parse-community / **parse-server** Public

Code Issues 370 Pull requests 102 Actions Projects Wiki Se

Commit `770be86`

mtrezza authored last week · ✓ 21 / 22 · Verified

```
fix: Auth data exposed via verify password endpoint ([GHSA-wp76-gg32-8258](GHSA-wp76-gg32-8258)) (#10323)
```

`alpha` (#10323) · 9.8.0-alpha.4 ••• 9.7.0-alpha.7

1 parent `f537e67` commit `770be86`

2 files changed +93 -2 lines changed

Top

Filter files…

- spec
  - vulnerabilities.spec.js
- src/Routers
  - UsersRouter.js

2 files changed +93 -2 lines changed

Search within code

spec/vulnerabilities.spec.js

```
@@ -4600,4 +4600,95 @@ describe('(GHSA-p2w6-rmh7-w8q3) SQL Injection via aggregate and distinct field n
4600 4600        expect(meResponse.data.authData?.mfa).toEqual({ status: 'enabled' });
4601 4601      });
4602 4602    });
     4603 +
     4604 +  describe('(GHSA-wp76-gg32-8258) /verifyPassword leaks raw authData via missing afterFind', () => {
     4605 +    const headers = {
     4606 +      'X-Parse-Application-Id': 'test',
```

```
4607 +       'X-Parse-REST-API-Key': 'rest',
4608 +       'Content-Type': 'application/json',
4609 +     };
4610 +
4611 +     it('does not leak raw MFA authData via /verifyPassword', async () => {
4612 +       await reconfigureServer({
4613 +         auth: {
4614 +           mfa: {
4615 +             enabled: true,
4616 +             options: ['TOTP'],
4617 +             algorithm: 'SHA1',
4618 +             digits: 6,
4619 +             period: 30,
4620 +           },
4621 +         },
4622 +         verifyUserEmails: false,
4623 +       });
4624 +       const user = await Parse.User.signUp('username', 'password');
4625 +       const sessionToken = user.getSessionToken();
4626 +       const OTPAuth = require('otpauth');
4627 +       const secret = new OTPAuth.Secret();
4628 +       const totp = new OTPAuth.TOTP({
4629 +         algorithm: 'SHA1',
4630 +         digits: 6,
4631 +         period: 30,
4632 +         secret,
4633 +       });
4634 +       const token = totp.generate();
4635 +       // Enable MFA
4636 +       await user.save(
4637 +         { authData: { mfa: { secret: secret.base32, token } } },
4638 +         { sessionToken }
4639 +       );
4640 +       // Verify MFA data is stored (master key)
4641 +       await user.fetch({ useMasterKey: true });
4642 +       expect(user.get('authData').mfa.secret).toBe(secret.base32);
4643 +       expect(user.get('authData').mfa.recovery).toBeDefined();
4644 +       // POST /verifyPassword should NOT include raw MFA data
4645 +       const response = await request({
4646 +         headers,
```

```
4647 +         method: 'POST',
4648 +         url: 'http://localhost:8378/1/verifyPassword',
4649 +         body: JSON.stringify({ username: 'username', password: 'password' }),
4650 +       });
4651 +       expect(response.data.authData?.mfa?.secret).toBeUndefined();
4652 +       expect(response.data.authData?.mfa?.recovery).toBeUndefined();
4653 +       expect(response.data.authData?.mfa).toEqual({ status: 'enabled' });
4654 +     });
4655 +
4656 +     it('does not leak raw MFA authData via GET /verifyPassword', async () =>
       {
4657 +       await reconfigureServer({
4658 +         auth: {
4659 +           mfa: {
4660 +             enabled: true,
4661 +             options: ['TOTP'],
4662 +             algorithm: 'SHA1',
4663 +             digits: 6,
4664 +             period: 30,
4665 +           },
4666 +         },
4667 +         verifyUserEmails: false,
4668 +       });
4669 +       const user = await Parse.User.signUp('username', 'password');
4670 +       const sessionToken = user.getSessionToken();
4671 +       const OTPAuth = require('otpauth');
4672 +       const secret = new OTPAuth.Secret();
4673 +       const totp = new OTPAuth.TOTP({
4674 +         algorithm: 'SHA1',
4675 +         digits: 6,
4676 +         period: 30,
4677 +         secret,
4678 +       });
4679 +       await user.save(
4680 +         { authData: { mfa: { secret: secret.base32, token: totp.generate() }
       } },
4681 +         { sessionToken }
4682 +       );
4683 +       // GET /verifyPassword should NOT include raw MFA data
4684 +       const response = await request({
```

```
4685 +         headers,
4686 +         method: 'GET',
4687 +         url: `http://localhost:8378/1/verifyPassword?
       username=username&password=password`,
4688 +       });
4689 +       expect(response.data.authData?.mfa?.secret).toBeUndefined();
4690 +       expect(response.data.authData?.mfa?.recovery).toBeUndefined();
4691 +       expect(response.data.authData?.mfa).toEqual({ status: 'enabled' });
4692 +     });
4693 +   });
4603 4694   });
```

src/Routers/UsersRouter.js

```
@@ -422,10 +422,10 @@ export class UsersRouter extends ClassesRouter {
422 422
423 423     handleVerifyPassword(req) {
424 424       return this._authenticateUserFromRequest(req)
425     -       .then(user => {
    425 +       .then(async user => {
426 426           // Remove hidden properties.
427 427           UsersRouter.removeHiddenProperties(user);
428     -
    428 +         await req.config.authDataManager.runAfterFind(req, user.authData);
429 429           return { response: user };
430 430         })
431 431         .catch(error => {
```

Comments 0

Please sign in to comment.