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

Code Issues 370 Pull requests 102 Actions Projects Wiki Se

Commit a1d4e7b

mtrezza authored last week · ✕ 21 / 26 · Verified

```
fix: Auth data exposed via verify password endpoint ([GHSA-wp76-gg32-8258](GHSA-wp76-gg32-8258)) (#10324)
```

release-8.x.x (#10324) +
snyk-fix-1535e6b506c3eeecb036979980bc9706 (#10385, #10324)

8.6.73 ···
8.6.63

1 parent 4ff8b79 commit a1d4e7b

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

```
@@ -4142,3 +4142,94 @@ describe('(GHSA-37mj-c2wf-cx96) /users/me leaks raw authData via master context'
4142 4142      expect(meResponse.data.authData?.mfa).toEqual({ status: 'enabled' });
4143 4143    });
4144 4144  });
     4145 +
     4146 + describe('(GHSA-wp76-gg32-8258) /verifyPassword leaks raw authData via missing afterFind', () => {
     4147 +   const headers = {
```

```
4148 +      'X-Parse-Application-Id': 'test',
4149 +      'X-Parse-REST-API-Key': 'rest',
4150 +      'Content-Type': 'application/json',
4151 +    };
4152 +
4153 +    it('does not leak raw MFA authData via /verifyPassword', async () => {
4154 +      await reconfigureServer({
4155 +        auth: {
4156 +          mfa: {
4157 +            enabled: true,
4158 +            options: ['TOTP'],
4159 +            algorithm: 'SHA1',
4160 +            digits: 6,
4161 +            period: 30,
4162 +          },
4163 +        },
4164 +        verifyUserEmails: false,
4165 +      });
4166 +      const user = await Parse.User.signUp('username', 'password');
4167 +      const sessionToken = user.getSessionToken();
4168 +      const OTPAuth = require('otpauth');
4169 +      const secret = new OTPAuth.Secret();
4170 +      const totp = new OTPAuth.TOTP({
4171 +        algorithm: 'SHA1',
4172 +        digits: 6,
4173 +        period: 30,
4174 +        secret,
4175 +      });
4176 +      const token = totp.generate();
4177 +      // Enable MFA
4178 +      await user.save(
4179 +        { authData: { mfa: { secret: secret.base32, token } } },
4180 +        { sessionToken }
4181 +      );
4182 +      // Verify MFA data is stored (master key)
4183 +      await user.fetch({ useMasterKey: true });
4184 +      expect(user.get('authData').mfa.secret).toBe(secret.base32);
4185 +      expect(user.get('authData').mfa.recovery).toBeDefined();
4186 +      // POST /verifyPassword should NOT include raw MFA data
4187 +      const response = await request({
```

```
4188 +       headers,
4189 +       method: 'POST',
4190 +       url: 'http://localhost:8378/1/verifyPassword',
4191 +       body: JSON.stringify({ username: 'username', password: 'password' }),
4192 +     });
4193 +     expect(response.data.authData?.mfa?.secret).toBeUndefined();
4194 +     expect(response.data.authData?.mfa?.recovery).toBeUndefined();
4195 +     expect(response.data.authData?.mfa).toEqual({ status: 'enabled' });
4196 +   });
4197 +
4198 +   it('does not leak raw MFA authData via GET /verifyPassword', async () => {
4199 +     await reconfigureServer({
4200 +       auth: {
4201 +         mfa: {
4202 +           enabled: true,
4203 +           options: ['TOTP'],
4204 +           algorithm: 'SHA1',
4205 +           digits: 6,
4206 +           period: 30,
4207 +         },
4208 +       },
4209 +       verifyUserEmails: false,
4210 +     });
4211 +     const user = await Parse.User.signUp('username', 'password');
4212 +     const sessionToken = user.getSessionToken();
4213 +     const OTPAuth = require('otpauth');
4214 +     const secret = new OTPAuth.Secret();
4215 +     const totp = new OTPAuth.TOTP({
4216 +       algorithm: 'SHA1',
4217 +       digits: 6,
4218 +       period: 30,
4219 +       secret,
4220 +     });
4221 +     await user.save(
4222 +       { authData: { mfa: { secret: secret.base32, token: totp.generate() } } }
     },
4223 +       { sessionToken }
4224 +     );
4225 +     // GET /verifyPassword should NOT include raw MFA data
4226 +     const response = await request({
```

```
4227 +       headers,
4228 +       method: 'GET',
4229 +       url: `http://localhost:8378/1/verifyPassword?
       username=username&password=password`,
4230 +     });
4231 +     expect(response.data.authData?.mfa?.secret).toBeUndefined();
4232 +     expect(response.data.authData?.mfa?.recovery).toBeUndefined();
4233 +     expect(response.data.authData?.mfa).toEqual({ status: 'enabled' });
4234 +   });
4235 + });
```

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.