author	Namjae Jeon <linkinjeon@kernel.org>	2025-04-11 15:19:46 +0900
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2025-05-02 07:58:57 +0200
commit	1aec4d14cf81b7b3e7b69eb1cfa94144eed7138e (patch)	
tree	c2a53011840191dc32d8f345ba89cb14947c3cb5	
parent	0fc403192dcc8def1f6284959141608ac4c86699 (diff)	
download	linux-1aec4d14cf81b7b3e7b69eb1cfa94144eed7138e.tar.gz	

ksmbd: fix use-after-free in __smb2_lease_break_noti()

[Upstream commit 21a4e47578d44c6b37c4fc4aba8ed7cc8dbb13de]

Move tcp_transport free to ksmbd_conn_free. If ksmbd connection is
referenced when ksmbd server thread terminates, It will not be freed,
but conn->tcp_transport is freed. __smb2_lease_break_noti can be performed
asynchronously when the connection is disconnected. __smb2_lease_break_noti
calls ksmbd_conn_write, which can cause use-after-free
when conn->ksmbd_transport is already freed.

Cc: stable@vger.kernel.org
Reported-by: Norbert Szetei <norbert@doyensec.com>
Tested-by: Norbert Szetei <norbert@doyensec.com>
Signed-off-by: Namjae Jeon <linkinjeon@kernel.org>
Signed-off-by: Steve French <stfrench@microsoft.com>
Signed-off-by: Sasha Levin <sashal@kernel.org>

Diffstat

-rw-r--r--	fs/smb/server/connection.c	4
-rw-r--r--	fs/smb/server/transport_tcp.c	14
-rw-r--r--	fs/smb/server/transport_tcp.h	1

3 files changed, 13 insertions, 6 deletions

```
diff --git a/fs/smb/server/connection.c b/fs/smb/server/connection.c
index b0f69cee96a758..7aaea71a4f2061 100644
--- a/fs/smb/server/connection.c
+++ b/fs/smb/server/connection.c
@@ -39,8 +39,10 @@ void ksmbd_conn_free(struct ksmbd_conn *conn)
        xa_destroy(&conn->sessions);
        kvfree(conn->request_buf);
        kfree(conn->preauth_info);
-       if (atomic_dec_and_test(&conn->refcnt))
+       if (atomic_dec_and_test(&conn->refcnt)) {
+               ksmbd_free_transport(conn->transport);
                kfree(conn);
+       }
 }

 /**
```

```
diff --git a/fs/smb/server/transport_tcp.c b/fs/smb/server/transport_tcp.c
index 7f38a3c3f5bd69..abedf510899a74 100644
```

```
--- a/fs/smb/server/transport_tcp.c
+++ b/fs/smb/server/transport_tcp.c
@@ -93,17 +93,21 @@ static struct tcp_transport *alloc_transport(struct socket *client_sk)
        return t;
 }

-static void free_transport(struct tcp_transport *t)
+void ksmbd_free_transport(struct ksmbd_transport *kt)
 {
-       kernel_sock_shutdown(t->sock, SHUT_RDWR);
-       sock_release(t->sock);
-       t->sock = NULL;
+       struct tcp_transport *t = TCP_TRANS(kt);

-       ksmbd_conn_free(KSMBD_TRANS(t)->conn);
+       sock_release(t->sock);
        kfree(t->iov);
        kfree(t);
 }

+static void free_transport(struct tcp_transport *t)
+{
+       kernel_sock_shutdown(t->sock, SHUT_RDWR);
+       ksmbd_conn_free(KSMBD_TRANS(t)->conn);
+}
+
 /**
  * kvec_array_init() - initialize a IO vector segment
  * @new:       IO vector to be initialized

diff --git a/fs/smb/server/transport_tcp.h b/fs/smb/server/transport_tcp.h
index 8c9aa624cfe3ca..1e51675ee1b209 100644
--- a/fs/smb/server/transport_tcp.h
+++ b/fs/smb/server/transport_tcp.h
@@ -8,6 +8,7 @@

 int ksmbd_tcp_set_interfaces(char *ifc_list, int ifc_list_sz);
 struct interface *ksmbd_find_netdev_name_iface_list(char *netdev_name);
+void ksmbd_free_transport(struct ksmbd_transport *kt);
 int ksmbd_tcp_init(void);
 void ksmbd_tcp_destroy(void);
```