



# Une gestion dynamique des permissions en PHP

---





- ★ SCOP détenue à 100% par ses salarié·e·s
- ★ Gouvernance démocratique (1 personne = 1 voix) et transparente
- ★ Experts API, web & Cloud
- ★ Développement : PHP, JavaScript, Go, Rust...
- ★ Formation, audit & architecture





# Sécurité et Permissions

---

Notions de base

# Définitions

---



## UTILISATEUR·RICE

Entité qui demande à intéragir avec le système



## RESSOURCE

Donnée du système sur laquelle on souhaite effectuer des actions



## RÔLE

Étiquette attribuée à un groupe d'utilisateur·rice·s pour définir des permissions



## PERMISSION

Règle appliquée à un·e utilisateur·rice ou un rôle, qui accorde la capacité de faire une action sur une ressource

# Authentification

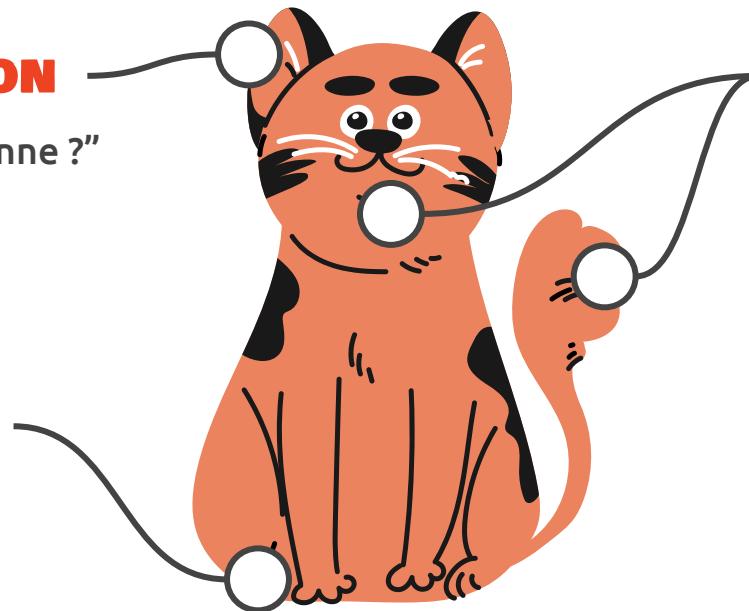
---

## IDENTIFICATION

“Qui est cette personne ?”

## CONFIRMER

L'identité de la personne



## FACTEURS D'AUTHENTIFICATION

Identifiant, mot de passe, clé d'API, token, carte d'identité...



# Autorisation

---



## PERMISSIONS & ACCÈS

"Que peut faire cette personne ?"



## DÉCISIONS D'AUTORISATION

Oui / Non / Ne se prononce pas



## EXEMPLE

Modifier le détail d'un événement



FORUM PHP  
PARIS 2024

# Principe du moindre privilège

---

- ★ Accorde le niveau d'accès **minimum requis** pour accomplir une tâche
- ★ Ouvrir l'accès **spécifiquement**
- ★ Utilisation de **règles directes**
- ★ Plus clair, lisible et surtout plus **sécurisé**
- ★ S'étend au-delà des personnes

```
Order deny,allow  
Deny from all  
Allow from xxx.xxx.xxx.xxx
```



# Discretionary Access Control (DAC)

## Contrôle d'accès discrétionnaire

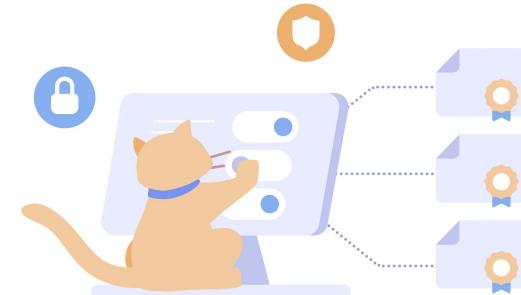
- ★ La propriétaire de la ressource a le **pouvoir décisionnaire** sur ses permissions
- ★ Elle peut **déléguer** les droits à d'autres entités
- ★ Contrôle **décentralisé** des accès



# Access Control List (ACL)

## Liste de contrôle d'accès

- ★ Ensemble de règles qui spécifient les **opérations** que pourra réaliser **une identité** sur une ressource
- ★ Identité + Permissions + Actions autorisées ou interdites
- ★ **Granularité fine**



# Mandatory Access Control (**MAC**)

## Contrôle d'accès obligatoire

- ★ Une **autorité centrale** (admin) régule les droits d'accès
- ★ Les décisions ne peuvent être prises par la propriétaire / pas de délégation
- ★ Les droits d'accès sont **organisés en niveaux** et accordés sous forme d'autorisation



# Role-Based Access Control (RBAC)

## Contrôle d'accès basé sur les rôles

- ★ Attribution d'un ou plusieurs **rôles** aux personnes
- ★ **Regroupe** les personnes par fonction similaire
- ★ La gestion des **permissions** est faite par rôle



# Attribute-Based Access Control (ABAC)

## Contrôle d'accès basé sur les attributs

- ★ Évalue un **ensemble de règles** pour gérer les droits d'accès en fonction d'**attributs**
- ★ Applique une **logique booléenne** pour accorder ou refuser l'accès
- ★ Évaluation plus ou moins complexe en fonction des **règles métier**
- ★ Contrôle d'accès **plus granulaire** qu'avec RBAC et plus **dynamique**



# Les voters de Symfony

---

- ★ Les **voters** centralisent la logique d'autorisation
- ★ Un voter retourne **GRANT**, **DENY** ou **ABSTAIN**
- ★ Plusieurs **stratégies** : affirmative, consensus, unanimous, priority
- ★ Utilisation de **voters personnalisés**



```
final class EventVoter extends Voter
{
    public function __construct(private readonly PlanningRepository $planningRepository) {}

    protected function supports(string $attribute, mixed $subject): bool
    {
        return 'event_update' === $attribute && $subject instanceof Event;
    }

    protected function voteOnAttribute(string $attribute, mixed $subject, TokenInterface $token): bool
    {
        $user = $token->getUser();
        if (!$user instanceof User) {
            return false;
        }

        $assigned = $user->isAssignedTo($subject);
        $planning = $this->planningRepository->findOneBy(['user' => $user, 'date' => new \DateTime()]);

        return $assigned && null !== $planning;
    }
}
```

# Les voters de Symfony

---

```
$authorizationChecker->isGranted('ROLE_ORGANIZER');
$authorizationChecker->isGranted('event_update', $event);

#[IsGranted(
    attribute: new Expression('user === subject'),
    subject: new Expression('args["event"].getOrganizer()'),
)]
public function __invoke(Event $event): Response {}
```





**Des permissions  
dynamiques et  
administrables**

---

# Problématique

---

- Un système de permissions basé sur les rôles atteint vite ses **limites**
- On souhaite un **système flexible et configurable**
- L'admin doit pouvoir **modifier les permissions en détail, sans avoir à modifier le code**, ou redéployer l'application
- Les règles doivent être appliquées à chaque requête
- Les règles peuvent **dépendre d'attributs, de l'état d'autres objets ou de relations**
- Sécurité vs **performances**



# Définir son modèle

- Examiner les **besoins métier**
  - ◆ type d'autorisations nécessaires
  - ◆ comment les personnes travaillent-elles ?
  - ◆ quelles sont leurs tâches ?
- Définir les **rôles et les attributs de sécurité**
- Définir les **politiques de contrôle d'accès**
- **Stocker les permissions**
- **Évaluer les permissions**



RBAC

**Accès globaux**

ABAC

**Règles métier**

ACL

**Granularité individuelle**

# Combinaison de plusieurs modèles

*ex : toutes les responsables de service peuvent accéder aux ressources*

*ex : uniquement les responsables du département financier, en poste à un instant T, avec ancienneté > 2ans*

*ex : cibler l'entité propriétaire d'une ressource*



# Stocker le modèle

## Fichier de config

```
default allow := false

allow {
    input.action == "update"
    input.user.role == "organizer"
    input.event.owner_id == input.user.id
}

allow {
    input.action == "create"
    input.user.role == "organizer"
    input.user.seniority >= 1
}

allow {
    input.action == "update"
    input.user.role == "organizer"
    input.user.team == input.event.team
}
```

```
{
    "rules": [
        {
            "action": "update",
            "resource": "event",
            "attributes": {
                "role": "manager",
                "department": "HR",
                "min_seniority": 2
            }
        }
    ]
}
```



# Stocker le modèle



## Solution d'identité (ex : keycloak)

The screenshot shows the Keycloak interface for managing clients. On the left, a sidebar menu is open under the "Photoz" client, with "Clients" selected. The main area displays the "photoz-restful-api" client details. The "Authorization" tab is active, showing a table of permissions:

Name	Type	Associated policy	Description
Admin Resource Permission	Resource-Based	Administration Policy	General policy for any administrative resource.
Album Resource Permission	Scope-Based	Only Owner and Administrators Policy	A default permission that defines access for any album resource
Default Resource Permission	Resource-Based	Only From @keycloak.org or Admin	Defines who is allowed to view common resources

Source : [keycloak.org](https://keycloak.org)



# Stocker le modèle

## Solution d'identité (ex : casbin)



Node-Casbin v5.31.0

Model	Policy
<p>ABAC with policy rule</p> <pre>1 [request_definition] 2 r = sub, obj, act 3 4 [policy_definition] 5 p = sub_rule, obj, act 6 7 [policy_effect] 8 e = some(where (p.eft == allow)) 9 10 [matchers] 11 m = eval(p.sub_rule) &amp;&amp; r.obj == p.obj &amp;&amp; r.act == p.act</pre>	<p>Ask AI</p> <pre>1 p, r.sub.Age &gt; 18 &amp;&amp; r.sub.Age &lt; 60, /data1, read</pre>
<p>Request</p> <p>r    p    e    m</p> <pre>1 { Age: 30}, /data1, read</pre>	<p>Enforcement Result</p> <p>Ask AI</p> <pre>1 true Reason: ["r.sub.Age &gt; 18 &amp;&amp; r.sub.Age &lt; 60","/data1","read"]</pre>

Source : [casbin.org/editor](https://casbin.org/editor)

# Stocker le modèle

---

## Au niveau du **serveur**

- ★ Plugin de sécurité sur le **serveur web**
- ★ Accorde les accès aux ressources selon les attributs du **jeton JWT**, les polices d'autorisation et les règles ACL
- ★ Gère les autorisations avant même que la requête n'arrive à l'application
- ★ Impose d'avoir des rôles fixes ou des attributs présents dans le jeton JWT



# Stocker le Modèle

---

## En base de données



- ★ Modélisation en **base de données**
- ★ Définir des **politiques** qui conditionnent l'accès
- ★ Les politiques sont **évaluées** en fonction des attributs de l'utilisateur·rice, de la ressource, du contexte...
- ★ Un **dictionnaire d'accès** prend la décision en fonction des résultats des évaluateurs de politiques
- ★ **Interface d'administration** (type CRUD) des règles
- ★ Parfait pour des besoins spécifiques basées sur des **règles métier**



```
#[ORM\Entity]
class PermissionAttribute
{
    private ?int $id;
    private ?string $name;
    private Collection $rules;
}
```



```
#[ORM\Entity]
class PermissionRule
{
    private ?int $id;
    private ?PermissionAttribute $attribute;
    private ?Team $team;
    private ?Event $event;
    private ?string $role;
    private ?int $minSeniority;
    private bool $checkPlanning = false;
    private ?string $job;
    private ?Skill $skill;
    private ?string $subjectType;
    private ?string $expression;
}
```

## PermissionRule

<b>id</b>	91	92
<b>attribute</b>	42 (event_update)	42 (event_update)
<b>team_id</b>	12	null
<b>event_id</b>	null	null
<b>role</b>	ROLE_ORGANIZER	ROLE_VOLUNTEER
<b>minSeniority</b>	2	4
<b>checkPlanning</b>	true	true
<b>job</b>	null	Head Volunteer
<b>subjectType</b>	App\\Entity\\Event	null
<b>expression</b>	subject.getOrganizer() == user	null



```
final class PermissionVoter extends Voter
{
    public function __construct(
        private readonly PermissionAttributeRepository $attributeRepository,
        private readonly PermissionEngineInterface $engine,
    ) {}

    protected function supports(string $attribute, mixed $subject): bool
    {
        return null !== $this->attributeRepository->findOneBy(['name' => $attribute]);
    }

    protected function voteOnAttribute(string $attribute, mixed $subject, TokenInterface $token): bool
    {
        $permission = $this->attributeRepository->findOneBy(['name' => $attribute]);

        return $this->engine->can($token->getUser(), $permission, $subject);
    }
}
```



```
final class PermissionEngine implements PermissionEngineInterface
{
    public function __construct(
        private PermissionRuleRepository $ruleRepos,
        #[AutowireIterator('permission.handlers')] private iterable $handlers
    ) {}

    public function can(User $user, PermissionAttribute $attribute, mixed $subject): bool
    {
        $rules = $this->ruleRepos->findBy(['attribute' => $attribute]);
        foreach ($rules as $rule) {
            if ($this->check($user, $rule, $subject)) { return true; }
        }
        return false;
    }

    private function check(User $user, PermissionRule $rule, mixed $subject): bool
    {
        foreach ($this->handlers as $handler) {
            if (!$handler->evaluate($user, $rule, $subject)) { return false; }
        }
        return true;
    }
}
```

```
final class EventAssignmentHandler implements RuleHandlerInterface
{
    public function evaluate(User $user, PermissionRule $rule, mixed $subject): bool
    {
        return !$rule->getEvent() || $user->isAssignedTo($rule->getEvent());
    }
}
```

```
final class ExpressionHandler implements RuleHandlerInterface
{
    public function evaluate(User $user, PermissionRule $rule, mixed $subject): bool
    {
        if (!$rule->getExpression()) { return true; }

        $expressionLanguage = new ExpressionLanguage();
        $context = ['user' => $user, 'subject' => $subject, 'now' => new \DateTime()];

        return $expressionLanguage->evaluate($rule->getExpression(), $context);
    }
}
```



# Performances

---

- ✓ Les voters sont **lazy loaded**
- ✓ **Optimisation** des requêtes
  - Indexes
  - Réduire le nombre de requêtes nécessaires pour vérifier les permissions, repenser son modèle
- ✓ **Précalcul** des infos agrégées en base



```
CREATE TRIGGER update_permissions
AFTER INSERT OR UPDATE ON user_teams
FOR EACH ROW
BEGIN
    UPDATE user_permissions SET can_update_team = TRUE WHERE user_id = NEW.user_id;
END;
```

```
CREATE MATERIALIZED VIEW user_planning AS
SELECT u.id AS user_id, p.date AS planning_date, t.name AS team_name
FROM users u
JOIN plannings p ON u.id = p.user_id
JOIN teams t ON u.team_id = t.id;

// mise à jour
REFRESH MATERIALIZED VIEW user_planning;
```



# Performances

---



- ★ Mise en **cache** des attributs supportés par le voter
- ★ Utilisation de **cache de second niveau** avec **Doctrine**
- ★ Mise en cache des **calculs ou décisions**
  - Utilisation d'un **TTL** adéquat
  - Invalidation du cache

# Administration des règles

Interface CRUD pour gérer les rôles de chaque attribut



A large, stylized orange icon resembling a key or a set of interlocking rings, centered on the left side of the slide.

# Exposer Les permissions au frontend

---



# Avec Twig

---

```
{% if is_granted('UPDATE_EVENT', event) %}

    <a href="{{ path('update_event', { event: event.id }) }}">
        Mettre à jour
    </a>

{% endif %}
```



# Client front JavaScript

---

Comment exposer les permissions de la personne par l'API ?

- Le backend reste la **source de vérité**
- Un service collecte et **agrége l'ensemble des permissions possibles pour la personne authentifiée, à un instant T**
  - ◆ Rôles
  - ◆ Permissions spécifiques par attributs
  - ◆ ACL...



# Client front JavaScript

Comment exposer les permissions de la personne par l'API ?

## TOKEN JWT

Encodage des permissions dans le payload du token JWT



## ENDPOINT API

Appel à un endpoint qui renvoie la liste des permissions à jour



FORUM PHP  
PARIS 2024

# Client front JavaScript

---

Comment stocker et gérer les permissions côté front ?

- ★ Stockage en **localStorage**
  - Facile et natif
  - Pas de réactivité
- ★ Utilisation de **contextes et gestionnaires d'état** pour stocker les permissions de façon **dynamique**
  - Réactivité des données
  - Centralisation de la logique d'accès, partage entre les composants
  - Volatilité des données (pas de persistence)
- ★ **Hybride** gestionnaire d'état + persistence dans le **localStorage**





# CASL

---



## ABILITY

Une instance Ability charge et vérifie les autorisations



## RESTRICTIONS

Restreint les accès aux ressources ou actions



## INCRÉMENTAL

Adoptable de manière incrémentale



## PARTAGE

Gère et partage les autorisations entre les composants UI, services...



## ABAC

Implémentation ABAC ou RBAC



## BRIDGES

React, Vue, Angular, Aurelia



# CASL

---



S'appuie sur un modèle déclaratif, basé sur des règles définies sous forme d'objet

```
{ action, subject, fields, conditions }
```

- Récupération des permissions depuis l'**API**
- Chargement d'**objets JSON** dans CASL
- Intégration dans le **store** pour la **réactivité**



```
{  
  "permissions": [  
    {  
      "action": "read",  
      "subject": "Event"  
    },  
    {  
      "action": "update",  
      "subject": "Event",  
      "conditions": { "userId": 123 }  
    },  
    {  
      "action": "delete",  
      "subject": "Event",  
      "conditions": { "role": "admin" }  
    }  
  ]  
}
```

```
export const usePermissionStore = defineStore('permission', {
  state: () => ({ ability: createMongoAbility([]) }),
  actions: {
    setPermissions(permissions) {
      localStorage.setItem('permissions', JSON.stringify(permissions));
      this.ability.update(permissions);
    },
    loadPermissionsFromStorage() {
      const storedPermissions = localStorage.getItem('permissions');
      if (storedPermissions) {
        const permissions = JSON.parse(storedPermissions);
        this.ability.update(permissions);
      }
      const response = await fetchPermissions()
      this.setPermissions(response);
    },
  },
});
```

```
export const usePermissionStore = defineStore('permission', {
  state: () => ({ ability: createMongoAbility([]) }),
  actions: {
    setPermissions(permissions) {
      localStorage.setItem('permissions', JSON.stringify(permissions))
      this.ability.update(permissions)
    },
    loadPermissionsFromStorage() {
      const storedPermissions = localStorage.getItem('permissions')
      if (storedPermissions) {
        const permissions = JSON.parse(storedPermissions)
        this.ability.update(permissions)
      }
      const response = await fetchPermissions()
      this.setPermissions(response)
    }
  }
})
```



## Liste des événements

ID	Nom de l'événement	Début	Fin	Organisateur	Nombre de sponsors	Nombre d'inscrits	État	Actions
1	Forum PHP 2024	10/10/2024	11/10/2024	AFUP	12	800	Brouillon	<button> Voir</button> <button> Éditer</button> <button> Supprimer</button>

```
<div>
    <h1>Liste des événements</h1>
    <button v-if="can('CREATE_EVENT')">Ajouter un événement</button>
</div>

<td>
    <button v-if="can('READ_EVENT')">Voir</button>
    <button v-if="can('UPDATE_EVENT')">Éditer</button>
    <button v-if="can('DELETE_EVENT')">Supprimer</button>
    <button v-if="can('PUBLISH_EVENT')">Publier</button>
</td>
```

# Problématique

---



- ★ On récupère les permissions calculées par l'API
- ★ Les permissions sont **stockées** pendant un certain temps par le frontend
- ★ Si l'admin change les permissions pendant la durée de la session, ou si les propriétés de la personne connectée change, **les permissions ne sont plus à jour**



A large, stylized orange menorah icon is positioned on the left side of the slide. It features a central vertical stem with three circular shapes at the top, each containing a small circle. The base of the stem is flanked by two curved, flame-like shapes. The entire icon is set against a black background.

# Rafraîchissement des permissions

---

- ★ Alternative aux websockets
- ★ Mise en place de Mercure côté API, **pousse les mises à jour vers les clients connectés**
  - Mise à jour des attributs
  - Mise à jour des règles liées aux attributs
  - Mise à jour plus complexe des propriétés de la personne  
(ex : changement d'équipe) si cela fait évoluer ses permissions
- ★ Le client **reçoit les notifications en temps réel dès qu'un changement (de permission) survient**

```
private function sendUpdate(User $user, array $permissions): void
{
    $update = new Update(
        '/users/' . $user->getId() . '/permissions',
        json_encode($permissions),
        true, // private
    );

    $this->hub->publish($update);
}
```

```
export const usePermissionStore = defineStore('permission', {
  state: () => ({ ability: createMongoAbility([]), mercureEventSource: null }),

  actions: {
    setPermissions(permissions) { /* ... */ },
    loadPermissionsFromStorage() {
      /* ... */
      this.subscribeToMercure()
    },
    subscribeToMercure() {
      const url = new URL(`#${API_URL}/.well-known/mercure`)
      url.searchParams.append('topic', `/users/${userId}/permissions`)

      this.mercureEventSource = new EventSource(subscribeURL, { withCredentials: true })
      this.mercureEventSource.onmessage = (event) => {
        const newPermissions = JSON.parse(event.data)
        this.setPermissions(newPermissions)
      }
    },
    unsubscribeToMercure() {}
  }
})
```



## Liste des événements

ID	Nom de l'événement	Début	Fin	Organisateur	Nombre de sponsors	Nombre d'inscrits	État	Actions
1	Forum PHP 2024	10/10/2024	11/10/2024	AFUP	12	800	Brouillon	<button>Voir</button> <button>Éditer</button> <button>Supprimer</button>

[POST] [https://localhost/permission\\_rules](https://localhost/permission_rules)

Response: 201 Created IP: 172.25.0.1 Profiled on: October 4, 2024 at 3:53:18 PM Token: a26edf

#	Time	Memory	Topics	Data	Private	ID	Type	Retry
1	3 ms	4.00 MB	/users/42/permissions	<pre>["action": "DELETE_EVENT"], {"action": "UPDATE_EVENT"}, {"action": "PUBLISH_EVENT"}, {"action": "CREATE_EVENT"}]</pre>	true			



## Liste des événements

ID	Nom de l'événement	Début	Fin	Organisateur	Nombre de sponsors	Nombre d'inscrits	État	Actions
1	Forum PHP 2024	10/10/2024	11/10/2024	AFUP	12	800	Brouillon	<button>Voir</button> <button>Éditer</button> <button>Supprimer</button> <button>Publier</button>

Ajouter un événement





- ★ Le frontend est ainsi notifié des **mises à jour en temps réel**
- ★ L'interface sera **rafraîchie sans avoir à recharger la page**
- ★ Tout est transparent pour l'utilisateur·rice final·e

# MERCI !



@coopTilleuls / les-tilleuls.coop