**Librării Criptografice**

*Bcrypt*

În cadrul aplicației [QuickBank](https://github.com/LauraElenaCozma/Banking-App), ce implementează un API în Node.js cu ajutorul căruia pot fi integrate servicii de banking, , am utilizat *node.bcrypt.js* în vederea stocării sigure a parolelor în baza de date. Bcrypt este o librărie pentru Node.js ce utilizează o funcție hash necesară pentru criptarea parolelor, bazată pe algoritmul de criptare Blowfish.

**Stocarea parolelor**

La autentificarea în cadrul aplicațiilor, orice utilizator este nevoit să furnizeze o parolă, ce va fi ulterior utilizată pentru accesarea contului. Stocarea acestora nu trebuie făcută în text clar, deoarece conturile vor deveni vulnerabile atacurilor.

O abordare utilizată în stocarea sigură a parolelor este aplicarea unei funcții hash asupra acestora. În baza de date este stocată parola asupra căreia s-a aplicat funcția hash, iar când userul introduce parola pentru autentificare, acesteia i se aplică funcția hash și este comparată cu ce există în baza de date.

**Ce este o funcție hash?**

O funcție hash este o funcție ce comprimă secvențe de lungime variabilă în secvențe de lungime fixă. Este one-way, deoarece este prea dificil sau imposibil recreezi inputul corespunzător unei valori hash. De asemenea, este deterministă, același mesaj având același rezultat în urma aplicării funcției hash de fiecare dată. O altă proprietate pe care o îndeplinește este rezistența la coliziuni, două mesaje diferite generând mereu două hash-uri diferite.

**De ce este nevoie de salt**

Un aspect important abordat de bcrypt este folosirea unui salt. Una dintre tehnicile folosite de atacatori este precalcularea hash-ului pentru o listă de posibile parole, astfel având posibilitatea să compare rezultatul aplicării funcției hash asupra parola utilizatorului cu variantele pe care acesta le are în baza de date.

Salt este o secvență de text generată aleator, care este adăugată inputului funcției hash, pentru a crea hash-uri diferite chiar dacă inputul nu este unic. Astfel, funcția hash va părea nedeterministă, iar parolele duplicate nu vor fi identificate. Prin aplicarea unei funcții hash asupra unei parole concatenate cu un salt, rezultatul nu va mai fi predictibil.

**Structura unui hash rezultat în urma bcrypt**

Aplicând funcția hash bcrypt pe textul clar Password@123!2021 se va obține rezultatul:

$2a$05$JrD5y.cOxsuZPjUlDtjcceUe8cxsUaG1wfXEA86QArpAZx20tArfm

Alg. Cost Salt Hash

Astfel, se vor identifica componentele:

* 2a – versiunea algoritmului bcrypt utilizată
* 5 – costul 5, reprezentând 25 iterații ale funcției de derivare a cheii
* JrD5y.cOxsuZPjUlDtjcce – saltul aplicat, 16 bytes
* Ue8cxsUaG1wfXEA86QArpAZx20tArfm – rezultatul în urma aplicării funcției hash asupra inputului format din parola Password@123!2021 concatenată cu salt-ul, 24 bytes

**Implementare**

Într-un fișier de configurare setăm valoarea SALT\_ROUND. Aceasta indică timpul necesar pentru a calcula un singur hash folosind bcrypt. Cu cât aceasta este mai mare, numărul de iterații va crește, iar execuția va fi mai lentă. Alegerea costului se face astfel încât să existe rezistență la atacurile prin forță brută, iar puterea de calcul să se mențină ridicată.

În momentul creării unui nou utilizator, asupra parolei introduse de acesta concatenată cu un salt generat automat va fi aplicată funcția hash cu ajutorul bcrypt.hashSync, iar rezultatul va fi stocat în baza de date.



Validarea parolei cu ajutorul funcției hash se va realiza în mutația *login* din fișierul *mutationType.js*. Aceasta va primi ca parametri adresa de email împreună cu o parolă. Se va selecta din baza de date utilizatorul corespunzător adresei de email introduse și cu ajutorul funcției *compare* putem verifica dacă parola în text clar are drept corespundent hash-ul stocat, astfel știind dacă autentificarea a fost realizată cu succes.



**Concluzie**

Bcrypt reprezintă o soluție sigură și eficientă pentru stocarea parolelor în baza de date. Odată ce parolei i s-a aplicat o funcție hash, un atac asupra acesteia își pierde fezabilitatea, conturile utilizatorilor având o securitate ridicată.

**Bibliografie**

* <https://auth0.com/blog/hashing-in-action-understanding-bcrypt/>
* <https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/>
* <https://auth0.com/blog/hashing-passwords-one-way-road-to-security/>
* <https://codahale.com/how-to-safely-store-a-password/>
* <https://culttt.com/2013/01/21/why-do-you-need-to-salt-and-hash-passwords/>
* <https://www.npmjs.com/package/bcrypt>