Security

Security best practices for Derafu Auth

Anonymous · 09/09/2026

Security

Environment Variables

Always use environment variables for sensitive configuration:

```
# □ Good
KEYCLOAK_CLIENT_SECRET=your-secret-here

# □ Bad
'client_secret' => 'your-secret-here'
```

HTTPS Configuration

Enable HTTPS in production:

```
# Development.
KEYCLOAK_SECURE_COOKIES=false
KEYCLOAK_HTTP_VERIFY=false

# Production.
KEYCLOAK_SECURE_COOKIES=true
KEYCLOAK_HTTP_VERIFY=true
```

Session Security

```
// Secure session configuration.
$config = new AuthConfiguration([
    'session_lifetime' => 3600,
    'secure_cookies' => true,
]);

// Additional session security that can be set.
ini_set('session.cookie_httponly', true);
```

```
ini_set('session.cookie_samesite', 'Lax');
ini_set('session.use_strict_mode', true);
```

CSRF Protection

The package automatically validates the `state` parameter to prevent CSRF attacks.

Token Security

Never store tokens in client-side storage:

```
// ✅ Good - Server-side session.
$session->set('access_token', $accessToken);

// ❌ Bad - Never expose to client.
return new JsonResponse(['token' => $accessToken]);
```

Authorization

```
// Validate role format.
private function isValidRole(string $role): bool
{
    return preg_match('/^[a-zA-Z0-9_]+$/', $role) === 1;
}

// Check role hierarchy.
private function hasRole(array $userRoles, string $requiredRole): bool
{
    $roleHierarchy = [
        'super_admin' => ['admin', 'user'],
        'admin' => ['user'],
    ];

    if (in_array($requiredRole, $userRoles)) {
        return true;
    }

    foreach ($userRoles as $userRole) {
        if (isset($roleHierarchy[$userRole]) &&
            in_array($requiredRole, $roleHierarchy[$userRole])) {
            return true;
        }
    }

    return false;
```

```
}
```

Error Handling

Don’t expose sensitive information in production:

```
public function handle(ServerRequestInterface $request, Throwable $exception):
ResponseInterface
{
    $isProduction = getenv('APP_ENV') === 'production';

    if ($exception instanceof AuthenticationException) {
        return new JsonResponse([
            'error' => 'Authentication failed',
            'code' => 'UNAUTHORIZED'
        ], 401);
    }

    return new JsonResponse([
        'error' => $isProduction ? 'Internal server error' : $exception->getMessage(),
        'code' => 'INTERNAL_ERROR'
    ], 500);
}
```

Security Headers

```
class SecurityHeadersMiddleware
{
    public function process(ServerRequestInterface $request, RequestHandlerInterface
$handler): ResponseInterface
    {
        $response = $handler->handle($request);

        return $response
            ->withHeader('X-Content-Type-Options', 'nosniff')
            ->withHeader('X-Frame-Options', 'DENY')
            ->withHeader('X-XSS-Protection', '1; mode=block')
            ->withHeader('Referrer-Policy', 'strict-origin-when-cross-origin')
            ->withHeader('Content-Security-Policy', "default-src 'self'")
            ->withHeader('Strict-Transport-Security', 'max-age=31536000;
includeSubDomains');
    }
}
```

Keycloak Security

Client Configuration

1. **Access Type**: `confidential`.
2. **Valid Redirect URIs**: Only your application URLs.
3. **Web Origins**: Your application domain.
4. **Client Authentication**: Enabled.

Realm Settings

1. **Password Policy**: Strong password requirements.
2. **Brute Force Detection**: Enabled.
3. **Session Timeout**: Configured appropriately.
4. **SSL Required**: `external` or `all`.

Last updated on 09/09/2026