

# Elytron: BCrypt

# Elytron: BCRYPT

Nous allons reprendre la configuration elytron mais pour maintenant utiliser des mots de passe hashés en BCRYPT à la place du clearText.

# Elytron: BCrypt

Pour cela on va devoir stocker nos utilisateurs avec un mot de passe hashé , on va devoir donc comprendre comment l'algorithme BCrypt fonctionne.

# Elytron: BCRYPT

En plus de la transformation classique d'un algorithme de hashage, bcrypt répète son processus un certain nombre de fois (round ou itération)

The screenshot shows the CyberChef web application interface. The browser address bar displays the URL: `gchq.github.io/CyberChef/#recipe=Bcrypt(1)&input=Y291Y291`. The interface is divided into three main sections:

- Operations:** A sidebar on the left containing a list of operations. "Bcrypt" is selected and highlighted in blue. Other visible operations include "bcry", "Bcrypt parse", "Bcrypt compare", "Blowfish Decrypt", "Blowfish Encrypt", "CipherSaber2 Decrypt", "CipherSaber2 Encrypt", "Favourites", "Data format", "Encryption / Encoding", "Public Key", "Arithmetic / Logic", and "Networking".
- Recipe:** The central area shows the "Bcrypt" recipe selected. It includes a "Rounds" input field with the value "1".
- Input:** The input field contains the text "coucou".
- Output:** The output field displays the result of the encryption: `$2a$04$2DcECAwddJ2j5qHoY1ALFeKwQKCxQ7M0xfWfV5yq05hoKz/wLtS7C`.

Additional UI elements include a "Download CyberChef" button, a "Last build: 2 months ago" status message, and icons for saving, deleting, and pausing the recipe.

# Elytron: BCRYPT

De plus l'implémentation d'elytron rajoute un sel (un mot rajouté à chaque mot de passe avant ou après permettant d'éviter les attaques rainbow table).

Nous devons donc avoir 3 informations pour pouvoir générer notre hash:

- Un mot de passe (différent pour chaque utilisateur)
- Un salt (de préférence différent pour chaque utilisateur)
- Un nombre d'itération

# Elytron: BCRYPT

Le mot de passe est choisi lors de la création de l'utilisateur  
Pour le sel nous allons laissé la librairie d'elytron s'en chargé,  
mais on pourrait prendre le nom de l'utilisateur accolé à un  
chiffre par exemple.

On choisira 31 itérations (plus le nombre d'itérations est grand ,  
plus le hash est sécurisé, mais plus l'authentification prendra du  
temps,l'implementation bcrypt de base d'elytron n'autorise de  
31 itérations max).

Le hash et le salt seront stocké encodé en base64.

# Elytron: BCRYPT

En utilisant la doc officielle, nous allons créer un endpoint pour ajouter un admin à notre base de donnée, avant de mettre en place bcrypt

[https://github.com/wildfly/wildfly/blob/main/docs/src/main/asciidoc/\\_elytron/components/JDBC\\_Security\\_Realm.adoc#using-a-hashed-password-representation](https://github.com/wildfly/wildfly/blob/main/docs/src/main/asciidoc/_elytron/components/JDBC_Security_Realm.adoc#using-a-hashed-password-representation)

```

1  @GET
2      @Path("/test")
3      public void test(){
4          PasswordFactory passwordFactory = null;
5          try {
6              passwordFactory = PasswordFactory.getInstance(BCryptPassword.ALGORITHM_BCRYPT, ELYTRON_PROVIDER);
7          } catch (NoSuchAlgorithmException e) {
8              // TODO Auto-generated catch block
9              e.printStackTrace();
10         }
11         String mdp="Lepetitchatestmort";
12         int iterationCount = 31;
13
14         byte[] salt = new byte[BCryptPassword.BCRYPT_SALT_SIZE];
15         SecureRandom random = new SecureRandom();
16         random.nextBytes(salt);
17
18         IteratedSaltedPasswordAlgorithmSpec iteratedAlgorithmSpec = new IteratedSaltedPasswordAlgorithmSpec(iterationCount, salt, mdp.toCharArray());
19         EncryptablePasswordSpec encryptableSpec = new EncryptablePasswordSpec(mdp.toCharArray(), iteratedAlgorithmSpec);
20
21         BCryptPassword original = null;
22         try {
23             original = (BCryptPassword) passwordFactory.generatePassword(encryptableSpec);
24         } catch (InvalidKeySpecException e) {
25             // TODO Auto-generated catch block
26             e.printStackTrace();
27         }
28
29         byte[] hash = original.getHash();
30
31         Encoder encoder = Base64.getEncoder();
32         System.out.println("Encoded Salt = " + encoder.encodeToString(salt));
33         System.out.println("Encoded Hash = " + encoder.encodeToString(hash));
34         Utilisateur u=new Utilisateur();
35         u.setPseudo("seb2");
36         u.setSalt(encoder.encodeToString(salt));
37         u.setPassword(encoder.encodeToString(hash));
38         em.persist(u);
39
40     }

```




Le endpoint /test créé un utilisateur seb2 avec le hash bcrypt et le salt


|   | id<br>[PK] bigint | date_naissan<br>timestamp w | email<br>character va | nom<br>character vai | password<br>character varying(255) | prenom<br>character vai | pseudo<br>character vai | telephone<br>character vai | salt<br>character varying(255) |
|---|-------------------|-----------------------------|-----------------------|----------------------|------------------------------------|-------------------------|-------------------------|----------------------------|--------------------------------|
| 1 | 1                 |                             |                       |                      | BM4Spphke/urnhp3SdgRZVDx/P3Kpjk=   |                         | seb2                    |                            | J83ucoX407JCDzqSqDrK0Q==       |

Nous allons maintenant modifier le JDBCRealm pour passer du ClearText au BCrypt: Security->SecurityRealm->JDBC Realm->Selection de notre realm

JBK Management Console

« Back / Configuration ⇒ Subsystems / Subsystem ⇒ Security ▾ / Settings ⇒ Security Realms ▾

 Security Realm >

 Realm Mappers >

JDBC Realm: jdbcrealm > Principal Query

## Principal Query



The authentication query used to authenticate users based on specific key types.

Showing 1 to 1 of 1 Items

SQL ^

select password,'Admin' From Utilisateur where pseudo =?

Attributes Clear Password Mapper Bcrypt Mapper Modular Crypt Mapper Salted Simple Digest Ma

 Edit  Help

|                   |  |
|-------------------|--|
| SQL               | select password,'Admin' From Utilisateur where pseudo =? |
| Data Source       | PostgresDS   |
| Attribute Mapping | groups ⇒ 2   |

# Remove sur le clear password mapper

The authentication query used to authenticate users based on specific

SQL ^

```
select password,'Admin' From Utilisateur where pseudo =?
```

Attributes

Clear Password Mapper

Bcrypt Mapper

Modul

 Edit  Reset  Remove  Help

Password Index 1

Nous modifions la principal query en rajoutant le salt, et le nombre d'itération (ici en constante). Encore une fois ces données pourraient être dans des colonnes ou venant de jointures d'autres tables

The screenshot shows a web application interface for configuring a SQL query and attribute mapping. At the top, there is a search bar and a table with one item. The table has two columns: 'SQL ^' and 'Data Source'. The single item in the table has the SQL query 'select password,'Admin',salt,31 From Utilisateur where pseudo =?' and the Data Source 'PostgresDS'. Below the table, there are tabs for 'Attributes', 'Clear Password Mapper', 'Bcrypt Mapper', 'Modular Crypt Mapper', 'Salted Simple Digest Mapper', 'Simple Digest Mapper', and 'Scram Mapper'. The 'Attributes' tab is selected. Below the tabs, there is a 'Help' link. The main configuration area has three sections: 'SQL \*' with a text input containing the same SQL query, 'Data Source \*' with a dropdown menu showing 'PostgresDS', and 'Attribute Mapping' with a text input containing 'groups=2'. Below the 'Attribute Mapping' input, there is a note: 'Add new mappings as to=index pairs. Press , to add and X to remove them.'

| SQL ^  | Data Source |
|--|-------------|
| select password,'Admin',salt,31 From Utilisateur where pseudo =? | PostgresDS  |

Showing 1 to 1 of 1 Items

Attributes Clear Password Mapper Bcrypt Mapper Modular Crypt Mapper Salted Simple Digest Mapper Simple Digest Mapper Scram Mapper

Help

SQL \* select password,'Admin',salt,31 From Utilisateur where pseudo =?

Data Source \* PostgresDS

Attribute Mapping groups=2

Add new mappings as to=index pairs. Press , to add and X to remove them.

Nous notons que le salt est la 3eme colonne, et le nombre d'itération (31)

On va configurer le bcrypt: Add sur le bcrypt mapper

Add Bcrypt Mapper

?

 Help

Iteration Count Index \*

Password Index \*

Salt Index \*

Required fields are marked with \*

Cancel

Add

# On va configurer le bcrypt: Add sur le bcrypt mapper

**Add Bcrypt Mapper** ✕

[? Help](#)

Iteration Count Index \*

4

Password Index \*

1

Salt Index \*

3

Required fields are marked with \*

Cancel

Add

Dans notre principal query: password est colonne 1, salt est colonne 3 et le nombre d'itération colonne 4

```
1 select password, 'Admin', salt, 100 From Utilisateur where pseudo =?
```

Une fois la configuration terminée on redémarre Wildfly.  
Et Voilà!