ChurchCRM / **CRM** Public

Code | Issues 137 | Pull requests 23 | Discussions | Actions | Projects

Commit 214694e

DawoudIO authored last week · 25 / 26 · Verified

```
security: fix SQLi in FinancialService + harden API login (#8607)
```

master (#8607) · 7.2.1 ••• 7.2.0-rc2

1 parent 8404e9d commit 214694e

4 files changed +120 -15 lines changed

Top

Filter files…

- cypress/e2e/api/public
  - public.user.spec.js
- src
  - ChurchCRM
    - Service
      - FinancialService.php
    - Slim
      - SlimUtils.php
  - api/routes/public
    - public-user.php

4 files changed +120 -15 lines changed

Search within code

cypress/e2e/api/public/public.user.spec.js

```
     ••• @@ -1,7 +1,7 @@
1    1   /// <reference types="cypress" />
2    2
```

```
3   3     describe("API Public User", () => {
4         -     it("Login", () => {
      4   +     it("Login with valid credentials returns apiKey", () => {
5   5             const user = {
6   6                 userName: "admin",
7   7                 password: "changeme",
    ↕         @@ -19,6 +19,43 @@ describe("API Public User", () => {
19  19            });
20  20        });
21  21
      22  +     it("Login with non-existent user returns 401 (not 404)", () => {
      23  +         cy.apiRequest({
      24  +             method: "POST",
      25  +             url: "/api/public/user/login",
      26  +             headers: { "content-type": "application/json" },
      27  +             body: { userName: "nonexistent_user_xyz", password: "anything" },
      28  +             failOnStatusCode: false,
      29  +         }).then((resp) => {
      30  +             // Should return 401 (same as wrong password) to prevent username
          enumeration
      31  +             expect(resp.status).to.eq(401);
      32  +         });
      33  +     });
      34  +
      35  +     it("Login with wrong password returns 401", () => {
      36  +         cy.apiRequest({
      37  +             method: "POST",
      38  +             url: "/api/public/user/login",
      39  +             headers: { "content-type": "application/json" },
      40  +             body: { userName: "admin", password: "wrong_password" },
      41  +             failOnStatusCode: false,
      42  +         }).then((resp) => {
      43  +             expect(resp.status).to.eq(401);
      44  +         });
      45  +     });
      46  +
      47  +     it("Login with empty userName returns 401", () => {
      48  +         cy.apiRequest({
      49  +             method: "POST",
      50  +             url: "/api/public/user/login",
```

```
      51  +             headers: { "content-type": "application/json" },
      52  +             body: { userName: "", password: "anything" },
      53  +             failOnStatusCode: false,
      54  +         }).then((resp) => {
      55  +             expect(resp.status).to.eq(401);
      56  +         });
      57  +     });
      58  +
22    59          describe("Password Reset", () => {
23    60              it("Successful password reset request with valid user", () => {
24    61                  cy.apiRequest({
```

src/ChurchCRM/Service/FinancialService.php

```
      @@ -84,17 +84,15 @@ public function getMemberByScanString(string
      $tScanString): array
84    84              if (!$routeAndAccount) {
85    85                  throw new \Exception('error in locating family');
86    86              }
87          -         $sSQL = 'SELECT fam_ID, fam_Name FROM family_fam WHERE fam_scanCheck="'
      . $routeAndAccount . '"';
88          -         $rsFam = FunctionsUtils::runQuery($sSQL);
89          -         $row = mysqli_fetch_array($rsFam);
      87  +         $family = FamilyQuery::create()->findOneByScanCheck($routeAndAccount);
90    88              $iCheckNo = $micrObj->findCheckNo($tScanString);
91    89
92    90              return [
93    91                  'ScanString'      => $tScanString,
94    92                  'RouteAndAccount' => $routeAndAccount,
95    93                  'CheckNumber'     => $iCheckNo,
96          -             'fam_ID'          => $row['fam_ID'],
97          -             'fam_Name'        => $row['fam_Name'],
      94  +             'fam_ID'          => $family?->getId(),
      95  +             'fam_Name'        => $family?->getName(),
98    96              ];
99    97          }
100   98
```

src/ChurchCRM/Slim/SlimUtils.php

```
@@ -7,8 +7,11 @@
7 7  use Exception;
8 8  use Psr\Http\Message\ResponseInterface as Response;
9 9  use Psr\Http\Message\ServerRequestInterface as Request;
  10 + use Slim\Exception\HttpBadRequestException;
  11 + use Slim\Exception\HttpForbiddenException;
10 12  use Slim\Exception\HttpMethodNotAllowedException;
11 13  use Slim\Exception\HttpNotFoundException;
  14 + use Slim\Exception\HttpUnauthorizedException;
12 15  use Slim\Interfaces\RouteInterface;
13 16  use Slim\Psr7\Response as Psr7Response;
14 17  use Slim\Routing\RouteContext;
@@ -173,7 +176,7 @@ public static function registerDefaultJsonErrorHandler($errorMiddleware)
173 176                  'user_agent' => $request->getHeaderLine('User-Agent')
174 177              ];
175 178              $logger->error('Uncaught exception: ' . $exception->getMessage(), $requestContext);
176     -
    179 +
177 180              $response = new Psr7Response();
178 181
179 182              // Determine appropriate HTTP status code based on exception type
@@ -182,6 +185,12 @@ public static function registerDefaultJsonErrorHandler($errorMiddleware)
182 185                  $statusCode = 404;
183 186              } elseif ($exception instanceof HttpMethodNotAllowedException) {
184 187                  $statusCode = 405;
    188 +            } elseif ($exception instanceof HttpUnauthorizedException) {
    189 +                $statusCode = 401;
    190 +            } elseif ($exception instanceof HttpForbiddenException) {
    191 +                $statusCode = 403;
    192 +            } elseif ($exception instanceof HttpBadRequestException) {
    193 +                $statusCode = 400;
185 194              }
186 195
187 196              // Sanitize error message to prevent credential disclosure
```

src/api/routes/public/public-user.php

```
@@ -1,5 +1,6 @@
<?php

+ use ChurchCRM\Emails\users\LockedEmail;
use ChurchCRM\Emails\users\ResetPasswordTokenEmail;
use ChurchCRM\model\ChurchCRM\Token;
use ChurchCRM\model\ChurchCRM\UserQuery;
@@ -8,7 +9,6 @@
use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;
use Slim\Exception\HttpBadRequestException;
- use Slim\Exception\HttpNotFoundException;
use Slim\Exception\HttpUnauthorizedException;
use Slim\Routing\RouteCollectorProxy;

@@ -30,7 +30,8 @@
 *         @OA\JsonContent(
 *             required={"userName","password"},
 *             @OA\Property(property="userName", type="string", example="admin"),
- *             @OA\Property(property="password", type="string", format="password", example="secret")
+ *             @OA\Property(property="password", type="string", format="password", example="secret"),
+ *             @OA\Property(property="otp", type="string", description="One-time password or recovery code for 2FA (required when 202 is returned)", example="123456")
 *         )
 *     ),
 *     @OA\Response(
@@ -41,26 +42,86 @@
 *         )
 *     ),
 *     @OA\Response(response=401, description="Invalid username or password"),
- *     @OA\Response(response=404, description="User not found")
+ *     @OA\Response(
+ *         response=202,
+ *         description="Password valid but 2FA verification required",
```

```
+  *         @OA\JsonContent(
+  *             @OA\Property(property="requiresOTP", type="boolean", example=true)
+  *         )
+  *     )
    * )
    */
   function userLogin(Request $request, Response $response, array $args): Response
   {
+      $logger = LoggerUtils::getAppLogger();
       $body = json_decode($request->getBody(), true, 512, JSON_THROW_ON_ERROR);
+
+      // Use a generic error message to prevent username enumeration
+      $genericError = gettext('Invalid login or password');
+
       if (empty($body['userName'])) {
-          throw new HttpNotFoundException($request);
+          throw new HttpUnauthorizedException($request, $genericError);
       }

       $user = UserQuery::create()->findOneByUserName($body['userName']);
-      if (empty($user)) {
-          throw new HttpNotFoundException($request);
+      if ($user === null) {
+          // Return same error as wrong password to prevent username enumeration
+          $logger->warning('API login attempt for non-existent user', ['username' => $body['userName']]);
+          throw new HttpUnauthorizedException($request, $genericError);
+      }
+
+      // Check account lockout before attempting password validation
+      // Use same generic error to prevent username enumeration via locked-account message
+      if ($user->isLocked()) {
+          $logger->warning('API login attempt for locked account', ['username' => $user->getUserName()]);
+          throw new HttpUnauthorizedException($request, $genericError);
       }

-      $password = $body['password'];
```

```
                80  +     $password = $body['password'] ?? '';
60              81        if (!$user->isPasswordValid($password)) {
61                  -         throw new HttpUnauthorizedException($request, gettext('Invalid
                        User/Password'));
                82  +         // Increment failed login counter
                83  +         $user->setFailedLogins($user->getFailedLogins() + 1);
                84  +         $user->save();
                85  +
                86  +         // Send locked email if account just became locked
                87  +         if ($user->isLocked() && !empty($user->getEmail())) {
                88  +             $logger->warning('API login: account locked after too many
                        failures', ['username' => $user->getUserName()]);
                89  +             $lockedEmail = new LockedEmail($user);
                90  +             $lockedEmail->send();
                91  +         }
                92  +
                93  +         $logger->warning('API login: invalid password', ['username' => $user-
                        >getUserName()]);
                94  +         throw new HttpUnauthorizedException($request, $genericError);
                95  +     }
                96  +
                97  +     // Check 2FA enrollment BEFORE resetting failed logins (only reset on full
                        auth)
                98  +     if ($user->is2FactorAuthEnabled()) {
                99  +         $otp = $body['otp'] ?? null;
               100  +
               101  +         if (empty($otp)) {
               102  +             // No OTP provided — tell client to prompt for it (don't reset
                        failed logins yet)
               103  +             return SlimUtils::renderJSON($response, ['requiresOTP' => true],
                        202);
               104  +         }
               105  +
               106  +         // Validate OTP or recovery code
               107  +         $otpValid = $user->isTwoFACodeValid($otp);
               108  +         $recoveryValid = !$otpValid && $user->isTwoFaRecoveryCodeValid($otp);
               109  +
               110  +         if (!$otpValid && !$recoveryValid) {
               111  +             $logger->warning('API login: invalid 2FA code', ['username' =>
                        $user->getUserName()]);
```

```
+                throw new HttpUnauthorizedException($request, $genericError);
+            }
+
+            // Persist consumed recovery code if one was used
+            if ($recoveryValid) {
+                $user->save();
+            }
        }

+       // Full authentication complete — reset failed login counter
+       $user->setFailedLogins(0);
+       $user->save();
+
        return SlimUtils::renderJSON($response, ['apiKey' => $user->getApiKey()]);
    }

```

Comments 0

Please sign in to comment.