How does the authentication work ?

First of all : The Security Component

The Security component provides a complete security system for your web application. It ships with facilities for authenticating using HTTP basic authentication, interactive form login, but also allows you to implement your own authentication strategies. Furthermore, the component provides ways to authorize authenticated users based on their roles, and it contains an advanced ACL system.

Source : https://symfony.com/doc/3.4/components/security.html

In our case, the application was a MVP (Minimum Viable Product), some of functionnalities were light, but the authentication system was in symfony standards.

Let's see what we had

The security system is configured in app/config/security.yml. This is the security.yml file before changes.

With this file, you can configure whatever you like for security in all your application, password encoders, user provider, firewalls, url access control, roles …

```
security:
    encoders:
        AppBundle\Entity\User: bcrypt

    providers:
        doctrine:
            entity:
                class: AppBundle:User
                property: username

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

        main:
            anonymous: ~
            pattern: ^/
            form_login:
                login_path: login
                check_path: login_check
                always_use_default_target_path: true
                default_target_path: /
            logout: ~


    access_control:
        - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/users, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/, roles: ROLE_USER }
```

How does security.yml work?

```
security:
#What we're talking about, the security component

    encoders:
        AppBundle\Entity\User: bcrypt

#Here, you can choose the encoder you want for users passwords, for
#example, bcrypt, sha12 or if you want none of it you can also choose
#plaintext.

    providers:
        doctrine:
            entity:
                class: AppBundle:User
                property: username

#This is the part in which you configure where your users are loaded.
#Here users are stored in database and loaded by username property.
#Source : https://symfony.com/doc/3.4/security/entity_provider.html

    firewalls:

#This is the big part of the security configuration. This is where you
#will configure access to urls
#You can configure several firewalls, like here:

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

#The dev firewall isn't important, it just makes sure that Symfony's
#development tools - which live under URLs
#like /_profiler and /_wdtaren't blocked by your security.
#All other URLs will be handled by the main firewall (no pattern key
#means it matches all URLs).
#Source : https://symfony.com/doc/3.4/security.html#initial-security-yml-setup-authentication

        main:
            anonymous: ~
            pattern: ^/
            form_login:
                login_path: login
                check_path: login_check
                always_use_default_target_path: true
                default_target_path: /
            logout: ~

#At this point, we create a form authentication.
#We give a route for the login method accessible via /login, which will
#call a loginAction method in the SecurityController. We will not need
#to implement a controller for the URL / login_check  since the firewall
#will automatically intercept and transform any form submitted to this
#URL(see Annex SecurityController.php).
#The always_use_default_target_path ignore the previously requested URL
#and always redirect to the default page define at default_target_path.
#The logout key is not configured, by default, when users log out from
#any firewall, their sessions are invalidated. This means that logging
#out from one firewall automatically logs them out from all.
#Source : https://symfony.com/doc/3.4/security/form_login_setup.html
```

```
    access_control:
        - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/users, roles: IS_AUTHENTICATED_ANONYMOUSLY }#
        - { path: ^/, roles: ROLE_USER }

#The most basic way to secure part of your application is to secure an
#entire URL pattern. We saw this earlier, where anything matching the
#regular expression with the path keys, theses paths require the roles
#key role.
#Here you can see that the / path requires ROLE_USER, it means that all
#the application need at least an ROLE_USER access, otherwise, you will
#redirect on the login page.
#The IS_AUTHENTICATED_ANONYMOUSLY means all users. For the login path
#it's ok because if you secure this part, you will be in a redirect loop
(see Source 2).
#But for the route /users, this is not normal, and this point is part of
the changes to make on the project, what we will see now.
#Source : https://symfony.com/doc/3.4/security.html#securing-url-patterns-access-control
#https://symfony.com/doc/current/security/form_login_setup.html#be-sure-the-login-page-isn-t-secure-redirect-loop
#https://symfony.com/doc/3.4/security.html#checking-to-see-if-a-user-is-logged-in-is-authenticated-fully
#https://symfony.com/doc/3.4/security/access_control.html
```

Annex SecurityController.php

```
// src/AppBundle/Controller/SecurityController.php
use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

public function loginAction(Request $request, AuthenticationUtils
$authUtils)
{
        // get the login error if there is one
        $error = $authUtils->getLastAuthenticationError();

        // last username entered by the user
        $lastUsername = $authUtils->getLastUsername();

        return $this->render('security/login.html.twig', array(
          'last_username' => $lastUsername,
          'error'         => $error,
        ));
}
```

Improve authentication against specifications

The specifications required several improvements, but for authentication, it is mainly this task that is important:

Only users with the administrator role should be able to access the user management pages.

We have seen above, to modify the access to particular urls, it was necessary to modify the file security.yml and more precisely the part access_control. We will simply add the role ROLE_ADMIN user on the path /url.

This role ROLE_ADMIN does not exist yet, it is necessary to define its hierarchy with respect to the role ROLE_USER used previously by adding a section role_hierarchy to our file security.yml

```
security:
    encoders:
        AppBundle\Entity\User: bcrypt

#Adding hierarchy of roles. In the above configuration, users
#with ROLE_ADMIN role will also have the ROLE_USER role.
    role_hierarchy:
        ROLE_ADMIN: ROLE_USER


    providers:
        doctrine:
            entity:
                class: AppBundle:User
                property: username

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

        main:
            anonymous: ~
            pattern: ^/
            form_login:
                login_path: login
                check_path: login_check
                always_use_default_target_path: true
                default_target_path: /
            logout: ~


    access_control:
        - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }

#change the role required to access urls / users by changing ROLE_USER
by ROLE_ADMIN.
        - { path: ^/users^, roles: ROLE_ADMIN }

#IS_AUTHENTICATED_FULLY isn't a role, but it kind of acts like one, and
#every user that has successfully logged in will have this.
#Source : https://symfony.com/doc/3.4/security.html#checking-to-see-if-a-user-is-logged-in-is-authenticated-fully
- { path: ^/, roles: IS_AUTHENTICATED_FULLY }
```

Once done, we will create a role attribute in our entity AppBundle\Entity\User, so that our security can know if a user has the right role (ROLE_ADMIN or ROLE_USER) or not to access urls.

```php
// src/AppBundle/Entity/User.php
namespace AppBundle\Entity;
use …;

class User implements UserInterface
{
        // … attributes

        // adding « roles » attribute
        private $roles;

        // … getters-setters

        // adding getter-setter for roles
        public function getRoles()
        {
                return $this->roles;
        }

        public function setRoles($roles)
        {
                $this->roles = $roles;
                return $this;
        }

}
```

NB : Roles has been added to the UserTypeForm as requested in the specifications.

Once an user can have a particular role, our firewall will know if the user in question has the role to access a certain url, here /admin.

For even more security, it has been added on the project a security annotation on the methods of the UserController, to restrict these methods only to the role ROLE_ADMIN.

Source : https://symfony.com/doc/3.4/security.html#securing-controllers-and-other-code

```
// src/AppBundle/Controller/UserController.php
namespace AppBundle\Controller;

// use …
// add this …
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;



/**
 * @Security("is_granted('ROLE_ADMIN')")
 */
class UserController extends Controller
{
        // … methods
}
```

Our system after modification

We now have an authentication system that allows:

- Allow access to the application only if you are authenticated
- Be redirected to the login form if you request a URL without being authenticated
- Have different user roles
- ROLE_ADMIN user roles can access urls / users while ROLE_USER users will be denied access

Thanks for reading !