Supabase Storage RLS Policy Fix Guide

🔍 Problem Explanation

You're getting the error: **"Failed to upload file: new row violates row-level security policy"**

What is Row-Level Security (RLS)?

Row-Level Security (RLS) is a security feature in Supabase that controls who can access and modify data at the row level. When enabled (which it is by default for Supabase Storage), **every operation requires an explicit policy** that grants permission.

Why Did This Error Occur?

Your `customer-logos` bucket is set to **Public**, which means:
- ✅ Anyone can **read/view** files (SELECT)
- ❌ But **no one can upload** files (INSERT) because there's no policy allowing it

Your application code uses the Supabase **anon key** (from `NEXT_PUBLIC_SUPABASE_ANON_KEY`) to authenticate storage operations. Without RLS policies granting the anon key permission to INSERT, the upload fails.

🛠️ The Solution

We need to create RLS policies that allow:
1. **Authenticated requests** (using your anon key) to **INSERT** (upload) files
2. **Everyone** to **SELECT** (read) files (for public display)
3. **Authenticated requests** to **UPDATE** and **DELETE** files

---

📋 Step-by-Step Fix Instructions

Step 1: Access Supabase SQL Editor

1. Go to your Supabase Dashboard: https://supabase.com/dashboard
2. Select your project
3. Navigate to **SQL Editor** in the left sidebar
4. Click **"+ New query"**

Step 2: Run the RLS Policy Creation Script

Copy and paste the **entire SQL script** below into the SQL Editor:

```sql
-- ==============================================================================
-- Supabase Storage RLS Policies for customer-logos Bucket
-- ==============================================================================
-- This script creates policies to allow authenticated users to upload, update,
-- and delete logos, while allowing public read access for displaying logos.
-- ==============================================================================

-- 1️⃣ POLICY: Allow authenticated users to INSERT (upload) files
-- This allows your application (using the anon key) to upload logo files
CREATE POLICY "Allow authenticated users to upload logos"
ON storage.objects
FOR INSERT
TO authenticated, anon
WITH CHECK (bucket_id = 'customer-logos');

-- 2️⃣ POLICY: Allow public SELECT (read) access to all files
-- This allows logos to be displayed publicly on your website
CREATE POLICY "Allow public read access to logos"
ON storage.objects
FOR SELECT
TO public
USING (bucket_id = 'customer-logos');

-- 3️⃣ POLICY: Allow authenticated users to UPDATE files
-- This allows replacing existing logo files
CREATE POLICY "Allow authenticated users to update logos"
ON storage.objects
FOR UPDATE
TO authenticated, anon
USING (bucket_id = 'customer-logos')
WITH CHECK (bucket_id = 'customer-logos');

-- 4️⃣ POLICY: Allow authenticated users to DELETE files
-- This allows removing old logo files when uploading a new one
CREATE POLICY "Allow authenticated users to delete logos"
ON storage.objects
FOR DELETE
TO authenticated, anon
USING (bucket_id = 'customer-logos');

-- ==============================================================================
-- VERIFICATION: List all policies for the storage.objects table
-- ==============================================================================
-- Run this query to verify the policies were created successfully
SELECT
    schemaname,
    tablename,
    policyname,
    permissive,
    roles,
    cmd,
    qual,
    with_check
FROM pg_policies
WHERE tablename = 'objects'
  AND schemaname = 'storage'
  AND policyname LIKE '%logos%'
ORDER BY policyname;
```

Step 3: Execute the Script

1. Click the **"Run"** button (or press `Ctrl+Enter` / `Cmd+Enter`)
2. You should see a success message confirming the policies were created
3. The verification query at the end will show you all the policies that were created

Step 4: Test the Logo Upload

1. Go back to your application
2. Navigate to the **Super Admin Dashboard → Customers** tab
3. Try uploading a customer logo again
4. ✅ The upload should now work successfully!

🧪 Verification Steps

Verify Policies Were Created

Run this query in the SQL Editor to check your policies:

```
SELECT
    policyname,
    cmd as operation,
    roles,
    CASE
        WHEN qual IS NOT NULL THEN 'Has USING clause'
        ELSE 'No USING clause'
    END as using_check,
    CASE
        WHEN with_check IS NOT NULL THEN 'Has WITH CHECK clause'
        ELSE 'No WITH CHECK clause'
    END as with_check_status
FROM pg_policies
WHERE tablename = 'objects'
  AND schemaname = 'storage'
  AND policyname LIKE '%logos%'
ORDER BY cmd, policyname;
```

Expected output should show **4 policies**:
- 1 for INSERT
- 1 for SELECT
- 1 for UPDATE
- 1 for DELETE

Test Upload Functionality

1. **Test Upload**: Upload a logo through your admin dashboard
   - Should succeed with no errors
   - Check the Supabase Storage browser to see the file

2. **Test Display**: View a customer page where the logo is displayed
   - Logo should load correctly
   - Check browser console for no 403 errors

3. **Test Update**: Upload a new logo for the same customer
   - Old logo should be replaced
   - New logo should display

4. **Test Delete**: Delete a customer logo
   - File should be removed from storage
   - No errors should occur

🔐 Security Explanation

Why These Policies Are Safe

1. **Bucket Isolation**: All policies are scoped to `bucket_id = 'customer-logos'`
   - Policies only affect files in this specific bucket
   - Other buckets remain protected

2. **Public Read is Safe**: The SELECT policy allows public access
   - This is intentional for displaying logos on your website
   - Only affects this bucket, not your entire database

3. **Controlled Write Access**: INSERT/UPDATE/DELETE require authentication
   - Only requests with valid anon key can modify files
   - Your anon key is secured in environment variables

4. **No File System Access**: Storage is isolated
   - These policies don't grant access to database tables
   - They only control access to files in Supabase Storage

Additional Security Considerations

Your application code already includes:

✅ **File Type Validation**: Only allows image files (.jpg, .jpeg, .png, .gif, .bmp)
✅ **File Size Limits**: Maximum 5MB per file
✅ **Authentication Check**: Only SUPER_ADMIN role can upload
✅ **File Naming Convention**: Files are named `customer-{id}.{ext}`

These application-level validations work **in addition to** the RLS policies.

🐛 Troubleshooting

Issue: Policies Already Exist

Error: `ERROR: policy "Allow authenticated users to upload logos" already exists`

Solution: Policies were already created. You can either:

1. **Skip this step** - Policies are already in place
2. **Delete and recreate** - Run this first:

```sql
-- Delete existing policies
DROP POLICY IF EXISTS "Allow authenticated users to upload logos" ON storage.objects;
DROP POLICY IF EXISTS "Allow public read access to logos" ON storage.objects;
DROP POLICY IF EXISTS "Allow authenticated users to update logos" ON storage.objects;
DROP POLICY IF EXISTS "Allow authenticated users to delete logos" ON storage.objects;
```

Then run the creation script again.

Issue: Upload Still Fails

Error: Still getting RLS errors after creating policies

Possible Causes:

1. **Environment Variables Not Set**: Verify `NEXT_PUBLIC_SUPABASE_ANON_KEY` is set

```bash
# Check your .env.local file
cat .env.local | grep SUPABASE
```

1. **Bucket Name Mismatch**: Verify the bucket name is exactly `customer-logos`

```sql
-- Check bucket name in Supabase
SELECT name, public FROM storage.buckets;
```

1. **Anon Key Invalid**: The anon key might be incorrect
   - Go to Supabase Dashboard → Settings → API
   - Copy the `anon` `public` key
   - Update your `.env.local` file

Issue: Logos Don't Display

Symptom: Upload succeeds but logos don't show on the website

Solution: Check bucket is public

```sql
-- Make sure bucket is public
UPDATE storage.buckets
SET public = true
WHERE name = 'customer-logos';
```

Issue: 403 Forbidden When Viewing Logos

Symptom: Logos uploaded but can't be viewed (403 error in browser console)

Solution: Ensure SELECT policy exists

```sql
-- Verify SELECT policy
SELECT * FROM pg_policies
WHERE tablename = 'objects'
  AND schemaname = 'storage'
  AND cmd = 'SELECT'
  AND policyname LIKE '%logos%';
```

If no results, create the SELECT policy again:

```sql
CREATE POLICY "Allow public read access to logos"
ON storage.objects
FOR SELECT
TO public
USING (bucket_id = 'customer-logos');
```

📚 Additional Resources

Supabase Documentation

- Storage RLS Policies (https://supabase.com/docs/guides/storage/security/access-control)
- Understanding RLS (https://supabase.com/docs/guides/auth/row-level-security)
- Storage Quickstart (https://supabase.com/docs/guides/storage)

PostgreSQL Documentation

- Row Security Policies (https://www.postgresql.org/docs/current/sql-createpolicy.html)

✅ Summary Checklist

Before considering this issue resolved, verify:

- [ ] All 4 RLS policies created successfully (INSERT, SELECT, UPDATE, DELETE)
- [ ] Policies verified with the verification query
- [ ] Logo upload tested and working
- [ ] Logo display tested and working
- [ ] Logo update tested and working
- [ ] No RLS errors in application logs
- [ ] No 403 errors in browser console

💡 What You Learned

1. **Row-Level Security (RLS)** controls data access at the row level in Supabase
2. **Public buckets** only allow READ access by default - write operations need explicit policies
3. **Storage objects** are stored in the `storage.objects` table with RLS enabled
4. **Policies** must be created for each operation type: SELECT, INSERT, UPDATE, DELETE
5. **Authentication roles** include `authenticated`, `anon`, and `public`
6. Your app uses the **anon key** for storage operations, which needs explicit permission

---

🎉 You're Done!

Your Supabase Storage is now properly configured with RLS policies. Logo uploads should work seamlessly. If you encounter any issues, refer to the Troubleshooting section above.

Need help? Check the Supabase Discord community or their documentation.

Last Updated: 2025-10-21
Repository: /home/ubuntu/github_repos/roblox-tool