chamilo / **chamilo-lms** Public

Code Issues 415 Pull requests 13 Discussions Actions Projects

Commit 1739371

AngelFQC committed 3 weeks ago Verified

```
Security: API: Restrict unprivileged user access to /api/users collection with
field-level and visibility filters.

See advisory GHSA-fp2p-fj6c-x3x9
```

master · v2.0.0 v2.0.0-RC.3

1 parent af6b700 commit 1739371

5 files changed +206 -10 lines changed

Top

Filter files…

- assets/vue/services
  - userService.js
- config
  - services.yaml
- src/CoreBundle
  - Entity
    - User.php
  - Serializer
    - UserSerializerContextBuilder.php
  - State
    - UserCollectionStateProvider.php

5 files changed +206 -10 lines changed

Search within code

assets/vue/services/userService.js

```
@@ -28,14 +28,6 @@ async function findUsersForSessionAdmin(searchParams) {
    return await baseService.get("/admin/sessionadmin/users", searchParams)
  }

- /**
-  * @param {string} username
-  * @returns {Promise<{totalItems, items}>}
-  */
- async function findByUsername(username) {
-   return await baseService.getCollection("/api/users", { username })
- }
-
  /**
   * @param {string} term
   * @returns {Promise<{totalItems, items}>}
@@ -58,7 +50,6 @@ export default {
    findById,
    findAll,
    findUsersForSessionAdmin,
-   findByUsername,
    findBySearchTerm,
    createOnAccessUrl,
  }
```

config/services.yaml

```
@@ -88,6 +88,12 @@ services:
                - name: kernel.event_subscriber
                  dispatcher: security.event_dispatcher.main

+    Chamilo\CoreBundle\Serializer\UserSerializerContextBuilder:
+        decorates: api_platform.serializer.context_builder
+        arguments:
+            - '@.inner'
+            - '@security.helper'
+
      Chamilo\CoreBundle\State\CStudentPublicationPostStateProcessor:
          bind:
              $persistProcessor:
'@api_platform.doctrine.orm.state.persist_processor'
```

src/CoreBundle/Entity/User.php

```
@@ -25,6 +25,7 @@
25 25  use Chamilo\CoreBundle\Entity\Listener\UserListener;
26 26  use Chamilo\CoreBundle\Filter\PartialSearchOrFilter;
27 27  use Chamilo\CoreBundle\Repository\Node\UserRepository;
   28 + use Chamilo\CoreBundle\State\UserCollectionStateProvider;
28 29  use Chamilo\CoreBundle\Traits\UserCreatorTrait;
29 30  use Chamilo\CourseBundle\Entity\CGroupRelTutor;
30 31  use Chamilo\CourseBundle\Entity\CGroupRelUser;
@@ -60,7 +61,10 @@
60 61              ),
61 62              new Put(security: "is_granted('EDIT', object)"),
62 63              new Delete(security: "is_granted('DELETE', object)"),
63    -         new GetCollection(security: "is_granted('ROLE_USER')"),
   64 +         new GetCollection(
   65 +             security: "is_granted('ROLE_USER')",
   66 +             provider: UserCollectionStateProvider::class,
   67 +         ),
64 68              new Post(security: "is_granted('ROLE_ADMIN')"),
65 69              new GetCollection(
66 70                  uriTemplate: '/users/{id}/skills',
@@ -163,6 +167,7 @@ class User implements UserInterface, EquatableInterface,
ResourceInterface, Reso
163 167         #[Groups([
164 168             'user_export',
165 169             'user:read',
   170 +         'user:read:public',
166 171             'resource_node:read',
167 172             'document:read',
168 173             'media_object_read',
@@ -179,6 +184,7 @@ class User implements UserInterface, EquatableInterface,
ResourceInterface, Reso
179 184
180 185         #[Groups([
181 186             'user:read',
   187 +         'user:read:public',
182 188             'course:read',
183 189             'resource_node:read',
```

```
184  190            'user_json:read',
     @@ -196,6 +202,7 @@ class User implements UserInterface, EquatableInterface,
     ResourceInterface, Reso
196  202        #[Groups([
197  203            'user_export',
198  204            'user:read',
     205 +          'user:read:public',
199  206            'user:write',
200  207            'course:read',
201  208            'course_rel_user:read',
     @@ -219,6 +226,7 @@ class User implements UserInterface, EquatableInterface,
     ResourceInterface, Reso
219  226        #[ApiProperty(iris: ['http://schema.org/name'])]
220  227        #[Groups([
221  228            'user:read',
     229 +          'user:read:public',
222  230            'user:write',
223  231            'resource_node:read',
224  232            'user_json:read',
     @@ -233,6 +241,7 @@ class User implements UserInterface, EquatableInterface,
     ResourceInterface, Reso
233  241
234  242        #[Groups([
235  243            'user:read',
     244 +          'user:read:public',
236  245            'user:write',
237  246            'resource_node:read',
238  247            'user_json:read',
```

…Serializer/UserSerializerContextBuilder.php

```
     @@ -0,0 +1,62 @@
     1 + <?php
     2 +
     3 + /* For licensing terms, see /license.txt */
     4 +
     5 + declare(strict_types=1);
     6 +
     7 + namespace Chamilo\CoreBundle\Serializer;
     8 +
```

```
9  + use ApiPlatform\Metadata\CollectionOperationInterface;
10 + use ApiPlatform\State\SerializerContextBuilderInterface;
11 + use Chamilo\CoreBundle\Entity\User;
12 + use Symfony\Bundle\SecurityBundle\Security;
13 + use Symfony\Component\HttpFoundation\Request;
14 +
15 + /**
16 +  * Decorates the API Platform serializer context builder to restrict the
17 +  * fields returned by GET /api/users (collection) for unprivileged users.
18 +  *
19 +  * Admins, teachers and session managers keep the full "user:read" group.
20 +  * All other authenticated roles receive only "user:read:public", which
21 +  * exposes id, username, firstname, lastname and illustrationUrl — never
22 +  * email, phone, roles or address.
23 +  */
24 + final readonly class UserSerializerContextBuilder implements
     SerializerContextBuilderInterface
25 + {
26 +     public function __construct(
27 +         private SerializerContextBuilderInterface $decorated,
28 +         private Security $security,
29 +     ) {}
30 +
31 +     public function createFromRequest(Request $request, bool $normalization, ?
     array $extractedAttributes = null): array
32 +     {
33 +         $context = $this->decorated->createFromRequest($request, $normalization,
     $extractedAttributes);
34 +
35 +         if (!$normalization) {
36 +             return $context;
37 +         }
38 +
39 +         if (($context['resource_class'] ?? null) !== User::class) {
40 +             return $context;
41 +         }
42 +
43 +         $operation = $context['operation'] ?? null;
44 +         if (!$operation instanceof CollectionOperationInterface) {
45 +             return $context;
```

```
+         }
+
+         if (
+             $this->security->isGranted('ROLE_ADMIN')
+             || $this->security->isGranted('ROLE_SUPER_ADMIN')
+             || $this->security->isGranted('ROLE_GLOBAL_ADMIN')
+             || $this->security->isGranted('ROLE_TEACHER')
+             || $this->security->isGranted('ROLE_SESSION_MANAGER')
+         ) {
+             return $context;
+         }
+
+         $context['groups'] = ['user:read:public'];
+
+         return $context;
+     }
+ }
```

...undle/State/UserCollectionStateProvider.php

```
@@ -0,0 +1,128 @@
+ <?php
+
+ /* For licensing terms, see /license.txt */
+
+ declare(strict_types=1);
+
+ namespace Chamilo\CoreBundle\State;
+
+ use ApiPlatform\Doctrine\Orm\Extension\FilterExtension;
+ use ApiPlatform\Doctrine\Orm\Extension\OrderExtension;
+ use ApiPlatform\Doctrine\Orm\Extension\PaginationExtension;
+ use ApiPlatform\Doctrine\Orm\Extension\QueryResultCollectionExtensionInterface;
+ use ApiPlatform\Doctrine\Orm\State\CollectionProvider;
+ use ApiPlatform\Doctrine\Orm\Util\QueryNameGenerator;
+ use ApiPlatform\Metadata\GetCollection;
+ use ApiPlatform\Metadata\Operation;
+ use ApiPlatform\State\ProviderInterface;
+ use Chamilo\CoreBundle\Entity\User;
+ use Chamilo\CoreBundle\Entity\UserRelUser;
+ use Chamilo\CoreBundle\Helpers\UserHelper;
```

```
21 + use Chamilo\CoreBundle\Repository\Node\UserRepository;
22 + use Symfony\Bundle\SecurityBundle\Security;
23 +
24 + /**
25 +  * State provider for GET /api/users (collection).
26 +  *
27 +  * - Admins, teachers and session managers receive the full "user:read" group
28 +  *   and no extra visibility filter: they can list all platform users.
29 +  *
30 +  * - Every other authenticated role receives only "user:read:public" (id,
31 +  *   username, firstname, lastname, illustrationUrl) and the result set is
32 +  *   restricted to users they are already related to via UserRelUser (friends,
33 +  *   boss/HRM relationships, etc.) plus themselves.
34 +  *
35 +  * All registered API Platform query filters (search, order, pagination) still
36 +  * apply on top of the visibility constraint.
37 +  *
38 +  * @template-implements ProviderInterface<User>
39 +  */
40 + final class UserCollectionStateProvider implements ProviderInterface
41 + {
42 +     private array $extensions;
43 +
44 +     public function __construct(
45 +         private readonly CollectionProvider $collectionProvider,
46 +         private readonly UserRepository $userRepository,
47 +         private readonly UserHelper $userHelper,
48 +         private readonly Security $security,
49 +         FilterExtension $filterExtension,
50 +         OrderExtension $orderExtension,
51 +         PaginationExtension $paginationExtension,
52 +     ) {
53 +         $this->extensions = [$filterExtension, $orderExtension,
    $paginationExtension];
54 +     }
55 +
56 +     public function provide(Operation $operation, array $uriVariables = [],
    array $context = []): iterable
57 +     {
58 +         if (!$operation instanceof GetCollection) {
```

```
59  +            return $this->collectionProvider->provide($operation,
    $uriVariables, $context);
60  +        }
61  +
62  +        $currentUser = $this->userHelper->getCurrent();
63  +
64  +        // Privileged roles: delegate entirely to the default provider with
    full groups.
65  +        if (
66  +            $this->security->isGranted('ROLE_ADMIN')
67  +            || $this->security->isGranted('ROLE_SUPER_ADMIN')
68  +            || $this->security->isGranted('ROLE_GLOBAL_ADMIN')
69  +            || $this->security->isGranted('ROLE_TEACHER')
70  +            || $this->security->isGranted('ROLE_SESSION_MANAGER')
71  +        ) {
72  +            return $this->collectionProvider->provide($operation,
    $uriVariables, $context);
73  +        }
74  +
75  +        // Unprivileged users: restrict visible rows.
76  +        // Field restriction (email/phone/roles) is handled separately by
77  +        // UserSerializerContextBuilder, which swaps the normalization group to
78  +        // "user:read:public" before the serializer runs.
79  +
80  +        // Build a visibility-scoped QueryBuilder and apply all API Platform
81  +        //    extensions on top (so search filters, ordering and pagination
    work).
82  +        if (!$currentUser instanceof User) {
83  +            return [];
84  +        }
85  +
86  +        $qb = $this->userRepository->createQueryBuilder('u');
87  +
88  +        $qb->andWhere(
89  +            $qb->expr()->orX(
90  +                // The user themselves.
91  +                'u = :currentUser',
92  +                // Relationships the current user initiated.
93  +                $qb->expr()->exists(
94  +                    'SELECT 1 FROM '.UserRelUser::class.' uru1
```

```
95  +                      WHERE uru1.user = :currentUser
96  +                        AND uru1.friend = u
97  +                        AND uru1.relationType NOT IN (:deletedRel)'
98  +                ),
99  +                // Relationships where the current user is the target.
100 +                $qb->expr()->exists(
101 +                    'SELECT 1 FROM '.UserRelUser::class.' uru2
102 +                     WHERE uru2.friend = :currentUser
103 +                       AND uru2.user = u
104 +                       AND uru2.relationType NOT IN (:deletedRel)'
105 +                ),
106 +            )
107 +        )
108 +            ->setParameter('currentUser', $currentUser)
109 +            ->setParameter('deletedRel',
    [UserRelUser::USER_RELATION_TYPE_DELETED])
110 +        ;
111 +
112 +        $queryNameGenerator = new QueryNameGenerator();
113 +        $items = [];
114 +
115 +        foreach ($this->extensions as $extension) {
116 +            $extension->applyToCollection($qb, $queryNameGenerator,
    User::class, $operation, $context);
117 +
118 +            if (
119 +                $extension instanceof QueryResultCollectionExtensionInterface
120 +                && $extension->supportsResult(User::class, $operation,
    $context)
121 +            ) {
122 +                $items = $extension->getResult($qb, User::class, $operation,
    $context);
123 +            }
124 +        }
125 +
126 +        return [] !== $items ? $items : $qb->getQuery()->getResult();
127 +    }
128 + }
```

Comments 0

Please sign in to comment.