Whop Login Blank Page Fix

Issue Report

Symptom: Clicking “Log in with Whop” redirects to Whop OAuth, then returns a blank/white page instead of the dashboard.

🔍 Root Cause Analysis

1️⃣ Route Identification

Found: The “Log in with Whop” button in `/app/auth/login/login-client.tsx` (lines 335-354) calls:

```
signIn('whop', { callbackUrl });
```

This triggers the following OAuth flow:

```
1. User clicks button
2. NextAuth redirects to: /api/auth/signin/whop
3. NextAuth redirects to: https://data.whop.com/api/v3/oauth/authorize
4. User authorizes on Whop
5. Whop redirects back to: https://catchbarrels.app/api/auth/callback/whop
6. NextAuth processes callback
7. PROBLEM: NextAuth redirect callback returns blank page
```

2️⃣ Issues Identified in `/lib/auth-options.ts`

Problem 1: Incomplete Whop Provider Configuration
- The Whop OAuth provider (lines 108-132) was using `as any` type casting
- Missing `wellKnown` endpoint for OpenID Connect discovery
- Token and userinfo endpoints were strings instead of objects
- No debug logging to trace OAuth flow

Problem 2: Missing NextAuth Configuration
- No `newUser` page configured for first-time OAuth sign-ins
- No `debug` flag enabled to help diagnose issues
- Redirect callback had no logging

Problem 3: Middleware Verification
- Confirmed `/api/auth/*` routes are public (line 22 of middleware.ts) ✅
- No blocking issues found in middleware

✅ Fixes Applied

Fix 1: Updated Whop OAuth Provider Configuration

File: `/lib/auth-options.ts`

Changes:

```
// BEFORE
{
  id: 'whop',
  name: 'Whop',
  type: 'oauth',
  clientId: process.env.WHOP_CLIENT_ID,
  clientSecret: process.env.WHOP_CLIENT_SECRET,
  authorization: {
    url: 'https://data.whop.com/api/v3/oauth/authorize',
    params: {
      scope: 'openid profile email',
      response_type: 'code',
    },
  },
  token: 'https://data.whop.com/api/v3/oauth/token',
  userinfo: 'https://api.whop.com/api/v2/me',
  profile(profile: any) {
    return {
      id: profile.id,
      name: profile.name || profile.username,
      email: profile.email,
      username: profile.username,
      whopUserId: profile.id,
    };
  },
} as any,

// AFTER
{
  id: 'whop',
  name: 'Whop',
  type: 'oauth',
  clientId: process.env.WHOP_CLIENT_ID,
  clientSecret: process.env.WHOP_CLIENT_SECRET,
  wellKnown: 'https://data.whop.com/api/v3/oauth/.well-known/openid-configuration',
  authorization: {
    url: 'https://data.whop.com/api/v3/oauth/authorize',
    params: {
      scope: 'openid profile email',
      response_type: 'code',
    },
  },
  token: {
    url: 'https://data.whop.com/api/v3/oauth/token',
  },
  userinfo: {
    url: 'https://api.whop.com/api/v2/me',
  },
  profile(profile: any) {
    console.log('[Whop OAuth] Profile received:', profile);
    return {
      id: profile.id,
      name: profile.name || profile.username,
      email: profile.email,
      username: profile.username,
      whopUserId: profile.id,
    };
  },
} as OAuthConfig<any>,
```

Key improvements:
- Added `wellKnown` endpoint for OpenID Connect auto-configuration
- Changed `token` and `userinfo` from strings to objects (NextAuth requirement)
- Added `console.log` in profile mapping for debugging
- Properly typed as `OAuthConfig<any>`

Fix 2: Added Debug Configuration

File: `/lib/auth-options.ts`

```
pages: {
  signIn: '/auth/login',
  error: '/auth/login',
  newUser: '/dashboard', // NEW: Redirect new users here after first OAuth sign-in
},
debug: process.env.NODE_ENV === 'development', // NEW: Enable debug logging
```

Fix 3: Enhanced Redirect Callback with Logging

File: `/lib/auth-options.ts`

Changes:
- Added comprehensive `console.log` statements to trace redirect flow
- Log every branch of redirect logic
- Log errors with full context

```
async redirect({ url, baseUrl }) {
  try {
    console.log('[NextAuth Redirect] url:', url, 'baseUrl:', baseUrl);
    // ... rest of redirect logic with logging at each step
  } catch (error) {
    console.error('[NextAuth Redirect] Error:', error);
    return `${baseUrl}/dashboard`;
  }
}
```

Fix 4: Enhanced JWT Callback Logging

File: `/lib/auth-options.ts`

```
if (account?.provider === 'whop' && token.whopUserId) {
  console.log('[Whop OAuth] Processing Whop login for user:', token.whopUserId);
  try {
    // ... existing logic with logging at each step
    console.log('[Whop OAuth] New user created:', dbUser.id);
  } catch (error) {
    console.error('Error syncing Whop membership:', error);
  }
}
```

🧪 Testing Instructions

Manual Test Steps

1. **Clear browser cache and cookies** for catchbarrels.app

2. **Navigate to:** https://catchbarrels.app/auth/login

3. **Click "Sign in with Whop"**

4. **Expected behavior:**
   - Redirects to Whop OAuth page
   - After authorization, redirects back to app
   - Shows loading animation briefly
   - Lands on `/dashboard` with user logged in
   - No blank page

5. **Check server logs for:**
   ```
   [Whop OAuth] Profile received: {...}
      [Whop OAuth] Processing Whop login for user: ...
      [Whop OAuth] Existing user found: ... (or "New user created")
      [NextAuth Redirect] url: ... baseUrl: ...
      [NextAuth Redirect] Relative URL detected: /dashboard
   ```

Browser Console Check

1. Open DevTools Console (F12)
2. Look for any errors during OAuth flow
3. Check Network tab for:
   - `/api/auth/signin/whop` → 302 redirect to Whop
   - Whop OAuth authorize → 302 redirect back
   - `/api/auth/callback/whop` → 302 redirect to dashboard
   - `/dashboard` → 200 OK (HTML rendered)

Common Issues to Check

If blank page persists:

1. **Check Whop Dashboard OAuth Settings:**
   - Redirect URI must be: `https://catchbarrels.app/api/auth/callback/whop`
   - Client ID matches `.env` : `WHOP_CLIENT_ID`
   - Client Secret matches `.env` : `WHOP_CLIENT_SECRET`

2. **Check Environment Variables:**
   ```
   bash
      grep -E "WHOP_|NEXTAUTH_URL" .env
   ```

Should show:
```
NEXTAUTH_URL=https://catchbarrels.app
   WHOP_CLIENT_ID=app_...
   WHOP_CLIENT_SECRET=apik_...
```

1. **Check Server Logs:**
   ```
   bash
   ```

```
   # Look for NextAuth logs
   pm2 logs catchbarrels --lines 100
```

2. **Verify NextAuth Callback URL is Not Blocked:**
   - `/api/auth/*` should be in middleware.ts public paths ✅ (confirmed)

📊 Expected Logs After Fix

Successful OAuth Flow Logs:

```
[Whop OAuth] Profile received: {
  id: 'user_...',
  name: 'John Doe',
  email: 'john@example.com',
  username: 'johndoe'
}
[Whop OAuth] Processing Whop login for user: user_...
[Whop OAuth] Existing user found: clxxxx... (or "New user created")
[NextAuth Redirect] url: /dashboard baseUrl: https://catchbarrels.app
[NextAuth Redirect] Relative URL detected: /dashboard
```

🚀 Deployment Checklist

- [x] Updated `/lib/auth-options.ts` with proper Whop OAuth config
- [x] Added debug logging to all OAuth callbacks
- [x] Added `newUser` redirect page
- [x] TypeScript compiles without errors
- [ ] Test Whop login flow on production
- [ ] Verify user is created in database
- [ ] Verify membership sync from Whop works
- [ ] Verify redirect to dashboard succeeds
- [ ] Check server logs for any errors

🔧 Rollback Plan

If issues persist, the Whop OAuth provider can be temporarily disabled by commenting out lines 108-139 in `/lib/auth-options.ts`. Users can still log in with credentials.

📖 Additional Resources

- NextAuth OAuth Provider Docs (https://next-auth.js.org/configuration/providers/oauth)
- Whop OAuth Documentation (https://docs.whop.com/)
- OpenID Connect Discovery (https://openid.net/specs/openid-connect-discovery-1_0.html)

Summary

Root Cause: Incomplete Whop OAuth provider configuration causing NextAuth to fail during callback processing.

Fix:
1. Properly configured Whop OAuth provider with `wellKnown` endpoint
2. Fixed token/userinfo endpoint format (object instead of string)
3. Added comprehensive debug logging
4. Added `newUser` redirect configuration

Status: ✅ **Ready for Testing**

Next Steps: Deploy to production and test Whop login flow with real users.