Policy-based Access Control für Stud.IP

Marcus Eibrink-Lunzenauer - elan e.V.

SET 26

Autorisierung

Warum?

Top 10 der OWASP Foundation

OWASP

https://owasp.org/Top10/2025/

https://owasp.org/Top10/2025/

- *2010* → Broken Access Control: **Platz 3**

https://owasp.org/Top10/2025/

- *2010* → Broken Access Control: **Platz 3**
- *2013* → Broken Access Control: **Platz 2**

https://owasp.org/Top10/2025/

- *2010* → Broken Access Control: **Platz 3**
- *2013* → Broken Access Control: **Platz 2**
- *2017* → Broken Access Control: **Platz 2**

https://owasp.org/Top10/2025/

- *2010* → Broken Access Control: **Platz 3**
- *2013* → Broken Access Control: **Platz 2**
- *2017* → Broken Access Control: **Platz 2**
- *2021* → Broken Access Control: **Platz 1**

https://owasp.org/Top10/2025/

- *2010* → Broken Access Control: **Platz 3**
- *2013* → Broken Access Control: **Platz 2**
- *2017* → Broken Access Control: **Platz 2**
- *2021* → Broken Access Control: **Platz 1**
- *2025* → Broken Access Control: **Platz 1**

Broken Access Control

Definition

> *“Access Control” sorgt dafür, dass Nutzende nicht über ihre vorgesehenen Berechtigungen hinaus handeln können. Fehler führen in der Regel dazu, dass Informationen unbefugt offengelegt, Daten verändert oder zerstört werden oder dass ein Geschäftsprozess außerhalb der Befugnisse der Nutzenden ausgeführt wird.*

https://owasp.org/Top10/2025/A01_2025-Broken_Access_Control/#description

How to prevent?

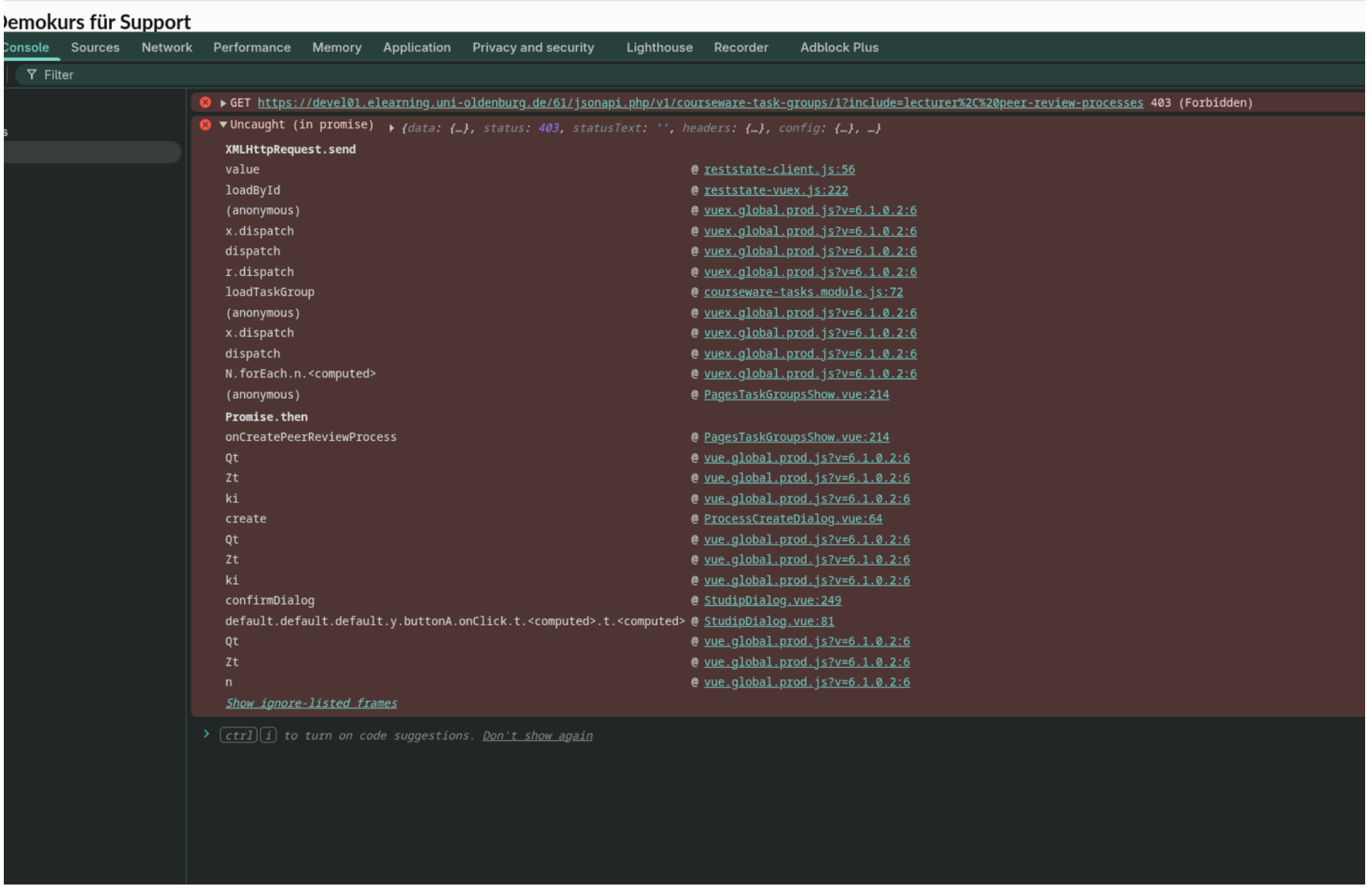
> *Implement access control mechanisms once and reuse them throughout the application*

https://owasp.org/Top10/2025/A01_2025-Broken_Access_Control/#how-to-prevent

Warum ist Access Control kaputt?

- hartkodierte Regeln
- verstreut über viele Dateien
- Ausnahmen und Sonderfälle
- Gefahr, dass nicht alle Feinheiten bei allen Entwickelnden im Kopf sind
- Jede Regeländerung birgt Risiken, dass etwas kaputt geht.

Aktuelles Beispiel



Lösungsvorschlag: Policy-based Access Control

Wer darf was unter welchen Bedingungen?

Ein alter Hut

1976: IBM Resource Access Control Facility für Mainframes

RBAC

Role-based access control

```
1 if (!$GLOBALS['perm']->have_studip_perm('tutor', $sem_id) {
2     throw new \AccessDeniedException();
3 }
```

ABAC

Attribute-based access control

```
1  if ($document->sensitivity === 'secret' && $user->clearance !== 'high') {
2      return false;
3  }
```

Zutaten

[Wer] darf [was] unter [welchen Bedingungen]?

Wer? → *Subject!*

Was? → *Action + Resource!*

Welche Bedingungen? → *Policies!*

UND: eine einheitliche Schnittstelle

Ein Prototyp

1. Einheitliche Schnittstelle

Interface

```php
interface AccessControl
{
    public function authorize(
        UnitEnum|string $action,
        object|string $resource,
        ?User $subject = null,
        mixed ...$context,
    ): void;

    public function allowedTo(
        UnitEnum|string $action,
        object|string $resource,
        ?User $subject = null,
        mixed ...$context,
    ): AccessDecision;
}
```

Interface

```php
interface AccessControl
{
    public function authorize(
        UnitEnum|string $action,
        object|string $resource,
        ?User $subject = null,
        mixed ...$context,
    ): void;

    public function allowedTo(
        UnitEnum|string $action,
        object|string $resource,
        ?User $subject = null,
        mixed ...$context,
    ): AccessDecision;
}
```

Interface

```php
interface AccessControl
{
    public function authorize(
        UnitEnum|string $action,
        object|string $resource,
        ?User $subject = null,
        mixed ...$context,
    ): void;

    public function allowedTo(
        UnitEnum|string $action,
        object|string $resource,
        ?User $subject = null,
        mixed ...$context,
    ): AccessDecision;
}
```

Verwendung

```php
$accessControl->authorize('view', $post);

if ($accessControl->allowedTo('edit', $post)) {
    // …
}
```

Verwendung

```
1 $accessControl->authorize('view', $post);
2
3 if ($accessControl->allowedTo('edit', $post)) {
4     // …
5 }
```

2. Policies

Berechtigungscode zentralisieren

```
1  if ($posting->user_id === $user->user_id
2      || $GLOBALS['perm']->have_studip_perm('tutor', $posting->range_id, $user
3      // …
4  }
5
6  function edit(User $user, Posting $posting)
7  {
8      return $posting->user_id === $user->user_id
9          || $GLOBALS['perm']->have_studip_perm('tutor', $posting->range_id, $
10 }
```

Berechtigungscode zentralisieren

```
 1 if ($posting->user_id === $user->user_id
 2     || $GLOBALS['perm']->have_studip_perm('tutor', $posting->range_id, $user
 3     // …
 4 }
 5
 6 function edit(User $user, Posting $posting)
 7 {
 8     return $posting->user_id === $user->user_id
 9         || $GLOBALS['perm']->have_studip_perm('tutor', $posting->range_id, $
10 }
```

Design-Entscheidungen

- Wie kennzeichnen wir Policies?
- Wie benenne ich Policies?
- Wo liegen Policies?
- Wie findet die `AccessControl` Policies?
- Wie werden Aktionen und Ressourcen abgebildet?
- Welche Signatur haben Policies?

Policy-Beispiel

```php
class PostingPolicy
{
    #[Policy]
    public function edit(User $user, Posting $resource): bool
    {
        return $posting->user_id === $user->user_id
            || $GLOBALS['perm']->have_studip_perm('tutor', $posting->range_id
    }
```

Rückgabewert

```php
final class AccessDecision
{
    public static function granted(): self
    {
        return new self(granted: true);
    }

    public static function denied(?string $message = null): self
    {
        return new self(granted: false, message: $message);
    }
}

return AccessDecision::denied("You must not delete the last item.");
```

Auflöse-Algorithmus

```php
/** @var Posting $post */
$accessControl->authorize('edit', $post);

#[Policy]
public function edit(User $user, Posting $resource): bool
{
    // …
}
```

Enums für Actions

```php
$accessControl->authorize(StudipNewsAction::View, $post);

enum StudipNewsAction
{
    case View;
    // …
}

#[Policy(action: StudipNewsAction::View)]
public function view(User $user, Posting $resource): bool
{
    // …
}
```

Klassenbasierte Policies

```php
$accessControl->authorize('create', StudipNews::class);

#[Policy(StudipNews::class)]
public function create(User $user): bool
{
    // …
}
```

Mehr Kontext

```php
/** @var Course course */
$accessControl->authorize('view-any-in-range', StudipNews::class, $course);

#[Policy(StudipNews::class)]
public function viewAnyInRange(User $user, Range|string $range): bool
{
    // …
}
```

Mehrere Policies für (Action, Resource)

3. Framework-Integration

Slim/Trails-Controller

```
#[Inject]
private readonly AccessControl $accessControl;
```

Route-Middleware

```
#[Can(action: "create", resource: StudipNews::class]
public function createNews(Request $request, Response $response) {
    // …
}
```

Templates

```
<? if (accessControl()->allowedTo('update', $news)) : ?>
```

```
{% can 'update', news %}
    {# The current user can update the StudipNews ... #}
{% elsecan 'create', 'StudipNews' %}
    {# The current user can create new StudipNews ... #}
{% else %}
    {# ... #}
{% endcan %}
```

4. DX

Code-Generierung

```
./cli/studip make:policy StudipNewsPolicy --model=StudipNews
```

PHPStan-Plugin

Testing

- spezielle `asserts`
- AccessControl für einen Test deaktivieren

Tooling

- Liste aller (im Code verwendeten) Policies
- Liste aller nicht verwendeten Policies?

Weitere Überlegungen

Policy Scopes für gefilterte Collections

```php
#[Scope(StudipNews::class)]
function viewAny(User $user, Query $query)
{
    if ($GLOBALS['perm']->have_perm('root', $user->id)) {
        return $query->all();
    } else {
        return $query->where(['published' => true])->get();
    }
}

class StudipNewsIndex {
    public function __invoke(Request $request, Response $response): Response
    {
        $data = policy_scope(StudipNews::class);

        return $responses->getContentResponse($data);
    }
}
```

Policy Scopes für gefilterte Collections

```php
#[Scope(StudipNews::class)]
function viewAny(User $user, Query $query)
{
    if ($GLOBALS['perm']->have_perm('root', $user->id)) {
        return $query->all();
    } else {
        return $query->where(['published' => true])->get();
    }
}

class StudipNewsIndex {
    public function __invoke(Request $request, Response $response): Response
    {
        $data = policy_scope(StudipNews::class);

        return $responses->getContentResponse($data);
    }
}
```

Policy Scopes für gefilterte Collections

```php
#[Scope(StudipNews::class)]
function viewAny(User $user, Query $query)
{
    if ($GLOBALS['perm']->have_perm('root', $user->id)) {
        return $query->all();
    } else {
        return $query->where(['published' => true])->get();
    }
}

class StudipNewsIndex {
    public function __invoke(Request $request, Response $response): Response
    {
        $data = policy_scope(StudipNews::class);

        return $responses->getContentResponse($data);
    }
}
```

Policy Scopes für gefilterte Collections

```php
#[Scope(StudipNews::class)]
function viewAny(User $user, Query $query)
{
    if ($GLOBALS['perm']->have_perm('root', $user->id)) {
        return $query->all();
    } else {
        return $query->where(['published' => true])->get();
    }
}

class StudipNewsIndex {
    public function __invoke(Request $request, Response $response): Response
    {
        $data = policy_scope(StudipNews::class);

        return $responses->getContentResponse($data);
    }
}
```

Policy Scopes für gefilterte Collections

```php
#[Scope(StudipNews::class)]
function viewAny(User $user, Query $query)
{
    if ($GLOBALS['perm']->have_perm('root', $user->id)) {
        return $query->all();
    } else {
        return $query->where(['published' => true])->get();
    }
}

class StudipNewsIndex {
    public function __invoke(Request $request, Response $response): Response
    {
        $data = policy_scope(StudipNews::class);

        return $responses->getContentResponse($data);
    }
}
```

Policies ohne Ressourcen

```
$accessControl->authorize('view-dashboard');

#[Policy(???)]
public function viewDashboard(User $user): bool
{
    // …
}
```

Policy-Composition `(f∘g)(x)`

- Wiederverwendbare Regeln `OwnedByUser`, `RequiresRole` …
- Policies komponiert aus diesen Bausteinen

Sicherheit

- Immer Fehler, falls Policy fehlt!
- Wie stellen wir sicher, dass wir immer auf aktuellen Daten arbeiten?
- `SimpleORMap::restore`
- Transactions/Locks

Dokumentation

- Richtlinien für das Schreiben von Policies
- Beispiele für typische CRUD-Aktionen
- Best Practices
    - *“Nicht nur im Template checken, immer in den Controllers/Routes”*
    - *“Keine ”Business Logic“ in Policies”*

Fragen