Deep Dive into the Symfony Security Component

February 22nd 2024 - Confoo - Montréal, Canada

Kodero

kodero

Hugo HAMON

Freelance PHP Consultant @ **KODERO**

Symfony Certified Expert Developer

Former SensioLabs Head of Training

@hhamon / hello@kodero.fr

Quick Introduction

The Symfony Security component provides many tools to secure an application.

From authenticating a user based on their provided credentials to controlling access to resources thanks to a customizable and easily extensible fine grained permissions system.

Extra security features can also be brought thanks to third party community bundles.

Authentication is the process that ensures the user is who they claim to be.

Authorization is the process that ensures the authenticated user have the necessary grants to access the resource or perform some action.

```yaml
# config/packages/security.yaml
security:

    # ———————————————— AUTHENTICATION ———————————————— #

    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'

    providers:
        users_in_memory: { memory: null }

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            provider: users_in_memory


    # ———————————————— AUTHORIZATION ———————————————— #

    role_hierarchy:
        ROLE_ADMIN: [ROLE_USER, ROLE_EDITOR, ROLE_ALLOWED_TO_SWITCH]

    access_control:
        # - { path: ^/admin, roles: ROLE_ADMIN }
        # - { path: ^/profile, roles: ROLE_USER }
```

```
$ symfony console debug:event-dispatcher kernel.request -e prod


Registered Listeners for "kernel.request" Event
===============================================

 ------- ------------------------------------------------------------------------------------ ----------
  Order   Callable                                                                             Priority
 ------- ------------------------------------------------------------------------------------ ----------
  #1      Symfony\Component\HttpKernel\EventListener\DebugHandlersListener::configure()        2048
  #2      Symfony\Component\HttpKernel\EventListener\ValidateRequestListener::onKernelRequest() 256
  #3      Symfony\Component\HttpKernel\EventListener\SessionListener::onKernelRequest()        128
  #4      Symfony\Component\HttpKernel\EventListener\LocaleListener::setDefaultLocale()        100
  #5      Symfony\Component\HttpKernel\EventListener\RouterListener::onKernelRequest()         32
  #6      Symfony\Component\HttpKernel\EventListener\LocaleListener::onKernelRequest()         16
  #7      Symfony\Component\HttpKernel\EventListener\LocaleAwareListener::onKernelRequest()    15
  #8      Symfony\Bundle\SecurityBundle\EventListener\FirewallListener::configureLogoutUrlGenerator() 8
  #9      Symfony\Bundle\SecurityBundle\EventListener\FirewallListener::onKernelRequest()      8
 ------- ------------------------------------------------------------------------------------ ----------
```

Authenticating Users

User

Primary Attributes (required)

- **Identifier** (email, username, uuid, etc.)
- **Roles**

Secondary Attributes (opt-in)

- **Password** (plaintext or encoded / hashed)
- **Salt** (for legacy password hashing algorithms)

```php
interface UserInterface
{
    public function getRoles(): array;

    public function eraseCredentials(): void;

    public function getUserIdentifier(): string;
}


interface PasswordAuthenticatedUserInterface
{
    public function getPassword(): ?string;
}


interface LegacyPasswordAuthenticatedUserInterface extends PasswordAuthenticatedUserInterface
{
    public function getSalt(): ?string;
}
```

```php
#[ORM\Entity(repositoryClass: UserRepository::class)]
class User
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    public function __construct(
        #[ORM\Column(length: 180, unique: true)]
        private string $email,

        #[ORM\Column]
        private string $password,

        #[ORM\Column(type: Types::JSON)]
        private array $roles = [],
    ) {
    }
}
```

```php
#[ORM\Entity(repositoryClass: UserRepository::class)]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    // ...

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getUserIdentifier(): string
    {
        return $this->email;
    }

    public function getRoles(): array
    {
        return \array_values(\array_unique(\array_merge($this->roles, ['ROLE_USER'])));
    }

    public function eraseCredentials(): void
    {
    }

    public function getPassword(): string
    {
        return $this->password;
    }

}
```

User Providers

- **In-Memory**
- **Doctrine Entity**
- **LDAP**
- **Chain**

- *... also, your custom made one!*
- *... or, a third party community one!*

```php
interface UserProviderInterface
{
    public function supportsClass(string $class): bool;

    /**
     * @throws UserNotFoundException
     */
    public function loadUserByIdentifier(string $identifier): UserInterface;

    /**
     * @throws UnsupportedUserException if the user is not supported
     * @throws UserNotFoundException    if the user is not found
     */
    public function refreshUser(UserInterface $user): UserInterface;
}
```

```yaml
# config/packages/security.yaml
security:

    # ...

    providers:

        app_user_provider:
            entity:
                class: App\Entity\User
                property: email

        app_api_user_provider:
            id: App\Security\User\ApiUserProvider

    firewalls:

        # ...

        api:
            # ...
            provider: app_api_user_provider

        admin:
            # ...
            provider: app_user_provider
```

Password Hashers & Auto Upgrade

Supported algorithms

- **Auto** (bcrypt by default)
- **Sodium** (Argon2 + libsodium)
- **PBKDF2**
- **Hash algos** (sha1, sh256, sha512, blowfish, etc.)
- **plaintext** 😅

Password upgrade

- **Auto**
- **From legacy hashing algorithm**

```yaml
# config/packages/security.yaml

security:
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'

when@test:
    security:
        password_hashers:
            # By default, password hashers are resource intensive and take time. This is
            # important to generate secure password hashes. In tests however, secure hashes
            # are not important, waste resources and increase test times. The following
            # reduces the work factor to the lowest possible values.
            Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
                algorithm: auto
                cost: 4 # Lowest possible value for bcrypt
                time_cost: 3 # Lowest possible value for argon
                memory_cost: 10 # Lowest possible value for argon
```

```php
class UserRepository extends ServiceEntityRepository implements PasswordUpgraderInterface
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, User::class);
    }

    public function upgradePassword(PasswordAuthenticatedUserInterface $user, string $newHashedPassword): void
    {
        if (!$user instanceof User) {
            throw new UnsupportedUserException(\sprintf('Instances of "%s" are not supported.', $user::class));
        }

        $user->setPassword($newHashedPassword);
        $this->getEntityManager()->persist($user);
        $this->getEntityManager()->flush();
    }
}
```

Firewalls & Authenticators

- **HTML Form Login**
- **JSON Credentials Login**
- **HTTP Basic**
- **Login Link**
- **Access Token**
- **X.509 Client Certificates**
- **Remote User**

- *... also, your custom made one!*
- *... or, a third party community one!*

```yaml
# config/packages/security.yaml
security:

    # …

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false

        api_documentation:
            pattern: ^/api/doc$
            security: false

        api:
            pattern: ^/api
            stateless: true
            provider: app_api_user_provider
            json_login:
                check_path: ^/api/access-tokens

        admin:
            pattern: ^/admin
            lazy: true
            provider: app_user_provider
            custom_authenticator: App\Security\AdminZoneAuthenticator

        main:
            lazy: true
            provider: app_user_provider
            form_login: ~
            remember_me: ~
            logout: ~
```

```php
interface AuthenticatorInterface
{
    public function supports(Request $request): ?bool;

    public function authenticate(Request $request): Passport;

    public function createToken(Passport $passport, string $firewallName): TokenInterface;

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response;

    public function onAuthenticationFailure(Request $request, AuthenticationException $exception): ?Response;
}


interface InteractiveAuthenticatorInterface extends AuthenticatorInterface
{
    public function isInteractive(): bool;
}


interface AuthenticationEntryPointInterface
{
    public function start(Request $request, ?AuthenticationException $authException = null): Response;
}
```

```php
class HttpBasicAuthenticator implements AuthenticatorInterface, AuthenticationEntryPointInterface
{
    public function __construct(
        private readonly string $realmName,
        private readonly UserProviderInterface $userProvider,
    ) {}

    public function start(Request $request, ?AuthenticationException $authException = null): Response
    {
        $response = new Response();
        $response->headers->set('WWW-Authenticate', sprintf('Basic realm="%s"', $this->realmName));
        $response->setStatusCode(401);

        return $response;
    }

    public function supports(Request $request): ?bool
    {
        return $request->headers->has('PHP_AUTH_USER');
    }

    public function authenticate(Request $request): Passport
    {
        $username = $request->headers->get('PHP_AUTH_USER');
        $password = $request->headers->get('PHP_AUTH_PW', '');

        $userBadge = new UserBadge($username, $this->userProvider->loadUserByIdentifier(...));
        $passport = new Passport($userBadge, new PasswordCredentials($password));

        if ($this->userProvider instanceof PasswordUpgraderInterface) {
            $passport->addBadge(new PasswordUpgradeBadge($password, $this->userProvider));
        }

        return $passport;
    }
}
```

```php
class HttpBasicAuthenticator implements AuthenticatorInterface, AuthenticationEntryPointInterface
{
    // …

    public function createToken(Passport $passport, string $firewallName): TokenInterface
    {
        return new UsernamePasswordToken($passport->getUser(), $firewallName, $passport->getUser()->getRoles());
    }

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response
    {
        return null;
    }

    public function onAuthenticationFailure(Request $request, AuthenticationException $exception): ?Response
    {
        return $this->start($request, $exception);
    }
}
```

Passport

Holding

- **User identifier** (email, username, uuid, etc.)
- **Badges** (password, CSRF, remember-me, etc.)
- **Arbitrary attributes** (metadata, key/value pairs, etc.)

Built-in Implementations

- `Passport`
- `SelfValidatingPassport`

Passport Badges

- **UserBadge**
- PasswordCredentials
- CustomCredentials
- PasswordUpgradeBadge
- RememberMeBadge
- CsrfTokenBadge
- PreAuthenticatedUserBadge

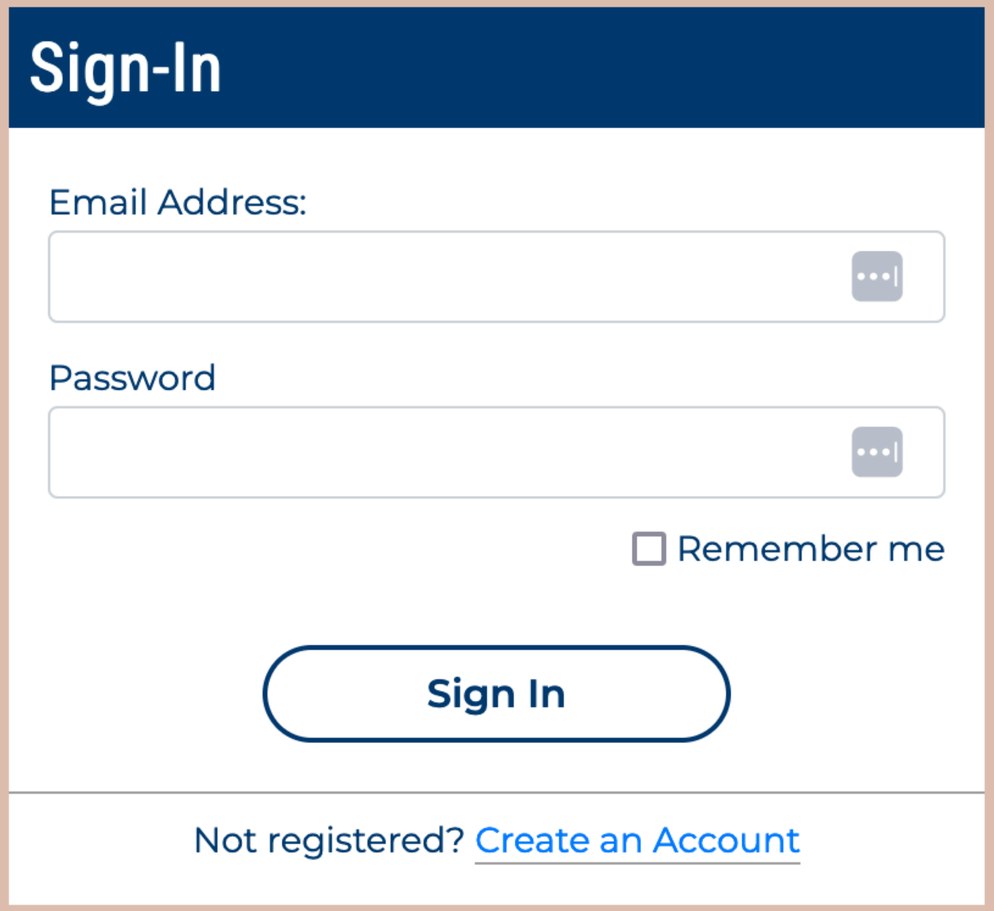
- *... also, add your own if you need to!*

```php
class FormLoginAuthenticator extends AbstractLoginFormAuthenticator
{
    // ...
    private array $options = [
        // ...
        'enable_csrf' => false,
        'csrf_parameter' => '_csrf_token',
        'csrf_token_id' => 'authenticate',
    ];

    // ...

    public function authenticate(Request $request): Passport
    {
        $credentials = $this->getCredentials($request);

        $passport = new Passport(
            userBadge: new UserBadge($credentials['username'], $this->userProvider->loadUserByIdentifier(...)),
            credentials: new PasswordCredentials($credentials['password']),
            badges: [new RememberMeBadge()],
        );

        if ($this->options['enable_csrf']) {
            $passport->addBadge(new CsrfTokenBadge($this->options['csrf_token_id'], $credentials['csrf_token']));
        }

        if ($this->userProvider instanceof PasswordUpgraderInterface) {
            $passport->addBadge(new PasswordUpgradeBadge($credentials['password'], $this->userProvider));
        }

        return $passport;
    }
}
```

Authentication in Practice

Form Login


Esc
F1
F2
F3
F4
Actual Size
Quit
Close
Rulers
Select All
Save
Undo/ Redo
Cut
Hold to constrain proportions
command
Function


Sign-In
Email Address:
Password
Remember me
Sign In
Not registered? Create an Account

```twig
{% extends 'base.html.twig' %}

{% block title %}Sign in{% endblock %}

{% block body %}
<div id="hero">
    <div class="container-fluid">
        <div class="row justify-content-center">
            <div class="col-10 col-sm-8 col-md-6 col-lg-9 col-xl-6" id="main-column">
                <form method="post" action="{{ path('app_login_check') }}”>
                    <div class="row login-form-row justify-content-center">
                        <div class="col-lg-6 form-space">
                            <h3 class="form-title">Sign-In</h3>
                            {% if error %}
                                <div class="alert alert-danger mx-3" role="alert">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
                            {% endif %}

                            <div class="form-group">
                                <label for="_username">Email Address:</label>
                                <input type="email" value="{{ last_username }}" name="_username" id="_username" class="form-control" autocomplete="email" required autofocus>
                            </div>

                            <div class="form-group">
                                <label for="_password">Password</label>
                                <input type="password" name="_password" id="_password" class="form-control" autocomplete="current-password" required>
                            </div>

                            <div class="form-check">
                                <input id="login__remember_me" type="checkbox" name="_remember_me" />
                                <label for="login__remember_me" class="form-check-label">Remember me</label>
                            </div>

                            <div class="text-center" id="login-button-container">
                                <input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}" />
                                <button class="btn btn-login" type="submit">Sign in</button>
                            </div>

                            <div class="text-center" id="create-account-section">
                                <p>Not registered?
                                <a href="{{ path('app_register') }}">Create an Account</a>
                                </p>
                            </div>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

```php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;
use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

final class SecurityController extends AbstractController
{
    #[Route(path: '/login', name: 'app_login', methods: ['GET'])]
    public function login(AuthenticationUtils $authenticationUtils): Response
    {
        return $this->render('security/login.html.twig', [
            'last_username' => $authenticationUtils->getLastUsername(),
            'error' => $authenticationUtils->getLastAuthenticationError(),
        ]);
    }

    #[Route(path: '/login', name: 'app_login_check', methods: ['POST'])]
    public function loginCheck(): void
    {
        throw new \BadMethodCallException('This action should not be invoked.');
    }
}
```

```yaml
# config/packages/security.yaml
security:

    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'

     providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email

     firewalls:
        #...

        main:
            lazy: true
            provider: app_user_provider
            form_login:
                login_path: app_login
                check_path: app_login_check
                username_parameter: _username
                password_parameter: _password
                remember_me: true
                csrf_parameter: _csrf_token
                csrf_token_id: authenticate
                enable_csrf: true
                post_only: true
                use_referer: true
```

Symfony Profiler

search on symfony.com

302 redirect from POST @app_login_check (34c4e7)

GET https://localhost:8000/

Response: 200 OK | Browse referrer URL | **IP:** 127.0.0.1 | **Profiled on:** February 18, 2024 at 6:06:26 PM | **Token:** c7eb28

Last 10 | Latest | Search

- Request / Response
- Performance
- Validator
- Forms
- Exception
- Logs
- Events
- Routing
- Cache
- Translation 13
- Security
- Twig
- Doctrine
- Debug

Security

Token | Firewall | Listeners | Authenticators | Access Decision

Username
john.doe@domain.tld

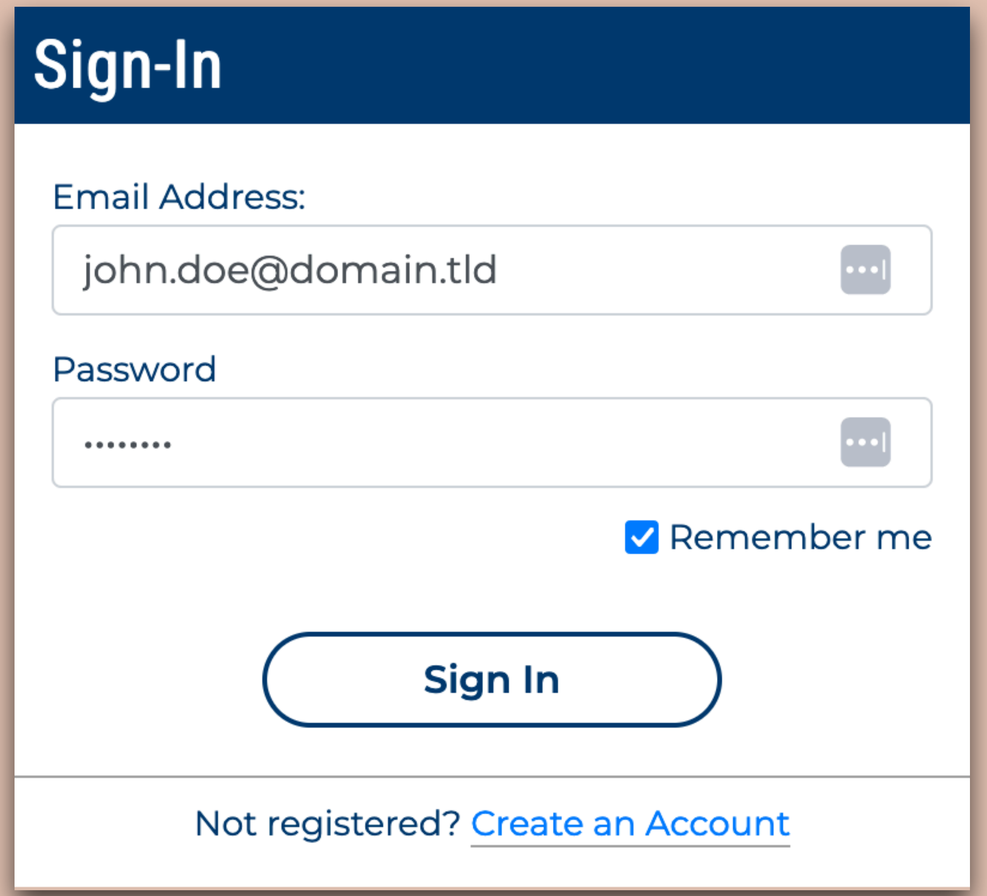
Authenticated
✔

Property	Value
Roles	[▼ "ROLE_USER"]
Inherited Roles	none
Token	Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordToken {#1174 ▼ -user: Proxies__CG__\App\Entity\User {#1045 …} -roleNames: [▶] -attributes: [] -firewallName: "main" }

Remember Me

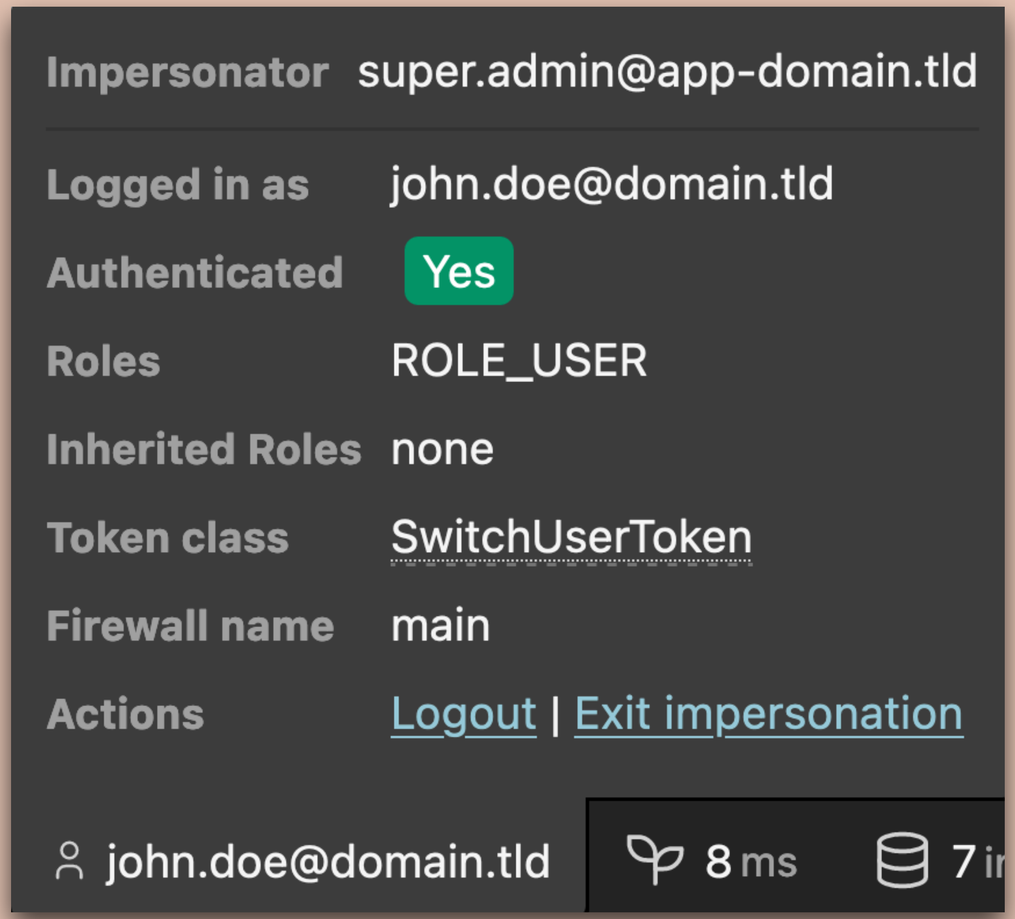

Esc
F1
F2
F3
F4
Actual Size
Quit
Close
Rulers
Select All
Save
Undo/ Redo
Cut
Hold to constrain proportions
command
Function

```yaml
# config/packages/security.yaml
security:

    firewalls:
        #...

        main:
            # ...

            remember_me:
                secret: '%env(APP_SECRET)%'
                lifetime: 604800
                Name: REMEMBERME
                remember_me_parameter: _remember_me
                signature_properties: [password]
```



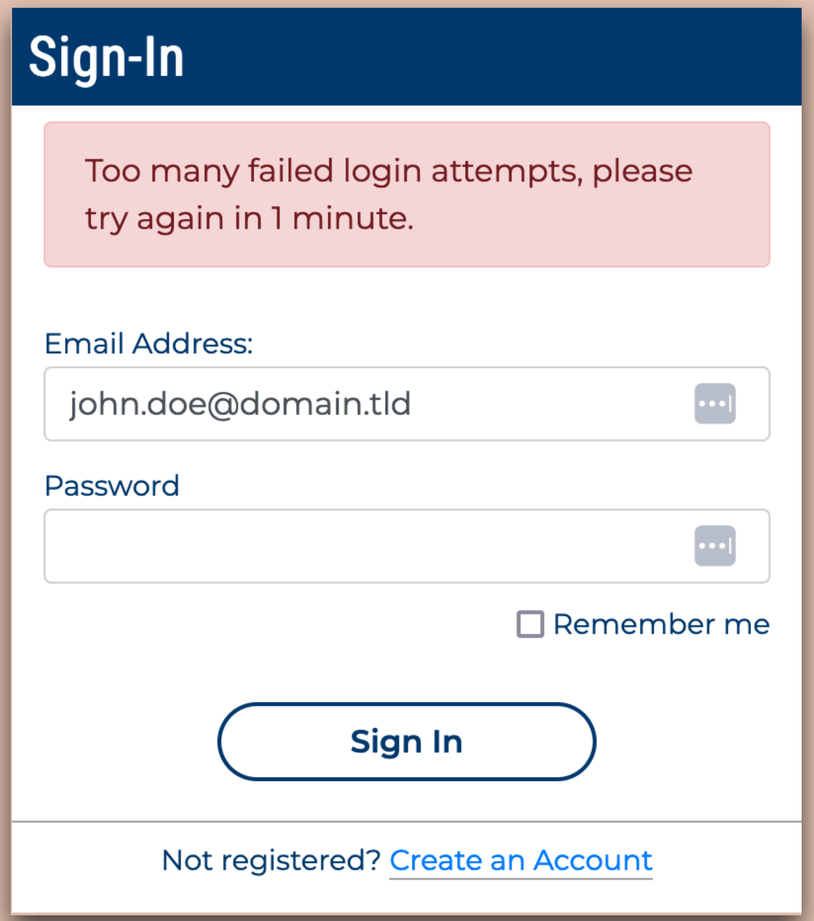
User Impersonation

```yaml
# config/packages/security.yaml

security:

    firewalls:
        #...

        main:
            # ...

            switch_user:
                parameter: _switch_user
                role: ROLE_ALLOWED_TO_SWITCH
```



Login Throttling

```yaml
# config/packages/security.yaml
security:

    #...

    firewalls:
        #...

        main:
            # ...

            login_throttling:
                max_attempts: 3
                interval: '1 minute'
```



Login Link


Esc
F1
F2
F3
F4
Actual Size
Quit
Close
Rulers
Select All
Save
Undo/ Redo
Cut
Hold to constrain proportions
command
Function

```php
final class LoginLinkController extends AbstractController
{
    #[Route('/login-link', name: 'app_login_link_check', methods: ['GET', 'POST'])]
    public function loginLinkCheck(): never
    {
        throw new \BadMethodCallException('This action should not be invoked.');
    }
}
```

```yaml
# config/packages/security.yaml

security:
    # ...
    firewalls:
        # ...
        main:
            # ...
            login_link:
                check_route: app_login_link_check
                signature_properties: ['id']
                lifetime: 300
                max_uses: 3
```

```php
final class LoginLinkController extends AbstractController
{
    #[Route('/auth/login-link', name: 'app_login_link', methods: ['GET', 'POST'])]
    public function requestLoginLink(): Response
    {
        return $this->render('security/request_login_link.html.twig');
    }
}
```

```twig
{% extends 'base.html.twig' %}

{% block body %}
<form method="post" action="{{ path('app_login_link') }}">
    <div>
        <label for="email">Email Address:</label>
        <input type="email" name="email" id="email">
        <button type="submit">Request login link</button>
    </div>
</form>
{%endblock %}
```

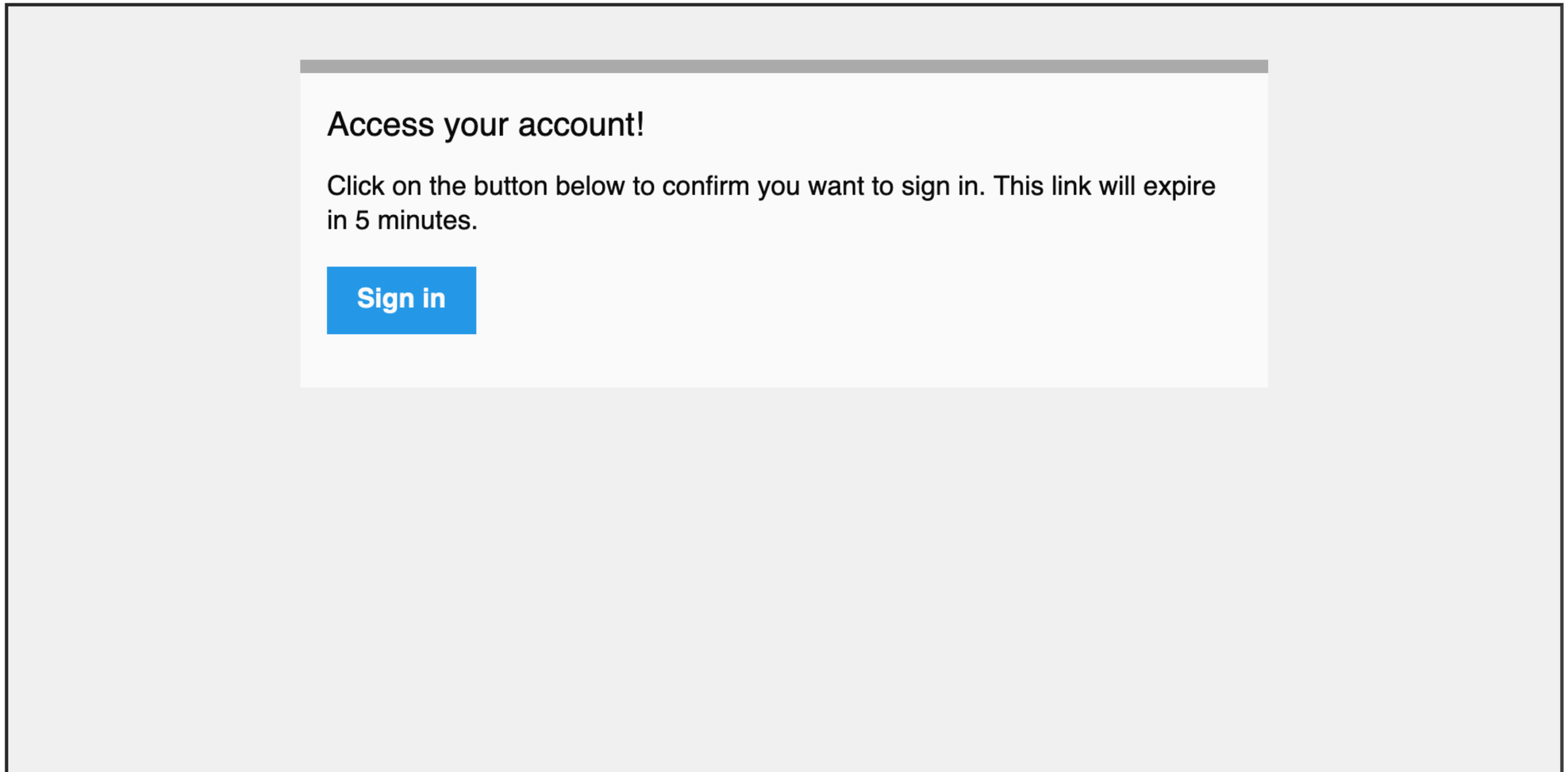
```php
// ...
use Symfony\Component\Notifier\NotifierInterface;
use Symfony\Component\Notifier\Recipient\Recipient;
use Symfony\Component\Security\Http\LoginLink\LoginLinkHandlerInterface;
use Symfony\Component\Security\Http\LoginLink\LoginLinkNotification;

final class LoginLinkController extends AbstractController
{
    public function requestLoginLink(
        LoginLinkHandlerInterface $loginLinkHandler,
        NotifierInterface $notifier,
        UserRepository $userRepository,
        Request $request
    ): Response {
        if ($request->isMethod('POST')) {
            $email = $request->getPayload()->get('email');

            if (!$user = $userRepository->findOneBy(['email' => $email])) {
                throw new BadRequestHttpException();
            }

            $notifier->send(
                new LoginLinkNotification(
                    $loginLinkHandler->createLoginLink($user),
                    'Access your account!',
                ),
                new Recipient($user->getEmail()),
            );

            return $this->render('security/login_link_sent.html.twig');
        }

        return $this->render('security/request_login_link.html.twig');
    }
}
```

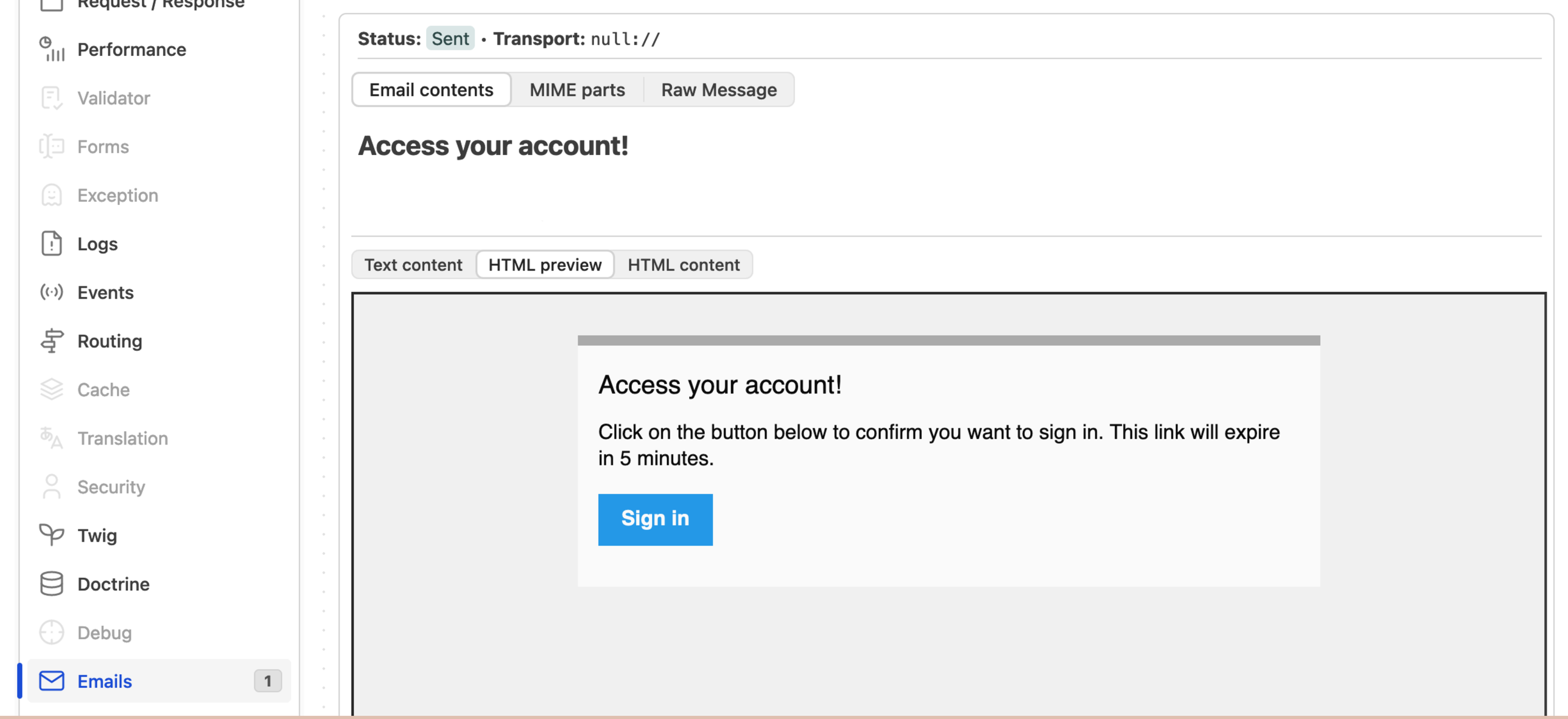
Status: Sent • **Transport:** `null://`

Email contents | MIME parts | Raw Message

Access your account!

Text content | HTML preview | HTML content



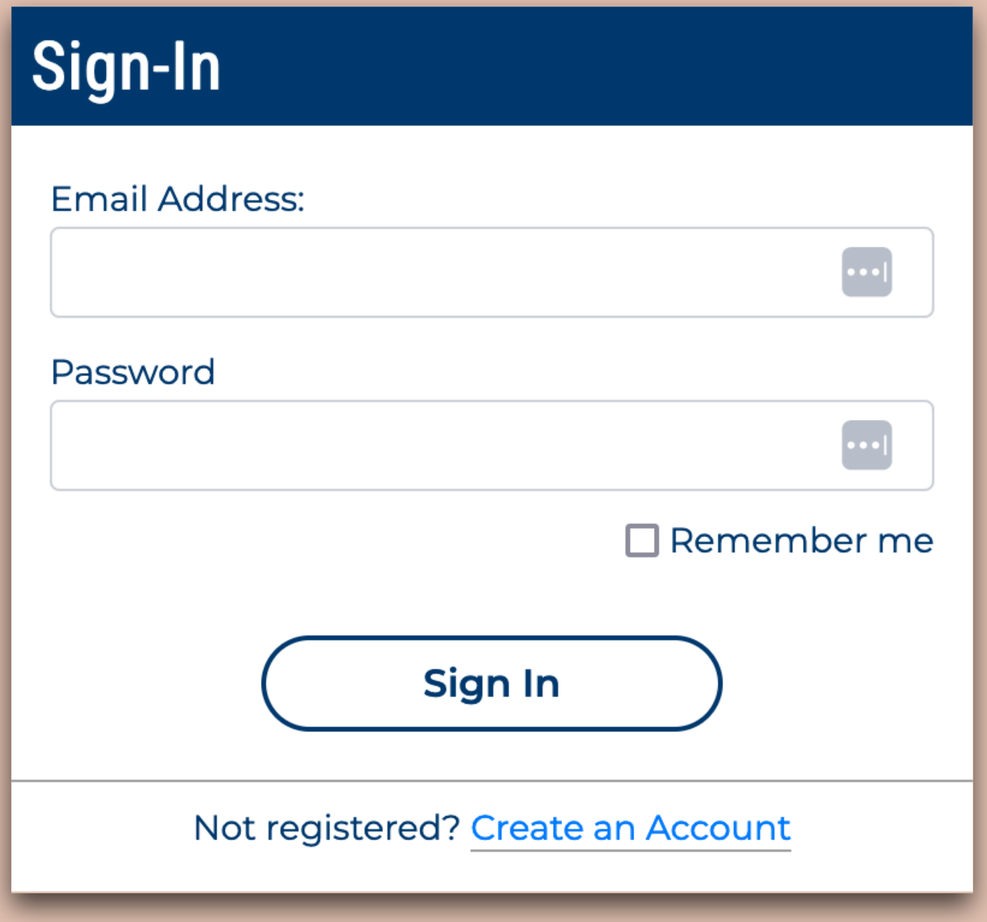


https://127.0.0.1:8000/login-link?user=user@email.com&expires=1708612912&hash=8bklF1yFdsSDY_KMMYuw8ddAt8zW4GrkkrnA_deeB84~eIQeE_ygqOyCWLamFKPdLBh7ab0dBmMqQ0Fu6OmJXUQ~

Logout


Esc
F1
F2
F3
F4
Actual Size
Quit
Close
Rulers
Select All
Save
Undo/ Redo
Cut
Hold to constrain proportions
command
Function

```yaml
# config/packages/security.yaml
security:

    #…

    firewalls:
        #...

        main:
            # ...

            logout:
                path: 'app_logout'
                target: 'app_login'
```



Custom Authenticator

```yaml
# config/packages/security.yaml
security:

    #…

    firewalls:
        #...

        main:
            # ...

            custom_authenticators:
                - App\Security\MyFirstCustomAuthenticator
                - App\Security\MyOtherCustomAuthenticator
```

Current Authenticated User

```php
namespace Symfony\Bundle\SecurityBundle;

class Security implements AuthorizationCheckerInterface
{
    public function getUser(): ?UserInterface
    {
        if (!$token = $this->getToken()) {
            return null;
        }

        return $token->getUser();
    }

    public function getToken(): ?TokenInterface
    {
        return $this->container->get('security.token_storage')->getToken();
    }

    public function getFirewallConfig(Request $request): ?FirewallConfig
    {
        return $this->container->get('security.firewall.map')->getFirewallConfig($request);
    }

    public function login(UserInterface $user, ?string $authenticatorName = null, ?string $firewallName = null, array $badges = []): ?Response;
    {
        // ...
    }

    public function logout(bool $validateCsrfToken = true): ?Response
    {
        // ...
    }
}
```

```php
use App\Entity\User;
use Symfony\Bundle\SecurityBundle\Security;

// ...

final readonly class ReservationController extends AbstractController
{
    public function __construct(
        // ...
        private ReservationRepository $repository,
        private Security $security,
    ) {
    }

    #[Route(path: '/reservations', name: 'reservations')]
    public function index(): Response
    {
        $user = $this->security->getUser();

        \assert($user instanceof User);

        $reservations = $this->repository->findReservationsByUser($user);

        // ...
    }
}
```

```php
use App\Entity\User;

final readonly class ReservationController extends AbstractController
{
    public function __construct(
        // ...
        private ReservationRepository $repository,
    ) {
    }

    #[Route(path: '/reservations', name: 'reservations')]
    public function index(#[CurrentUser] User $user): Response
    {
        $reservations = $this->repository->findReservationsByUser($user);

        // ...
    }
}
```

```
<div class="user-card">
    {% if app.user is defined %}
        Hello {{ app.user.displayName }}!
    {% else %}
         <a href="/login">Sign-in</a>
    {% endif %}
</div>
```

Controlling Resources Access

Roles & Role Hierarchy

- Combining access roles
- Prevent code duplication
- Ease role management & maintenance

```
# config/packages/security.yaml
security:

    role_hierarchy:
        ROLE_ADMIN: [ROLE_EDITOR, ROLE_ALLOWED_TO_SWITCH]
        ROLE_SALES_MANAGER: [ROLE_SALES, ROLE_ALLOWED_TO_SWITCH]
        ROLE_SALES: [ROLE_SALES]
        ROLE_EDITOR: [ROLE_USER]
```

Access Controls

- Secure parts of an application
- Use regular expression to match URL patterns
- Granular matching on request information

```
# config/packages/security.yaml

security:

    access_control:
        - { path: ^/(rate-offers|settings|auth)$, roles: ROLE_USER }
        - { path: ^/admin, roles: ROLE_ADMIN }
        - { path: ^/api, roles: ROLE_API }
        - { path: ^/, roles: PUBLIC_ACCESS }
```

Checking User's Authorizations

```php
namespace Symfony\Bundle\SecurityBundle;

class Security implements AuthorizationCheckerInterface
{
    // ...

    public function isGranted(mixed $attributes, mixed $subject = null): bool
    {
        return $this->container
            ->get('security.authorization_checker')
            ->isGranted($attributes, $subject);
    }
}
```

```php
final class ReservationController
{
    public function __construct(
        // ...
        private AuthorizationCheckerInterface $security,
    ) {
    }

    #[Route(path: '/reservations/{id<\d+>/cancel')]
    public function cancel(Request $request, Reservation $reservation): Response
    {
        $issueRefund = $request->request->getBoolean('refund');

        $this->reservationService->cancel($reservation);

        if ($issueRefund && $this->security->isGranted('ROLE_SALES_MANAGER')) {
            $this->refundService->refund($reservation);
        }

        // ...
    }
}
```

```php
final class ReservationController extends AbstractController
{
    #[Route(path: '/reservations/{id<\d+>/cancel')]
    public function cancel(Request $request, Reservation $reservation)
    {
        $issueRefund = $request->request->getBoolean('refund');

        $this->reservationService->cancel($reservation);

        if ($issueRefund && $this->isGranted('ROLE_SALES_MANAGER')) {
            $this->refundService->refund($reservation);
        }

        // ...
    }
}
```

```php
final class ReservationController extends AbstractController
{
    #[Route(path: '/reservations/{id<\d+>/cancel')]
    public function cancel(Request $request, Reservation $reservation)
    {
        $this->denyAccessUnlessGranted('ROLE_SALES')

        // ...
    }
}
```

Leveraging Attributes for Expressiveness

```php
final class ReservationController extends AbstractController
{
    #[IsGranted('ROLE_SALES')]
    #[Route(path: '/reservations/{id<\d+>}/cancel')]
    public function cancel(Request $request, Reservation $reservation)
    {
        $issueRefund = $request->request->getBoolean('refund');

        $this->reservationService->cancel($reservation);

        if ($issueRefund && $this->isGranted('ROLE_SALES_MANAGER')) {
            $this->refundService->refund($reservation);
        }

        // ...
    }
}
```

```php
class MyController extends AbstractController
{
    #[IsGranted(new Expression('is_granted("ROLE_ADMIN") or is_granted("ROLE_MANAGER")'))]
    public function show(): Response
    {
        // ...
    }

    #[IsGranted(new Expression(
        '"ROLE_ADMIN" in role_names or (is_authenticated() and user.isSuperAdmin())'
    ))]
    public function edit(): Response
    {
        // ...
    }
}
```

```php
class PostController extends AbstractController
{
    #[IsGranted(
        attribute: new Expression('user === subject'),
        subject: new Expression('args["post"].getAuthor()'),
    )]
    public function index(Post $post): Response
    {
        // ...
    }
}
```

```php
class ReservationController extends AbstractController
{
    #[IsGranted(
        attribute: new Expression(
            'user === subject["owner"] and subject["reservation"].isUpcoming()'
        ),
        subject: [
            'owner' => new Expression('args["reservation"].getOwner()'),
            'reservation',
        ],
    )]
    public function edit(Reservation $reservation): Response
    {
        // ...
    }
}
```

Voters

Input

- **An attribute** (i.e. a permission name or a role)
- **The authentication token**
- **An arbitrary subject** (usually a PHP object)

Output

- **Grant access**
- **Deny access**
- **Abstain from voting**

```php
interface VoterInterface
{
    public const ACCESS_GRANTED = 1;
    public const ACCESS_ABSTAIN = 0;
    public const ACCESS_DENIED = -1;

    public function vote(
        TokenInterface $token,
        mixed $subject,
        array $attributes
    ): int;
}
```

Implementing Custom Voters

```php
class ReservationController extends AbstractController
{
    public function cancel(Reservation $reservation): Response
    {
        $this->denyAccessUnlessGranted(
            ReservationVoter::CANCEL,
            $reservation,
        );

        // ...
    }
}
```

```php
final class ReservationVoter extends Voter
{
    public const CANCEL = 'RESERVATION_CANCEL';
    public const RESCHEDULE = 'RESERVATION_RESCHEDULE';

    protected function supports(string $attribute, mixed $subject): bool
    {
        return \in_array($attribute, [self::CANCEL, self::RESCHEDULE])
            && $subject instanceof Reservation;
    }

    protected function voteOnAttribute(string $attribute, mixed $subject, TokenInterface $token): bool
    {
        \assert($subject instanceof Reservation);

        $user = $token->getUser();

        if (!$user instanceof User) {
            return false;
        }

        return match ($attribute) {
            self::CANCEL => $this->canCancel($user, $subject),
            self::RESCHEDULE => $this->canReschedule($user, $subject),
            default => false,
        };
    }
}
```

```php
final class ReservationVoter extends Voter
{
    private function canCancel(User $user, Reservation $reservation): bool
    {
        if ($user->isSuperAdmin() || $user->isSalesManager()) {
            return true;
        }

        return $reservation->isOwnedBy($user)
            && $reservation->isUpcoming();
            && ! $reservation->isUpcomingWithin('2 days');
    }

    private function canReschedule(User $user, Reservation $subject): bool
    {
        // ...
    }
}
```

Show voter details

2 **DENIED** RESERVATION_CANCEL

```
App\Entity\Reservation {#12903 ▼
  -id: 1
  -user: App\Entity\User {#6704 …}
  -bookingOffer: App\Entity\BookingOffer {#5883 …}
  -dateOfBooking: DateTime @1606003200 {#4140 ▶}
  -adultNumber: 1
  -childNumber: 0
  -isPaidFor: false
  -bankTransferDate: null
  -bankTransferTitle: "M60clO7uZJKzEwA"
  -destination: null
  -totalCost: null
}
```

"Scheb\TwoFactorBundle\Security\Authorization\Voter\TwoFactorInProgressVoter"	ACCESS ABSTAIN
"App\Security\Voter\ReservationVoter"	ACCESS DENIED

Hide voter details

Show voter details

2 **GRANTED** RESERVATION_CANCEL

```
App\Entity\Reservation {#612 ▼
  -id: 1
  -user: App\Entity\User {#432 …}
  -bookingOffer: App\Entity\BookingOffer {#615 …}
  -dateOfBooking: DateTime @1606003200 {#601 ▶}
  -adultNumber: 1
  -childNumber: 0
  -isPaidFor: false
  -bankTransferDate: null
  -bankTransferTitle: "M60clO7uZJKzEwA"
  -destination: null
  -totalCost: null
}
```

"Scheb\TwoFactorBundle\Security\Authorization\Voter\TwoFactorInProgressVoter"	ACCESS ABSTAIN
"App\Security\Voter\ReservationVoter"	ACCESS GRANTED

Hide voter details

```php
class ReservationController extends AbstractController
{
    #[IsGranted(
        attribute: ReservationVoter::CANCEL,
        subject: new Expression('args["reservation"]'),
    )]
    public function cancel(Reservation $reservation): Response
    {
        return $this->render('reservations/cancel.html.twig', [
            'reservation' => $reservation,
        ]);
    }
}
```

Going Further

Third Party Bundles

Two Factor

- **SchebTwoFactorBundle**

https://symfony.com/bundles/SchebTwoFactorBundle

Webauthn

- **WebauthnBundle**

https://webauthn-doc.spomky-labs.com/

JWT

- **LexikJWTAuthenticationBundle**

https://symfony.com/bundles/LexikJWTAuthenticationBundle

Third Party Bundles

Misc

SymfonyCastsResetPasswordBundle

https://github.com/SymfonyCasts/reset-password-bundle

SymfonyCastsVerifyEmailBundle

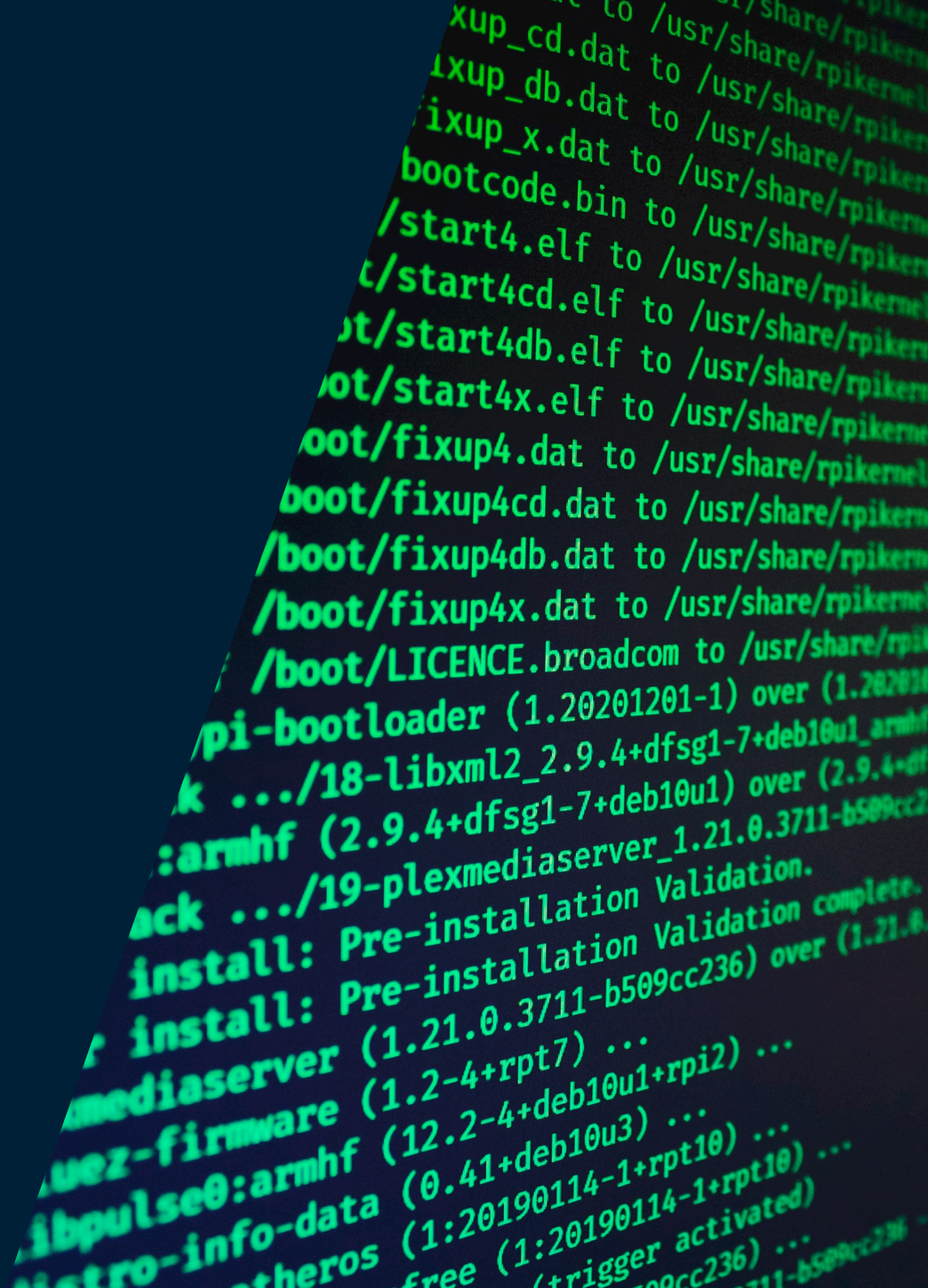
https://github.com/SymfonyCasts/verify-email-bundle

Security Events

- AuthenticationEvent
- AuthenticationSuccessEvent
- AuthenticationTokenCreatedEvent
- CheckPassportEvent
- InteractiveLoginEvent
- LazyResponseEvent
- LoginFailureEvent
- LoginSuccessEvent
- LogoutEvent
- SwitchUserEvent
- TokenDeauthenticatedEvent
- VoteEvent (dev only)

Useful Console Commands

```
› config:dump-reference

› debug:config
› debug:firewall

› make:auth
› make:registration-form
› make:reset-password
› make:security-form-login
› make:voter

› security:hash-password
```



Thank you for listening!

kodero

@hhamon / hello@kodero.fr