

# **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](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](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 Entwicklenden im Kopf sind
- Jede Regeländerung birgt Risiken, dass etwas kaputt geht.

# Aktuelles Beispiel

## Demokurs für Support

The screenshot shows the Chrome DevTools Console with the following content:

- Navigation bar: Console, Sources, Network, Performance, Memory, Application, Privacy and security, Lighthouse, Recorder, Adblock Plus.
- Filter: Filter
- Log entry 1: `▶ GET https://devel01.elearning.uni-oldenburg.de/61/jsonapi.php/v1/courseware-task-groups/1?include=lecturer%2C%20peer-review-processes 403 (Forbidden)`
- Log entry 2: `▼ Uncaught (in promise) ▶ {data: {...}, status: 403, statusText: '', headers: {...}, config: {...}, ...}`
- Stack trace for the error:
  - `XMLHttpRequest.send`
  - `value` @ `reststate-client.js:56`
  - `loadById` @ `reststate-vuex.js:222`
  - `(anonymous)` @ `vuex.global.prod.js?v=6.1.0.2:6`
  - `x.dispatch` @ `vuex.global.prod.js?v=6.1.0.2:6`
  - `dispatch` @ `vuex.global.prod.js?v=6.1.0.2:6`
  - `r.dispatch` @ `vuex.global.prod.js?v=6.1.0.2:6`
  - `loadTaskGroup` @ `courseware-tasks.module.js:72`
  - `(anonymous)` @ `vuex.global.prod.js?v=6.1.0.2:6`
  - `x.dispatch` @ `vuex.global.prod.js?v=6.1.0.2:6`
  - `dispatch` @ `vuex.global.prod.js?v=6.1.0.2:6`
  - `N.forEach.n.<computed>` @ `vuex.global.prod.js?v=6.1.0.2:6`
  - `(anonymous)` @ `PagesTaskGroupsShow.vue:214`
  - `Promise.then`
  - `onCreatePeerReviewProcess` @ `PagesTaskGroupsShow.vue:214`
  - `Qt` @ `vue.global.prod.js?v=6.1.0.2:6`
  - `Zt` @ `vue.global.prod.js?v=6.1.0.2:6`
  - `ki` @ `vue.global.prod.js?v=6.1.0.2:6`
  - `create` @ `ProcessCreateDialog.vue:64`
  - `Qt` @ `vue.global.prod.js?v=6.1.0.2:6`
  - `Zt` @ `vue.global.prod.js?v=6.1.0.2:6`
  - `ki` @ `vue.global.prod.js?v=6.1.0.2:6`
  - `confirmDialog` @ `StudipDialog.vue:249`
  - `default.default.default.y.buttonA.onClick.t.<computed>.t.<computed>` @ `StudipDialog.vue:81`
  - `Qt` @ `vue.global.prod.js?v=6.1.0.2:6`
  - `Zt` @ `vue.global.prod.js?v=6.1.0.2:6`
  - `n` @ `vue.global.prod.js?v=6.1.0.2:6`
- Footer: `> ctrl i to turn on code suggestions. Don't show again`



# **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

```
1 interface AccessControl
2 {
3     public function authorize(
4         UnitEnum|string $action,
5         object|string $resource,
6         ?User $subject = null,
7         mixed ...$context,
8     ): void;
9
10    public function allowedTo(
11        UnitEnum|string $action,
12        object|string $resource,
13        ?User $subject = null,
14        mixed ...$context,
15    ): AccessDecision;
16 }
```

## Interface

```
1 interface AccessControl
2 {
3     public function authorize(
4         UnitEnum|string $action,
5         object|string $resource,
6         ?User $subject = null,
7         mixed ...$context,
8     ): void;
9
10    public function allowedTo(
11        UnitEnum|string $action,
12        object|string $resource,
13        ?User $subject = null,
14        mixed ...$context,
15    ): AccessDecision;
16 }
```

## Interface

```
1 interface AccessControl
2 {
3     public function authorize(
4         UnitEnum|string $action,
5         object|string $resource,
6         ?User $subject = null,
7         mixed ...$context,
8     ): void;
9
10    public function allowedTo(
11        UnitEnum|string $action,
12        object|string $resource,
13        ?User $subject = null,
14        mixed ...$context,
15    ): AccessDecision;
16 }
```

## Verwendung

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

## 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

```
1 class PostingPolicy
2 {
3     #[Policy]
4     public function edit(User $user, Posting $resource): bool
5     {
6         return $posting->user_id === $user->user_id
7             || $GLOBALS['perm']->have_studip_perm('tutor', $posting->range_id
8     }
```

## Rückgabewert

```
1 final class AccessDecision
2 {
3     public static function granted(): self
4     {
5         return new self(granted: true);
6     }
7
8     public static function denied(?string $message = null): self
9     {
10        return new self(granted: false, message: $message);
11    }
12 }
13
14 return AccessDecision::denied("You must not delete the last item.");
```

## Auflöse-Algorithmus

```
/** @var Posting $post */  
$accessControl->authorize('edit', $post);  
  
#[Policy]  
public function edit(User $user, Posting $resource): bool  
{  
    // ...  
}
```

## Enums für Actions

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

enum StudipNewsAction
{
    case View;
    // ...
}

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

## Klassenbasierte Policies

```
$accessControl->authorize('create', StudipNews::class);  
  
#[Policy(StudipNews::class)]  
public function create(User $user): bool  
{  
    // ...  
}
```

## Mehr Kontext

```
/** @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 ... #}  
{% elseifcan '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

```
1 #[Scope(StudipNews::class)]
2 function viewAny(User $user, Query $query)
3 {
4     if ($GLOBALS['perm']->have_perm('root', $user->id)) {
5         return $query->all();
6     } else {
7         return $query->where(['published' => true])->get();
8     }
9 }
10
11 class StudipNewsIndex {
12     public function __invoke(Request $request, Response $response): Response
13     {
14         $data = policy_scope(StudipNews::class);
15
16         return $responses->getContentResponse($data);
17     }
18 }
```

## Policy Scopes für gefilterte Collections

```
1 #[Scope(StudipNews::class)]
2 function viewAny(User $user, Query $query)
3 {
4     if ($GLOBALS['perm']->have_perm('root', $user->id)) {
5         return $query->all();
6     } else {
7         return $query->where(['published' => true])->get();
8     }
9 }
10
11 class StudipNewsIndex {
12     public function __invoke(Request $request, Response $response): Response
13     {
14         $data = policy_scope(StudipNews::class);
15
16         return $responses->getContentResponse($data);
17     }
18 }
```

## Policy Scopes für gefilterte Collections

```
1 #[Scope(StudipNews::class)]
2 function viewAny(User $user, Query $query)
3 {
4     if ($GLOBALS['perm']->have_perm('root', $user->id)) {
5         return $query->all();
6     } else {
7         return $query->where(['published' => true])->get();
8     }
9 }
10
11 class StudipNewsIndex {
12     public function __invoke(Request $request, Response $response): Response
13     {
14         $data = policy_scope(StudipNews::class);
15
16         return $responses->getContentResponse($data);
17     }
18 }
```

## Policy Scopes für gefilterte Collections

```
1 #[Scope(StudipNews::class)]
2 function viewAny(User $user, Query $query)
3 {
4     if ($GLOBALS['perm']->have_perm('root', $user->id)) {
5         return $query->all();
6     } else {
7         return $query->where(['published' => true])->get();
8     }
9 }
10
11 class StudipNewsIndex {
12     public function __invoke(Request $request, Response $response): Response
13     {
14         $data = policy_scope(StudipNews::class);
15
16         return $responses->getContentResponse($data);
17     }
18 }
```

## Policy Scopes für gefilterte Collections

```
1 #[Scope(StudipNews::class)]
2 function viewAny(User $user, Query $query)
3 {
4     if ($GLOBALS['perm']->have_perm('root', $user->id)) {
5         return $query->all();
6     } else {
7         return $query->where(['published' => true])->get();
8     }
9 }
10
11 class StudipNewsIndex {
12     public function __invoke(Request $request, Response $response): Response
13     {
14         $data = policy_scope(StudipNews::class);
15
16         return $responses->getContentResponse($data);
17     }
18 }
```

## Policies ohne Ressourcen

```
$accessControl->authorize('view-dashboard');  
  
#[Policy(???)]  
public function viewDashboard(User $user): bool  
{  
    // ...  
}
```

## Policy-Composition ( $f \circ g$ ) ( 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