_net.c b/drivers/media/dvb-core/dvb_net.c
index 8a2feb33ce2..bdfc6609cb93 100644
--- a/drivers/media/dvb-core/dvb_net.c
+++ b/drivers/media/dvb-core/dvb_net.c
@@ -1564,15 +1564,42 @@ static long dvb_net_ioctl(struct file *file,
     return dvb_usercopy(file, cmd, arg, dvb_net_do_ioctl);
 }

+static int locked_dvb_net_open(struct inode *inode, struct file *file)
+{
+    struct dvb_device *dvbdev = file->private_data;
+    struct dvb_net *dvbnet = dvbdev->priv;
+    int ret;
+
+    if (mutex_lock_interruptible(&dvbnet->remove_mutex))
+        return -ERESTARTSYS;
+
+    if (dvbnet->exit) {
+        mutex_unlock(&dvbnet->remove_mutex);
+        return -ENODEV;
+    }
+
+    ret = dvb_generic_open(inode, file);
+
+    mutex_unlock(&dvbnet->remove_mutex);
+
+    return ret;
+}
+
static int dvb_net_close(struct inode *inode, struct file *file)
{
    struct dvb_device *dvbdev = file->private_data;
    struct dvb_net *dvbnet = dvbdev->priv;

+    mutex_lock(&dvbnet->remove_mutex);
+
    dvb_generic_release(inode, file);

-    if (dvbdev->users == 1 && dvbnet->exit == 1)
+    if (dvbdev->users == 1 && dvbnet->exit == 1) {
+        mutex_unlock(&dvbnet->remove_mutex);
+        wake_up(&dvbdev->wait_queue);
+    } else
+        mutex_unlock(&dvbnet->remove_mutex);

    return 0;
}

@@ -1580,7 +1607,7 @@ static int dvb_net_close(struct inode *inode, struct file *file)
static const struct file_operations dvb_net_fops = {
    .owner = THIS_MODULE,
    .unlocked_ioctl = dvb_net_ioctl,
-    .open = dvb_generic_open,
+    .open = locked_dvb_net_open,
    .release = dvb_net_close,
    .llseek = noop_llseek,
};

@@ -1599,10 +1626,13 @@ void dvb_net_release (struct dvb_net *dvbnet)
{
    int i;

+    mutex_lock(&dvbnet->remove_mutex);
+    dvbnet->exit = 1;
+    mutex_unlock(&dvbnet->remove_mutex);
+
    if (dvbnet->dvbdev->users < 1)
        wait_event(dvbnet->dvbdev->wait_queue,
-            dvbnet->dvbdev->users==1);
+            dvbnet->dvbdev->users == 1);

    dvb_unregister_device(dvbnet->dvbdev);
}

@@ -1621,6 +1651,7 @@ int dvb_net_init (struct dvb_adapter *adap, struct dvb_net *dvbnet,
int i;

+    mutex_init(&dvbnet->iocctl_mutex);
+    mutex_init(&dvbnet->remove_mutex);
+    dvbnet->demux = dmux;

    for (i=0; i<DVB_NET_DEVICES_MAX; i++)
diff --git a/include/media/dvb_net.h b/include/media/dvb_net.h
index 5e31d37f25fa..3e2ee5a05e5 100644
--- a/include/media/dvb_net.h
+++ b/include/media/dvb_net.h
@@ -41,6 +41,9 @@
 * @exit: flag to indicate when the device is being removed.
 * @demux: pointer to &struct dmux_demux.
 * @iocctl_mutex: protect access to this struct.
+ * @remove_mutex: mutex that avoids a race condition between a callback
+ *                called when the hardware is disconnected and the
+ *                file_operations of dvb_net
 *
 * Currently, the core supports up to %DVB_NET_DEVICES_MAX (10) network
 * devices.
@@ -53,6 +56,7 @@ struct dvb_net {
    unsigned int exit;
    struct dmux_demux *demux;
    struct mutex iocctl_mutex;
+    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 [PATCH 1/4] media: dvb-core: Fix use-after-free due to race condition occurring in dvb_frontend imv4bel
2022-11-15 13:18 imv4bel [this message]
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:8a2feb33ce dfblob:5e31d37f25f dfblob:bdfc6609cb9
dfblob:3e2eee5a05e

(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=20221115131822.6640-3-imv4bel@gmail.com \
  --to=imv4bel@gmail.com \
  --cc=cai.huoqing@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).