



author Jakub Kicinski <kuba@kernel.org> 2025-04-04 11:03:33 -0700
committer Greg Kroah-Hartman <gregkh@linuxfoundation.org> 2025-04-20 10:17:30 +0200
commit c665bef891e8972e1d3ce5bbc0d42a373346a2c3 (patch)
tree 18f17214b1e00004aefe54fb3a68c2fdcfa0144
parent e73c838c80dcc9e4f19becc11d9f3cb4a27d483 (diff)
download linux-c665bef891e8972e1d3ce5bbc0d42a373346a2c3.tar.gz

diff options

context: 3
space: include
mode: unified

net: tls: explicitly disallow disconnect

[Upstream commit 5071a1e606b30c0c11278d3c6620cd6a24724cf6]

syzbot discovered that it can disconnect a TLS socket and then run into all sort of unexpected corner cases. I have a vague recollection of Eric pointing this out to us a long time ago. Supporting disconnect is really hard, for one thing if offload is enabled we'd need to wait for all packets to be _acked_. Disconnect is not commonly used, disallow it.

The immediate problem syzbot run into is the warning in the strp, but that's just the easiest bug to trigger:

```
WARNING: CPU: 0 PID: 5834 at net/tls/tls_strp.c:486 tls_strp_msg_load+0x72e/0xa80 net/tls/tls_strp.c:486
RIP: 0010:tls_strp_msg_load+0x72e/0xa80 net/tls/tls_strp.c:486
Call Trace:
<TASK>
tls_rx_rec_wait+0x280/0xa60 net/tls/tls_sw.c:1363
tls_sw_recvmmsg+0x85c/0x1c30 net/tls/tls_sw.c:2043
inet6_recvmmsg+0x2c9/0x730 net/ipv6/af_inet6.c:678
sock_recvmmsg_nosec net/socket.c:1023 [inline]
sock_recvmmsg+0x109/0x280 net/socket.c:1045
__sys_recvfrom+0x202/0x380 net/socket.c:2237
```

Fixes: 3c4d7559159b ("tls: kernel TLS support")
Reported-by: syzbot+b4cd76826045a1eb93c1@syzkaller.appspotmail.com
Signed-off-by: Jakub Kicinski <kuba@kernel.org>
Reviewed-by: Eric Dumazet <edumazet@google.com>
Reviewed-by: Sabrina Dubroca <sd@queasysnail.net>
Link: <https://patch.msgid.link/20250404180334.3224206-1-kuba@kernel.org>
Signed-off-by: Paolo Abeni <pabeni@redhat.com>
Signed-off-by: Sasha Levin <sashal@kernel.org>

Diffstat

```
-rw-r--r-- net/tls/tls_main.c 6
```

1 files changed, 6 insertions, 0 deletions

diff --git a/net/tls/tls_main.c b/net/tls/tls_main.c

index 6b4b9f2749a6fd..0acf313deb01ff 100644

--- a/net/tls/tls_main.c

+++ b/net/tls/tls_main.c

```
@@ -809,6 +809,11 @@ static int tls_setsockopt(struct sock *sk, int level, int optname,
    return do_tls_setsockopt(sk, optname, optval, optlen);
}
```

```
+static int tls_disconnect(struct sock *sk, int flags)
+{
```

```

+         return -EOPNOTSUPP;
+     }
+
+     struct tls_context *tls_ctx_create(struct sock *sk)
+     {
+         struct inet_connection_sock *icsk = inet_csk(sk);
@@ -904,6 +909,7 @@ static void build_protos(struct proto prot[TLS_NUM_CONFIG][TLS_NUM_CONFIG],
+         prot[TLS_BASE][TLS_BASE] = *base;
+         prot[TLS_BASE][TLS_BASE].setsockopt      = tls_setsockopt;
+         prot[TLS_BASE][TLS_BASE].getsockopt      = tls_getsockopt;
+         prot[TLS_BASE][TLS_BASE].disconnect      = tls_disconnect;
+         prot[TLS_BASE][TLS_BASE].close           = tls_sk_proto_close;

+         prot[TLS_SW][TLS_BASE] = prot[TLS_BASE][TLS_BASE];

```