Netbird

- Recovering NetBird Owner Access on a Self-Hosted Deployment

Recovering NetBird Owner Access on a Self-Hosted Deployment

This tutorial documents how to recover **Owner** access in a self-hosted NetBird deployment when the account you can log into is only an **Admin**, and the original **Owner** account is inaccessible or duplicated.

Important: This process directly edits the NetBird SQLite database. Always stop the NetBird containers and create a backup before making any changes.

Scenario

In this case, the NetBird dashboard showed multiple users with the same name/email. The account currently being used was listed as **Admin**, while another duplicate account was listed as **Owner**.

Because only an Owner can transfer ownership from the NetBird dashboard, we had to correct the role directly in the NetBird database.

Environment

- Self-hosted NetBird
- Docker Compose deployment
- NetBird server container: `netbirdio/netbird-server:latest`
- Database: SQLite
- Host OS: Ubuntu

Step 1: Confirm Running Containers

First, list all NetBird-related containers:

```
docker ps -a
```

Example output:

```
CONTAINER ID   IMAGE                              STATUS
NAMES
11e45271e508   netbirdio/reverse-proxy:latest     Up 47 hours
netbird-proxy
87c06d43a648   traefik:v3.6                       Up 47 hours
netbird-traefik
40d2c469d27a   crowdsecurity/crowdsec:v1.7.7      Up 47 hours (healthy)
netbird-crowdsec
a7e45677a803   netbirdio/dashboard:latest         Up 47 hours
netbird-dashboard
39fc0dfa7b68   netbirdio/netbird-server:latest    Up 47 hours
netbird-server
```

You can also use a cleaner formatted view:

```
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}"
```

Step 2: Locate the NetBird Data Volume

Inspect the `netbird-server` container mounts:

```
docker inspect netbird-server --format '{{json .Mounts}}' | jq
```

Example output:

```
[
  {
    "Type": "volume",
    "Name": "ubuntu_netbird_data",
    "Source": "/var/lib/docker/volumes/ubuntu_netbird_data/_data",
    "Destination": "/var/lib/netbird",
    "Driver": "local",
    "Mode": "rw",
    "RW": true,
    "Propagation": ""
  },
  {
    "Type": "bind",
    "Source": "/home/ubuntu/config.yaml",
    "Destination": "/etc/netbird/config.yaml",
    "Mode": "rw",
    "RW": true,
    "Propagation": "rprivate"
  }
]
```


In this example, the NetBird data is stored on the host at:

```
/var/lib/docker/volumes/ubuntu_netbird_data/_data
```

Step 3: Verify the Database Files

List the contents of the NetBird data directory:

```
sudo ls -lah /var/lib/docker/volumes/ubuntu_netbird_data/_data
```

Example output:

```
total 71M
drwxr-xr-x 2 root root 4.0K May 11 00:01 .
drwx-----x 3 root root 4.0K May  9 00:55 ..
-rw-r--r-- 1 root root  63M May  9 00:55 GeoLite2-City_20260428.mmdb
-rw-r--r-- 1 root root  52K May 10 23:52 events.db
-rw-r--r-- 1 root root 7.1M May  9 00:55 geonames_20260428.db
-rw-r--r-- 1 root root 104K May 10 13:40 idp.db
-rw-r--r-- 1 root root 520K May 11 00:01 store.db
```

The main NetBird database in this deployment was:

```
store.db
```

You can also search for database files:

```
sudo find /var/lib/docker/volumes/ubuntu_netbird_data/_data -maxdepth 3 -type
f \( -name "*.db" -o -name "*.sqlite" -o -name "*store*" \) -print
```

Example output:

```
/var/lib/docker/volumes/ubuntu_netbird_data/_data/events.db
/var/lib/docker/volumes/ubuntu_netbird_data/_data/geonames_20260428.db
/var/lib/docker/volumes/ubuntu_netbird_data/_data/idp.db
/var/lib/docker/volumes/ubuntu_netbird_data/_data/store.db
```

Step 4: Stop the NetBird Stack

Before editing the database, stop the NetBird containers:

```
cd /home/ubuntu
docker compose down
```

Example output:

```
[+] down 6/6
 ? Container netbird-proxy      Removed
 ? Container netbird-traefik    Removed
 ? Container netbird-dashboard  Removed
 ? Container netbird-server     Removed
 ? Container netbird-crowdsec   Removed
 ? Network ubuntu_netbird       Removed
```

Why stop the containers? Editing the database while NetBird is running can cause corruption or cause your changes to be overwritten.

Step 5: Back Up the NetBird Data Volume

Create a compressed backup of the NetBird data volume:

```
sudo tar -czvf netbird_data_backup_$(date +%F-%H%M).tar.gz -C
/var/lib/docker/volumes/ubuntu_netbird_data _data
```

Verify that the backup was created:

```
ls -lh netbird_data_backup_*.tar.gz
```

Example output:

```
-rw-r--r-- 1 root root 33M May 11 00:03 netbird_data_backup_2026-05-11-
0003.tar.gz
```

Backup completed: Do not continue until you have confirmed that the backup file exists.

Step 6: Install SQLite

If SQLite is not installed, install it:

```
sudo apt update
sudo apt install sqlite3 -y
```

Step 7: Open the NetBird Database

Open the NetBird SQLite database:

```
sudo sqlite3 /var/lib/docker/volumes/ubuntu_netbird_data/_data/store.db
```

You should see something similar to:

```
SQLite version 3.45.1 2024-01-30 16:01:20
Enter ".help" for usage hints.
sqlite>
```

Step 8: Review the Database Tables

Inside SQLite, list the available tables:

```
.tables
```

Example output:

```
access_log_entries      network_addresses       proxy_access_tokens
account_onboardings     network_resources       records
accounts                network_routers         routes
domains                 networks                services
extra_settings          peers                   setup_keys
group_peers             personal_access_tokens  targets
groups                  policies                user_invites
installations           policy_rules            users
jobs                    posture_checks          zones
name_server_groups      proxies
```

The table we need is:

```
users
```

Step 9: Review the Users Table Schema

Enable headers and column mode:

```
.headers on
.mode column
```

Then view the schema for the `users` table:

```
.schema users
```

Example output:

```
CREATE TABLE `users` (
  `id` text,
  `account_id` text,
  `role` text,
  `is_service_user` numeric,
  `non_deletable` numeric,
  `service_user_name` text,
  `auto_groups` text,
  `blocked` numeric,
  `pending_approval` numeric,
  `last_login` datetime,
  `created_at` datetime,
  `issued` text DEFAULT "api",
  `integration_ref_id` integer,
  `integration_ref_integration_type` text,
  `name` text DEFAULT "",
  `email` text DEFAULT "",
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_accounts_users_g` FOREIGN KEY (`account_id`) REFERENCES
`accounts`(`id`)
);
```

Step 10: Identify the Correct User Account

Query the users table:

```
SELECT id, name, email, role, last_login FROM users;
```

In this environment, the `name` and `email` values appeared encrypted or masked, but the `role` and `last_login` values were visible.

Example output:

```
id                                                              role
last_login
--------------------------------------------------------------  -----
----------------------------------
CiQ5ZmU1M2NhNi03MDc1LTQ3NzQtOTM1Ni0wNjQ0NmEzZWYwYTYSBWxvY2Fs  admin

CkBkZmRhZTY2OWY2ZDYyYTA2NTE0YjY5ODNiZTI3NDNjMzRhY2MyOTExNTE5  owner

CgIxMBIeYXV0aGVudGlrLWQ3djhrOHBxbTQzMDAwOGNsNDQw              admin  2026-05-
10 01:53:30.291437858+00:00
```

The correct account was identified by the most recent `last_login` timestamp. That account was the working login, but its role was only `admin`.

Important: If there are duplicate users with the same name or email, do not update based only on email. Use `last_login` or the exact `id` to avoid promoting the wrong account.

Step 11: Promote the Recently Logged-In Admin to Owner

Because the working account was the only Admin account with a `last_login` value, this command promoted that account to Owner:

```
UPDATE users
SET role = 'owner'
WHERE role = 'admin'
```

```
  AND last_login IS NOT NULL;
```

Then verify the change:

```
SELECT id, role, last_login FROM users;
```

Expected result:

```
id                                                          role
last_login
----------------------------------------------------------  -----
-----------------------------------
CiQ5ZmU1M2NhNi03MDc1LTQ3NzQtOTM1Ni0wNjQ0NmEzZWYwYTYSBWxvY2Fs  admin

CkBkZmRhZTY2OWY2ZDYyYTA2NTE0YjY5ODNiZTI3NDNjMzRhY2MyOTExNTE5  owner

CgIxMBIeYXV0aGVudGlrLWQ3djhrOHBxbTQzMDAwOGNsNDQw             owner  2026-05-
10 01:53:30.291437858+00:00
```

The recently logged-in account is now an **Owner**.

Success: The working Admin account has been promoted to Owner.

Alternative: Promote by Exact User ID

If you want to be more specific, you can update a single user by ID:

```
UPDATE users
SET role = 'owner'
WHERE id = 'PASTE-THE-USER-ID-HERE';
```

This is the safest method when you have multiple recently active users or several duplicate accounts.

Step 12: Exit SQLite

Exit the SQLite shell:

```
.exit
```

Step 13: Start NetBird Again

Start the NetBird Docker Compose stack:

```
cd /home/ubuntu
docker compose up -d
```

Confirm the containers are running:

```
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}"
```

Expected containers:

```
NAMES               IMAGE                                STATUS
netbird-proxy       netbirdio/reverse-proxy:latest       Up
netbird-traefik     traefik:v3.6                         Up
netbird-crowdsec    crowdsecurity/crowdsec:v1.7.7        Up
netbird-dashboard   netbirdio/dashboard:latest           Up
netbird-server      netbirdio/netbird-server:latest      Up
```

Step 14: Confirm Access in the Dashboard

1. Open the NetBird dashboard.
2. Log out completely.
3. Log back in with the working account.
4. Go to the Users page.
5. Confirm that the working account now shows as **Owner**.

Step 15: Clean Up Duplicate Accounts

After confirming the working account has Owner access, you can clean up duplicate users from the NetBird dashboard.

Do not delete the old Owner immediately. First confirm that the current working login has Owner rights. Once confirmed, you can demote or remove the broken duplicate account.

Recommended final state:

```
Working Jediael Oliveira account     Owner
Other admin/service accounts          Admin
Broken duplicate account              Removed or demoted after verification
```

Rollback Plan

If something goes wrong, stop the stack:

```
cd /home/ubuntu
docker compose down
```

Restore the backup by extracting it back into the Docker volume location. Replace the backup filename with your actual backup file:

```
sudo tar -xzvf netbird_data_backup_2026-05-11-0003.tar.gz -C
/var/lib/docker/volumes/ubuntu_netbird_data
```

Then start NetBird again:

```
docker compose up -d
```

Summary

This issue was caused by a duplicate or inaccessible Owner account in a self-hosted NetBird deployment. Since the dashboard requires Owner permissions to transfer ownership, the role had to be corrected directly in the SQLite database.

The key fix was:

```
UPDATE users
SET role = 'owner'
WHERE role = 'admin'
  AND last_login IS NOT NULL;
```

After restarting NetBird, the working account was able to log in with Owner permissions.