author	Vladimir Oltean <vladimir.oltean@nxp.com>	2025-04-15 00:29:13 +0300
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2025-04-25 10:47:46 +0200
commit	9ee6d3a368ed34f2457863da3085c676e9e37a3d (patch)	
tree	0197ae63803a7ec3b8009c33b343176a83662c0c	
parent	3665695e3572239dc233216f06b41f40cc771889 (diff)	
download	linux-9ee6d3a368ed34f2457863da3085c676e9e37a3d.tar.gz	

net: dsa: mv88e6xxx: fix -ENOENT when deleting VLANs and MST is unsupported

[Upstream commit ea08dfc35f83cfc73493c52f63ae4f2e29edfe8d]

Russell King reports that on the ZII dev rev B, deleting a bridge VLAN
from a user port fails with -ENOENT:
https://lore.kernel.org/netdev/Z_lQXNP0s5-IiJzd@shell.armlinux.org.uk/

This comes from mv88e6xxx_port_vlan_leave() -> mv88e6xxx_mst_put(),
which tries to find an MST entry in &chip->msts associated with the SID,
but fails and returns -ENOENT as such.

But we know that this chip does not support MST at all, so that is not
surprising. The question is why does the guard in mv88e6xxx_mst_put()
not exit early:

```
        if (!sid)
                return 0;
```

And the answer seems to be simple: the sid comes from vlan.sid which
supposedly was previously populated by mv88e6xxx_vtu_get().
But some chip->info->ops->vtu_getnext() implementations do not populate
vlan.sid, for example see mv88e6185_g1_vtu_getnext(). In that case,
later in mv88e6xxx_port_vlan_leave() we are using a garbage sid which is
just residual stack memory.

Testing for sid == 0 covers all cases of a non-bridge VLAN or a bridge
VLAN mapped to the default MSTI. For some chips, SID 0 is valid and
installed by mv88e6xxx_stu_setup(). A chip which does not support the
STU would implicitly only support mapping all VLANs to the default MSTI,
so although SID 0 is not valid, it would be sufficient, if we were to
zero-initialize the vlan structure, to fix the bug, due to the
coincidence that a test for vlan.sid == 0 already exists and leads to
the same (correct) behavior.

Another option which would be sufficient would be to add a test for
mv88e6xxx_has_stu() inside mv88e6xxx_mst_put(), symmetric to the one
which already exists in mv88e6xxx_mst_get(). But that placement means
the caller will have to dereference vlan.sid, which means it will access
uninitialized memory, which is not nice even if it ignores it later.

So we end up making both modifications, in order to not rely just on the
sid == 0 coincidence, but also to avoid having uninitialized structure

fields which might get temporarily accessed.

Fixes: acaf4d2e36b3 ("net: dsa: mv88e6xxx: MST Offloading")
Signed-off-by: Vladimir Oltean <vladimir.oltean@nxp.com>
Link: https://patch.msgid.link/20250414212913.2955253-1-vladimir.oltean@nxp.com
Signed-off-by: Jakub Kicinski <kuba@kernel.org>
Signed-off-by: Sasha Levin <sashal@kernel.org>

Diffstat

-rw-r--r-- drivers/net/dsa/mv88e6xxx/chip.c 13

1 files changed, 12 insertions, 1 deletions

```
diff --git a/drivers/net/dsa/mv88e6xxx/chip.c b/drivers/net/dsa/mv88e6xxx/chip.c
index e20d9d62032e31..df1df601541217 100644
--- a/drivers/net/dsa/mv88e6xxx/chip.c
+++ b/drivers/net/dsa/mv88e6xxx/chip.c
@@ -1878,6 +1878,8 @@ static int mv88e6xxx_vtu_get(struct mv88e6xxx_chip *chip, u16 vid,
        if (!chip->info->ops->vtu_getnext)
                return -EOPNOTSUPP;

+       memset(entry, 0, sizeof(*entry));
+
        entry->vid = vid ? vid - 1 : mv88e6xxx_max_vid(chip);
        entry->valid = false;

@@ -2013,7 +2015,16 @@ static int mv88e6xxx_mst_put(struct mv88e6xxx_chip *chip, u8 sid)
        struct mv88e6xxx_mst *mst, *tmp;
        int err;

-       if (!sid)
+       /* If the SID is zero, it is for a VLAN mapped to the default MSTI,
+        * and mv88e6xxx_stu_setup() made sure it is always present, and thus,
+        * should not be removed here.
+        *
+        * If the chip lacks STU support, numerically the "sid" variable will
+        * happen to also be zero, but we don't want to rely on that fact, so
+        * we explicitly test that first. In that case, there is also nothing
+        * to do here.
+        */
+       if (!mv88e6xxx_has_stu(chip) || !sid)
                return 0;

        list_for_each_entry_safe(mst, tmp, &chip->msts, node) {
```