SensioLabs

Symfony

Hacking & Extending Symfony2
SF2C3

the SENSIOLABS network

The Security Component and Bundle

Provides **Authentication** and **Authorization** mechanisms

Authentication

ensures the user is who he claims to be.

Authorization grants or denies someone to perform an action.

Some key concepts. . .

Tool	Meaning
encoder	An encoder is used for hashing and comparing passwords
provider	A provider knows how to create users
firewall	A firewall defines the authentication mechanisms for the whole application or for just a part of it
access_control	Access control rules secure parts of your application with roles

Encoding and checking passwords

An **encoder** is used for hashing and comparing passwords

The PasswordEncoderInterface interface

```
namespace Symfony\Component\Security\Core\Encoder;

interface PasswordEncoderInterface
{
    function encodePassword($raw, $salt);

    function isPasswordValid($encoded, $raw, $salt);
}
```

Configuring encoders

```
# app/config/security.yml

security:
    encoders:
        Sensio\UserBundle\User: plaintext
        Sensio\UserBundle\User: sha512
        Sensio\UserBundle\User:
            algorithm: sha512
            encode_as_base64: true
            iterations: 5000
        Sensio\UserBundle\User:
            id: my.custom.encoder.service.id
```

Encoding a password

```
$user = new Sensio\UserBundle\User();

$factory = $this->get('security.encoder_factory');
$encoder = $factory->getEncoder($user);
$pwd = $encoder->encodePassword('secret', '^H4x0r$');

$user->setPassword($pwd);
```

Checking the validity of the password

```
$encoder->isPasswordValid('e5e[...]ca5', 'secret', '^H4x0r$');
```

Fetching users on a user provider

A **provider** knows how to retrieve and create users.

Built-in user providers in Symfony SE

Type	Meaning
memory	Fetches users from a configuration file (security.yml)
chain	Fetches users by chaining multiple providers
entity	Fetches users from a Doctrine entity model
propel	Fetches users from a Propel active record model

```php
interface UserProviderInterface
{
    /**
     * Loads the user for the given username.
     *
     * @return UserInterface
     */
    function loadUserByUsername($username);

    /**
     * Refreshes the user properties like credentials.
     *
     * @return UserInterface
     */
    function refreshUser(UserInterface $user);

    /**
     * Whether this provider supports the given user class
     *
     * @return Boolean
     */
    function supportsClass($class);
}
```

Storing users in memory

```
security:
    providers:
        administrators:
            memory:
                users:
                    jsmith:
                        password: secret
                        roles: [ 'ROLE_USER' ]
                    hhamon:
                        password: azerty
                        roles: [ 'ROLE_TRAINER' ]
                    fabpot:
                        password: qwerty
                        roles: ['ROLE_TRAINER', 'ROLE_ADMIN' ]
```

Representing a user in the security layer

A **user** object stores **credentials** and associated **roles**.

The UserInterface interface

```
interface UserInterface
{
    function getRoles();

    function getPassword();

    function getSalt();

    function getUsername();

    function eraseCredentials();
}
```

The AdvancedUserInterface interface

```
interface AdvancedUserInterface extends UserInterface
{
    function isEnabled();

    function isCredentialsNonExpired();

    function isAccountNonLocked();

    function isAccountNonExpired();
}
```

Managing users' roles

Roles are strings but they can be any objects of type RoleInterface.

The RoleInterface interface

```
interface RoleInterface
{
    /**
     * Returns the role name.
     *
     * @return string The role name
     */
    function getRole();
}
```

The roles hierarchy

```
# app/config/security.yml
security:
    role_hierarchy:

        ROLE_ADMIN:       ROLE_USER
        ROLE_TRAINER:     ROLE_USER
        ROLE_SUPERADMIN:
            - ROLE_USER
            - ROLE_ADMIN
            - ROLE_ALLOWED_TO_SWITCH
```

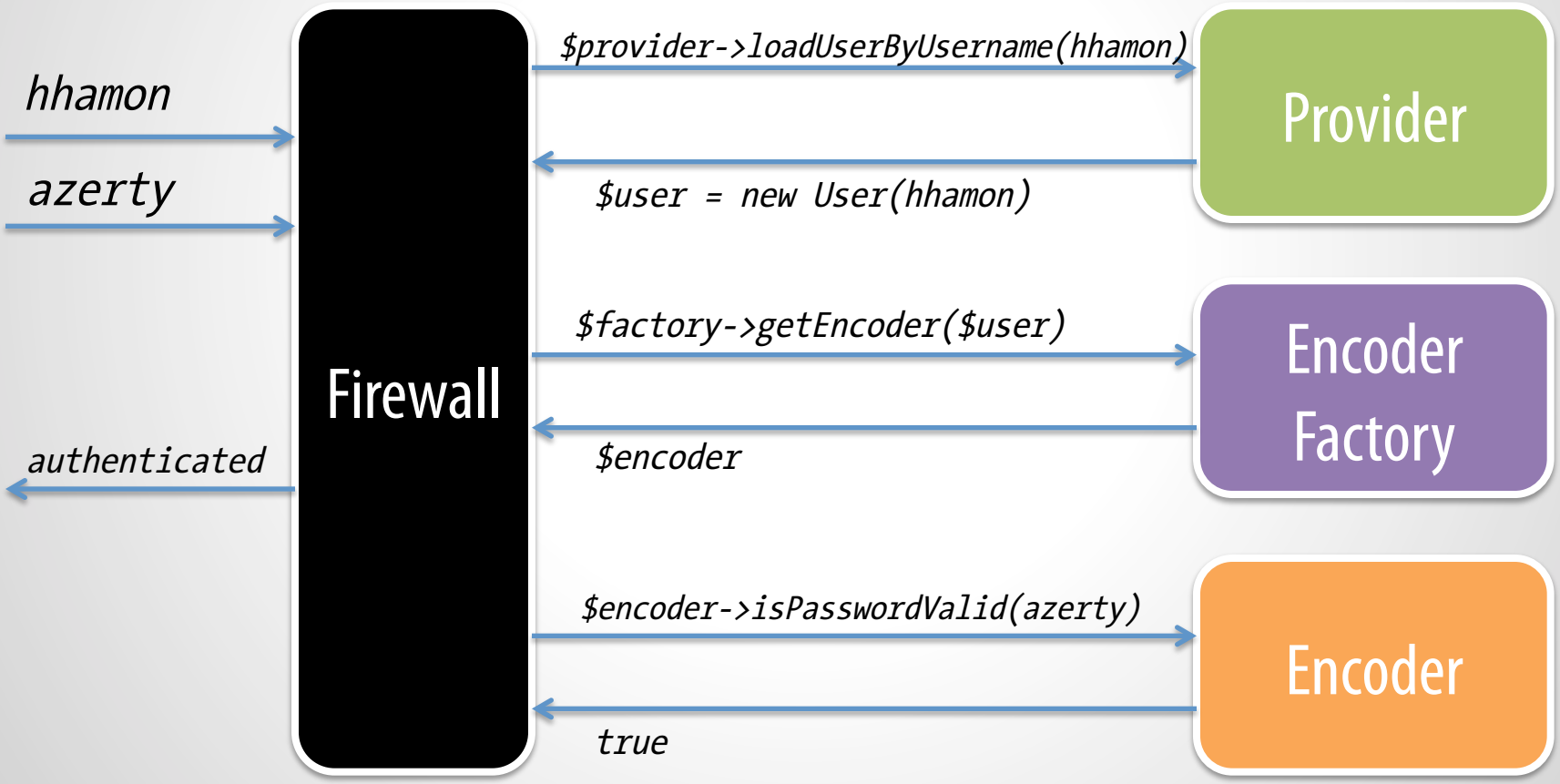
Authenticating against a firewall

The **firewall** determines whether or not the user needs to be authenticated.

Supported authentication firewalls

Type	Meaning
http-basic	Use basic HTTP authentication
http-digest	Use basic HTTP authentication with a hashed password
x.509	Use a x.509 certificate
form-based	Use a simple web form to ask for the login and password credentials

Authentication worflow

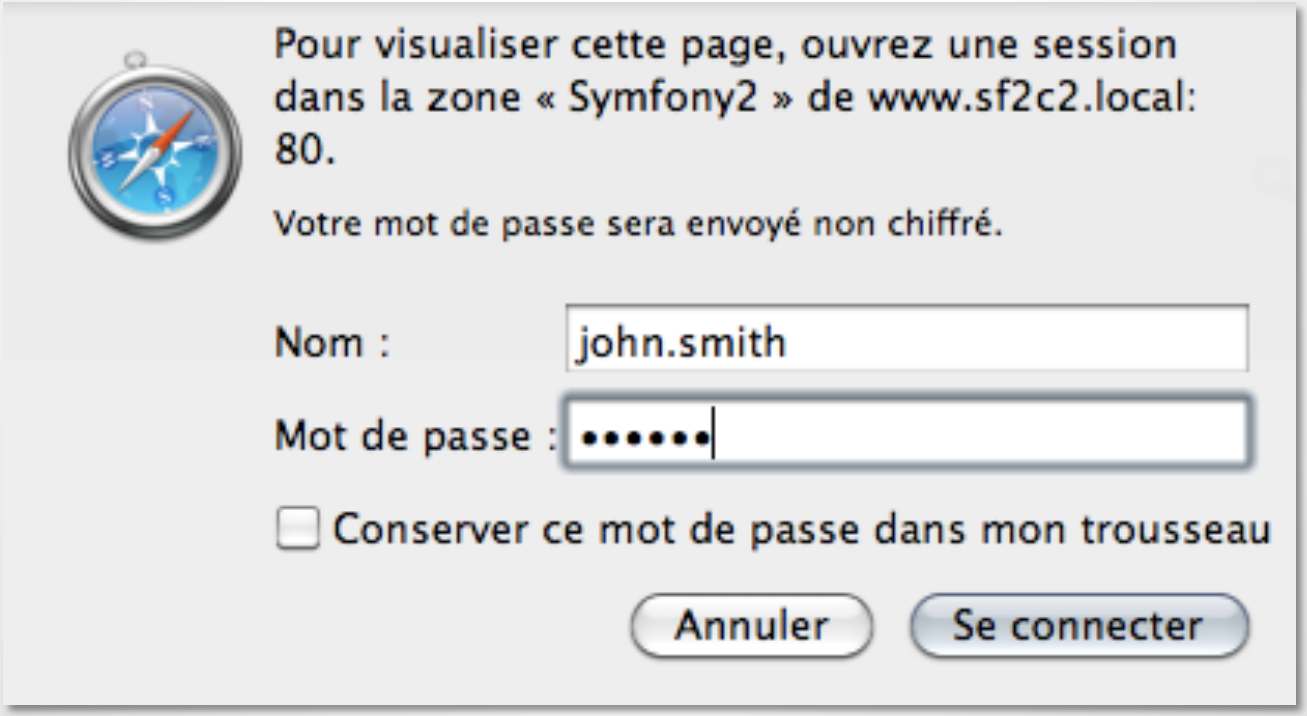


Configuring an HTTP Authentication

```
# app/config/security.yml
security:

    # ...
    firewalls:
        admin:
            provider:   administrators
            pattern:    ^/admin
            http_basic:
                realm:   "Secured Admin Area"
```

Authenticating with HTTP Basic



Authenticating with a login form

```
# app/config/security.yml
security:
    # ...
    firewalls:
        admin:
            provider:                           administrators
            pattern:                            ^/admin
            form_login:
                check_path:                     login_check
                login_path:                     signin
                default_target_path:            admin
                always_use_default_target_path: true
```

The login form

Adding routes for authentication

```
# app/config/routing.yml

login_check:
    path:           /admin/auth
    methods:        [ POST ]

signout:
    path:           /admin/logout

signin:
    path:           /login
    defaults:       { _controller: SensioUserBundle:Security:login }
```

```php
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Security\Core\SecurityContext;

class SecurityController extends Controller
{
    public function loginAction(Request $request)
    {
        $session = $request->getSession();

        if ($name  = $session->get(SecurityContext::LAST_USERNAME)) {
            $session->remove(SecurityContext::LAST_USERNAME);
        }

        if ($error = $session->get(SecurityContext::AUTHENTICATION_ERROR)) {
            $session->remove(SecurityContext::AUTHENTICATION_ERROR);
        }

        return $this->render(
            'SensioUserBundle:Security:signin.html.twig',
            array('last_username' => $name, 'error' => $error)
        );
    }
}
```

```
{% extends 'SensioUserBundle::layout.html.twig' %}

{% block content %}
    <h1>Login</h1>

    {% if error %}
        <div class="error">{{ error.message }}</div>
    {% endif %}

    <form action="{{ path("login_check") }}" method="post">
        <div>
            <label for="username">Username</label>
            <input type="text" id="username"
                   name="_username" value="{{ last_username }}" />
        </div>
        <div>
            <label for="password">Password</label>
            <input type="password" id="password" name="_password" />
        </div>

        <button type="submit">login</button>
    </form>
{% endblock %}
```

Allowing logout capability

```
# app/config/security.yml
security:
    # ...
    firewalls:
        admin:
            # ...
            logout:
                path:   signout
                target: signin
```

Allowing anonymous authentication

```
# app/config/security.yml
security:
    # ...
    firewalls:
        frontend:
            # ...
            pattern:   ^/
            anonymous: true
```

Accessing the user from a controller

```
$user = $this->getUser();
```

Accessing the user from a template

```
{{ app.user.username }}
```

Allowing admin users to switch context

```
# app/config/security.yml
security:
    # ...
    firewalls:
        frontend:
            # ...
            pattern:   ^/
            switch_user: true
```

Allowing admin users to switch context

```
security:
    providers:
        administrators:
            memory:
                users:
                    superadmin:
                        password: superman
                        roles:
                            - 'ROLE_ADMIN'
                            - 'ROLE_ALLOWED_TO_SWITCH'
                    ...
```

Switching to another security user

http://my.domain.com/app_dev.php/admin?_switch_user=hhamon

Forcing the security token

```
$token = new UsernamePasswordToken(
    'hhamon',
    'p4SSwOrD',
    'administrators',
    array('ROLE_ADMIN')
);

$this->get('security.context')->setToken($token);
```

Supported authentication tokens

Class	Meaning
AnonymousToken	Token for anonymous users.
RememberMeToken	Used when authenticating with a remember me cookie.
PreAuthenticatedToken	Used when requests are already pre-authenticated.
UsernamePasswordToken	Used when authenticating with a username and password.
PersistentToken	Used when authenticating with a cookie.

Controlling access to application with ACLs

ACLs & ACEs

An Access Control List defines the set of authorizations a user has on the application domain objects instances.

An Access Control Entry defines an authorization rule for a given domain object or set of objects.

Access Control Entries Scopes

Scope	Meaning
Class	These entries apply to all objects with the same class.
Object	These entries apply on one specific object.
Class-Field	These entries apply to all objects with the same class, but only to a specific field of the objects.
Object-Field	These entries apply to a specific object, and only to a specific field of that object.

Controlling access to application with voters

The Access Decision Manager

The Access Decision Manager service is responsible of granting or denying the access to an end user according to a given strategy.

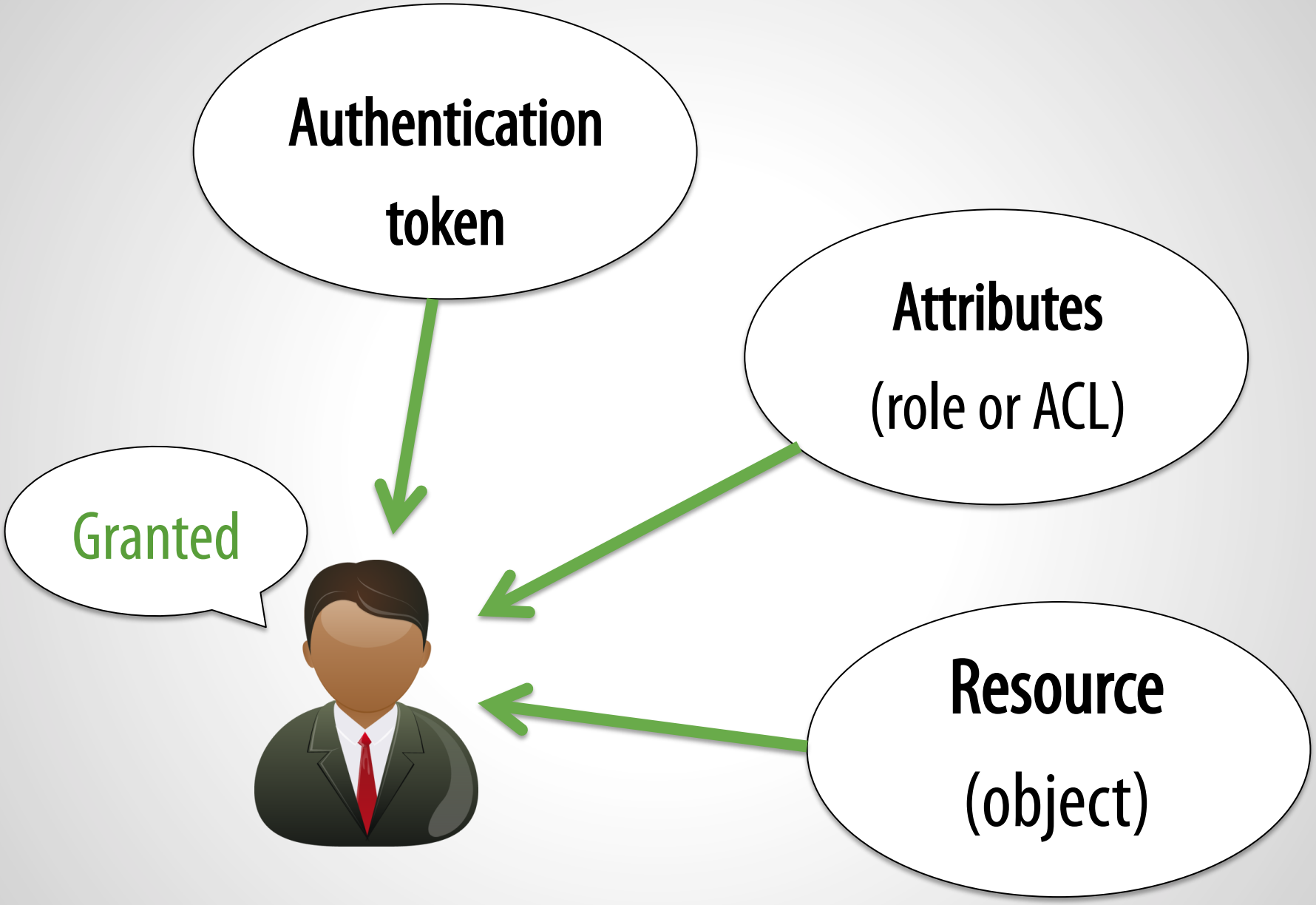
What's the role of a strategy

The strategy decides whether or not the user is granted to access a resource. The decision is delegated to a list of voters.

The strategy decides whether or not the user is granted to access a resource. The decision is delegated to a list of voters.

A voter is an object that decides whether or not a user, represented by its authentication token, can access to a secured resource for a given context.

The voter can grant the access, deny it, or abstain if it can't determine a decision.


Authentication token
Attributes (role or ACL)
Granted
Resource (object)

ROLE_TRAVELER

Attribute

Access token

Flight
XY1234

Resource

Allowed to
board the flight

Customs officer

ROLE_TERRORIST

Attribute

Access token

Flight
XY1234

Resource

Security, catch that guy now!

Customs officer

Supported strategies

Unanimous

Affirmative (default)

Consensus

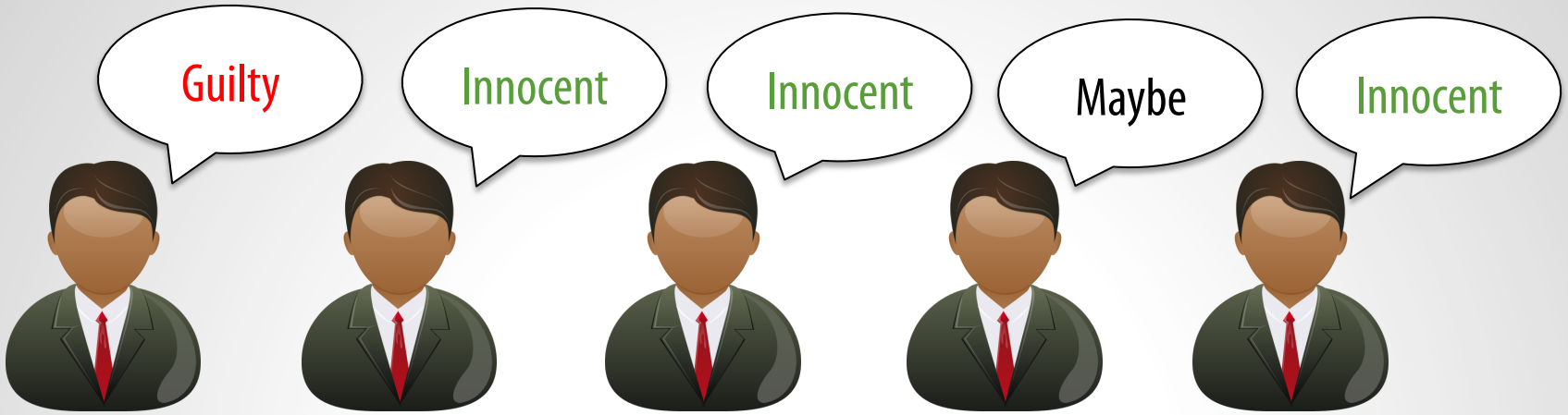
The Affirmative Strategy

The affirmative strategy grants the access if at least one voter among all others returns an affirmative decision.

The affirmative strategy grants the access if at least one voter among all others returns an affirmative decision.

The Consensus Strategy

The consensus strategy grants the access if the majority of all voters return an affirmative decision.



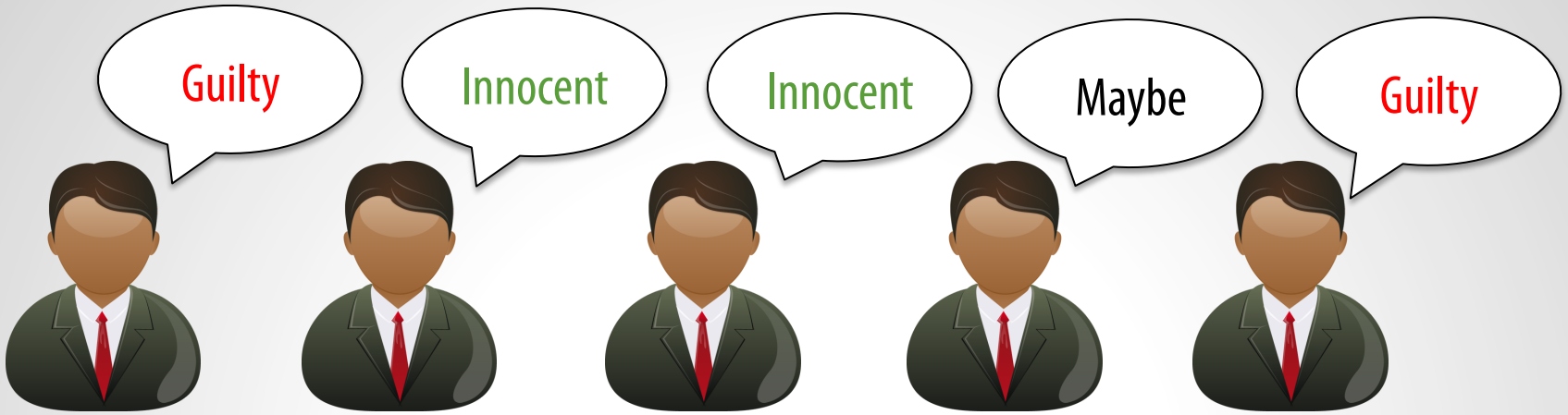
The consensus strategy grants the access if the majority of all voters returns an affirmative decision.

The consensus strategy grants the access if the majority of all voters returns an affirmative decision.

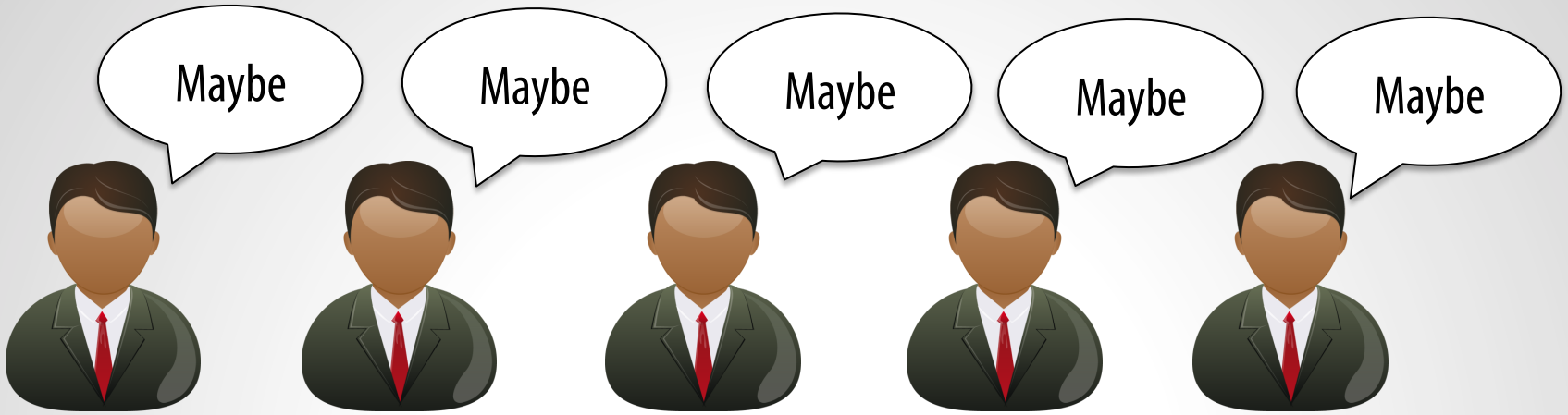
The Consensus Strategy – Special Cases

If the number of affirmative decisions equals the number of denied decisions, then the access is granted by default.

If all voters abstain from voting, then the access is denied by default.



If the number of affirmative decisions equals the number of denied decisions, then the access is granted by default.

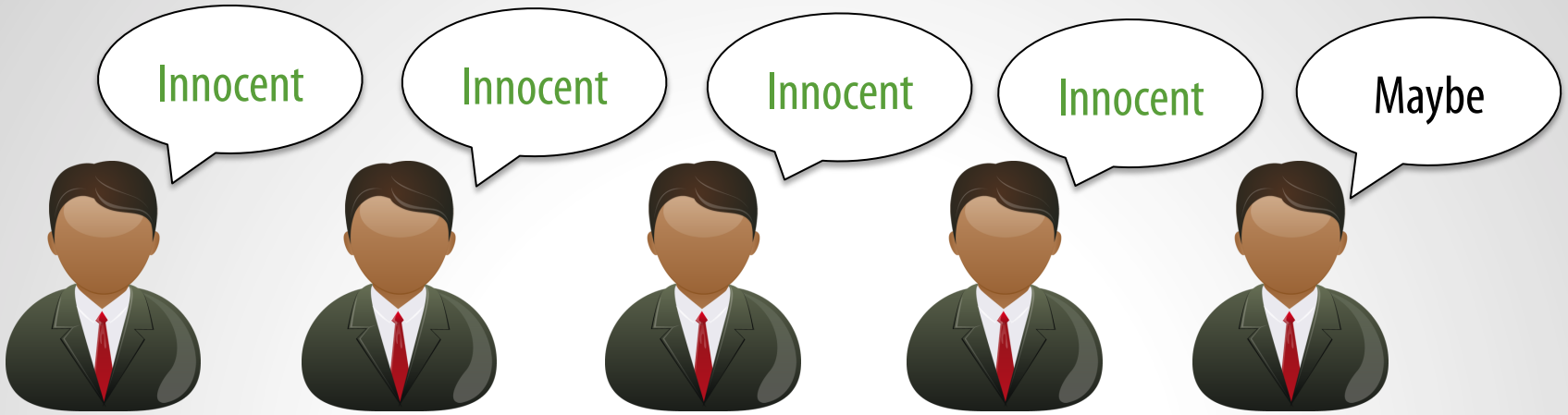


If all voters abstain from voting, then the access is denied by default.

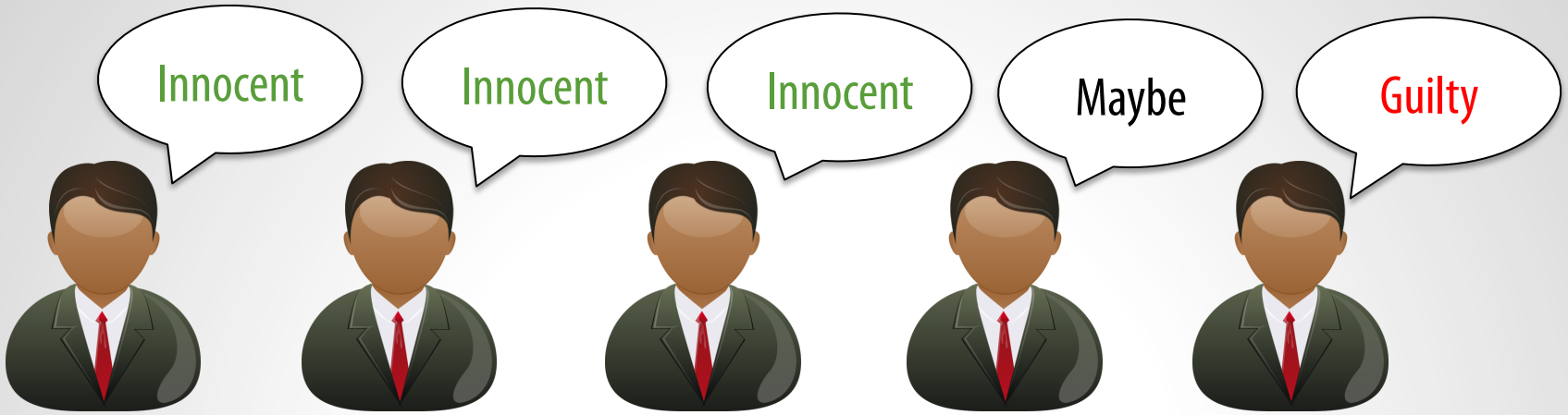
The Unanimous Strategy

The unanimous strategy grants the access if all voters return an affirmative decision.

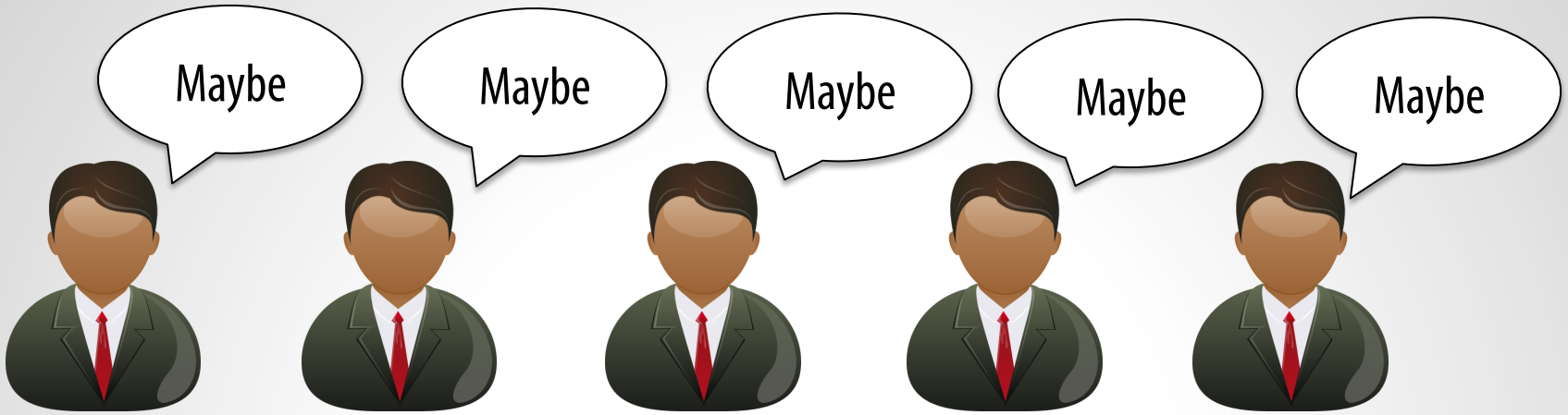
If all voters abstain from voting, then the access is denied by default.



The unanimous strategy grants the access if all voters return an affirmative decision.



The unanimous strategy grants the access if all voters return an affirmative decision.



If all voters abstain from voting, then the access is denied by default.

A voter must implement the VoterInterface. Its vote() method returns the voting decision.

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

    function supportsAttribute($attribute);

    function supportsClass($class);

    function vote(TokenInterface $token, $object, array $attributes);
}
```

```php
namespace Symfony\Component\Security\Core\Authorization\Voter;

use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;

class RoleVoter implements VoterInterface
{
    public function vote(TokenInterface $token, $object, array $attributes)
    {
        $result = VoterInterface::ACCESS_ABSTAIN;
        foreach ($attributes as $attribute) {
            if (!$this->supportsAttribute($attribute)) {
                continue;
            }

            $result = VoterInterface::ACCESS_DENIED;
            foreach ($this->extractRoles($token) as $role) {
                if ($attribute === $role->getRole()) {
                    return VoterInterface::ACCESS_GRANTED;
                }
            }
        }

        return $result;
    }
}
```

Registering a new voter: use case

« The application manages SVN repositories. We want to check if the connected user has the commit authorization on a given path of a repository. »

We need to register a new voter that check if the connected has the commit authorization by looking in the SVN Authz file.

The SVN Authz file

```
# /var/www/projects/training/etc/svn.authz

[/]
fabien.potencier = rw
hugo.hamon = rw

[/trunk]
marc.weistroff = rw

[/branches/v2]
alexandre.salome = rw
jenkins.ci = r
```

```php
namespace Acme\DemoBundle\Security\Authorization\Voter;

use Symfony\Component\Security\Core\Authorization\Voter\VoterInterface;
use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;

class SvnCommitVoter implements VoterInterface
{
    private $permissions;

    public function __construct($authz)
    {
        if (!file_exists($authz)) {
            throw new \InvalidArgumentException('Invalid SVN Authz file: '. $authz);
        }

        $this->permissions = parse_ini_file($authz, true);
    }

    public function supportsAttribute($attribute)
    {
        return 'commit' === $attribute;
    }

    public function supportsClass($class)
    {
        return true;
    }
```

```php
class SvnCommitVoter implements VoterInterface
{
    // ...
    public function vote(TokenInterface $token, $object, array $attributes)
    {
        // $object is a relative path on the repository
        $username = $token->getUser()->getUsername();

        while ('/' !== $object) {
            // Check if the user is granted to commit on the path
            if (isset($this->permissions[$object][$username])) {
                $acls = $this->permissions[$object][$username];
                if (isset($acls[1]) && 'w' === $acls[1]) {
                    return VoterInterface::ACCESS_GRANTED;
                }
            }

            $object = pathinfo($object, PATHINFO_DIRNAME);
        }

        return VoterInterface::ACCESS_DENIED;
    }
}
```

Declaring the voter as a service in the DIC

```
<!-- src/Acme/AcmeBundle/Resources/config/services.xml -->
<service
  id="security.access.svn_commit_voter"
  class="Acme\DemoBundle\Security\Authorization\Voter\SvnCommitVoter"
  public="false">
    <argument>/var/projects/training/etc/svn.authz</argument>
    <tag name="security.voter" />
</service>
```

Try the voter with a secured action

```php
namespace Acme\DemoBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\Security\Core\Exception\AccessDeniedException;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

class RepositoryController extends Controller
{
    /**
     * @Route("/repository/commit"),
     * @Template()
     */
    public function commitAction()
    {
        if (!$this->get('security.context')->isGranted('commit', '/trunk/foo/bar')) {
            throw new AccessDeniedException();
        }

        return array();
    }
}
```

Training Department

SensioLabs Training

92-98 Boulevard Victor Hugo

92 115 Clichy Cedex

FRANCE

Phone: +33(0)140 998 211

Email: trainings@sensiolabs.com

symfony.com - trainings.sensiolabs.com