ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΑΣΦΑΛΕΙΑ ΔΙΚΤΥΩΝ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2022-23

ΓΙΩΡΓΟΣ ΚΟΥΡΟΣ → 3190095

83.212.106.137

ΕΡΓΑΣΙΑ #2

ΕΝΔΕΔΕΙΓΜΕΝΕΣ ΛΥΣΕΙΣ ΑΠΟΘΗΚΕΥΣΗΣ ΚΩΔΙΚΟΥ ΧΡΗΣΤΗ ΓΙΑ ΙΔΙΩΤΙΚΟΤΗΤΑ

- Κατακερματισμός(hashing): Κατακερματισμός είναι η διαδικασία εφαρμογής μιας μαθηματικής συνάρτησης στον κωδικό πρόσβασης, δημιουργώντας μια έξοδο σταθερού μήκους που ονομάζεται hash. Οι λειτουργίες κατακερματισμού όπως bcrypt, scrypt και Argon2 θεωρούνται ασφαλείς και συνιστώνται για αποθήκευση κωδικού πρόσβασης.
- Κρυπτογράφηση (encryption): Η διαδικασία μετατροπής απλού κειμένου σε κρυπτογραφημένο κείμενο, το οποίο δεν μπορεί να διαβαστεί χωρίς κλειδί,. Ωστόσο, επειδή η κρυπτογράφηση είναι μια αμφίδρομη διαδικασία και ο κωδικός πρόσβασης μπορεί να αποκρυπτογραφηθεί, δεν συνιστάται η χρήση του για αποθήκευση κωδικού πρόσβασης.
- Χρήση 2FA (two-factor authentication): Το 2FA (έλεγχος ταυτότητας δύο παραγόντων) είναι μια μέθοδος ασφαλείας στην οποία ο χρήστης δίνει δύο ξεχωριστούς παράγοντες ελέγχου ταυτότητας για τον έλεγχο ταυτότητας.
- Password Manager: Η διαχείριση κωδικών πρόσβασης είναι ένα εργαλείο λογισμικού που επιτρέπει στους προγραμματιστές να δημιουργούν, να αποθηκεύουν και να διαχειρίζονται κωδικούς πρόσβασης. Αποτελεί μια έξυπνη τεχνική για τη διατήρηση των κωδικών πρόσβασης ώστε να παραμένουν ασφαλείς.
- Salting: Το Salting είναι μια τυχαία τιμή που προστίθεται στον κωδικό πρόσβασης πριν κατακερματιστεί. Αυτό βοηθά στην αποτροπή προ-υπολογισμένων επιθέσεων hashing, όπου οι εισβολείς μπορούν να δημιουργήσουν έναν πίνακα με προϋπολογισμένους κατακερματισμούς για κωδικούς πρόσβασης που χρησιμοποιούνται αρκετά συχνά.

Στον πίνακα users που δημιουργήσαμε, θέλουμε να εξασφαλίσουμε μεγαλύτερο επίπεδο ασφαλείας κρυπτογραφώντας τους κωδικούς με κατακερματισμό (hashing). Αυτό το επιτυγχάνουμε με την συνάρτηση crypt() έχοντας ενεργοποιήσει το pgcrypto extension της PostgreSQL εφαρμόζοντας salting με την συνάρτηση gen_salt() με παράμετρο ‘bf’ για χρήση του Blowfish αλγόριθμου.Ύστερα διαγράφουμε την στήλη password έχοντας πλέον στον πίνακα την καινούργια στήλη με τα κρυπτογραφημένα πλέον passwords την οποία μετονομάζουμε σε password για πρακτικούς λόγους. Τέλος, δημιουργήθηκε συνάρτηση hash() ώστε να πραγματοποιεί την διαδικασία της κρυπτογράφησης του κωδικού (hashing-salting) ώστε να μην χρειάζεται να γίνεται χειροκίνητα κάθε φορά που θέλουμε να προσθέτουμε νέα δεδομένα χρηστών. **(index.sql)**

Σχετικά με την άσκηση 3, έγιναν πολλές προσπάθειες να υλοποιήσω τον server που έφτιαξα να τρέχει μέσω python, να τρέχει με WSGI server (όπου κατέβαζε έκδοση 3.4 αρκετά παλιά και άμα κατέβαζα νεότερες εκδόσεις με νεότερο python interpreter δεν μπορούσαν να περάσουν μέσα στον Apache) από το

Default του Apache αλλά το πρόβλημα ήταν ότι μπορούσα να χρησιμοποιήσω μόνο python 2.7.5 interpreter με την εκκίνηση του wsgi server στον Apache και το script **index.py** είναι γραμμένο σε python3-newer, με αποτέλεσμα να μην μπορεί να αναγνωρίσει βιβλιοθήκες που χρησιμοποίησα όπως **flask, bcrypt, pg8000.** Αυτό είχε ως κατάληξη να τρέχει ο server μόνο χειροκίνητα μεταβαίνοντας στο /var/www/html και εκτελώντας python index.py, ύστερα έχοντας ρυθμίσει τον server να απαντά σε http://83.212.106.137:5000/, ανοίγοντας αυτό το link είναι πλήρως λειτουργικός. Προϋπόθεση βέβαια για να τρέχει εκεί ήταν να διαμορφώσω τις ρυθμίσεις του firewall και να επιτρέψω σύνδεση στο port 5000. Προσπάθησα επίσης να το σερβίρω σε https αλλά δεν μπορούσε να αναγνωριστεί, καταλάβαινε μόνο http και μου έβγαζε στο terminal ότι το port αυτό (443) χρησιμοποιείται ήδη. Γενικά δεν μπόρεσα να καταφέρω να είναι σε https στον default του apache. Άλλη μια προσπάθεια ήταν να χρησιμοποιήσω proxy server με τα modules του apache αλλά μου έβγαζε error permission denied κάτι το οποίο δεν μπορούσα να λύσω ακόμα και με ρυθμίσεις στο firewall. Σχετικά με το αρχείο **index.py** χρησιμοποιήθηκε το **Flask framework** για να τρέξει ο σερβερ (βιβλιοθήκη flask), η βιβλιοθήκη **bcrypt** για αποκρυπτογράφηση,η βιβλιοθήκη **pg8000** για σύνδεση του server με την **postgresql** βάση GDPR και η **datetime** για την διαχείριση των timestamps για την άσκηση 4. Πιο συγκεκριμένα για το flask framework είναι ένα built-in python framework από το οποίο κάνουμε import την κλάση Flask, την μέθοδο render_template() και την μέθοδο request. Η κλάση Flask χρησιμοποιείται στην γραμμή **app = Flask(__name__)** δημιουργώντας ένα αντικείμενο Flask στην μεταβλητή app ώστε πάνω σε αυτή τη μεταβλητή να εφαρμόσουμε τις μεθόδους του framework. Το **@app.route** χρησιμοποιείται για το routing της web εφαρμογής όπου έχουμε root '/' και login '/login' endpoint. Ύστερα με την **render_template()** μέθοδο φορτώνουμε στο root την φόρμα html που έχουμε δημιουργήσει και η μέθοδος **request** χρησιμοποιείται για την λήψη των δεδομένων που πληκτρολογήθηκαν στην φόρμα σύνδεσης. Επίσης η **app.template_folder** γνωστοποιεί στο flask framework που να ψάξει για html αρχεία ώστε να κάνει render το σωστό .html file. Εν τέλει φτάνουμε στις τελευταίες γραμμές του κώδικα όπου έχουμε την εντολή **app.run** η οποία εκκινεί τον server μας βάζοντας ως παραμέτρους τον host, το port και debug=True ώστε να μπορούμε με μεγαλύτερη ευκολία να διορθώνουμε τυχόν σφάλματα που προκύπτουν. ***Συνοψίζοντας τον server μπορείτε να τον τρέξετε και να τον τσεκάρετε μπαίνοντας ως teacher (δοκιμασμένο), πηγαίνοντας στο /var/www/html, εκτελώντας python index.py και ύστερα να μεταβείτε εδώ http://83.212.106.137:5000/. Τέλος τα στοιχεία των 2 users (3190095,admin) υπάρχουν σε αντίστοιχο notes.txt αρχείο για debugging.***

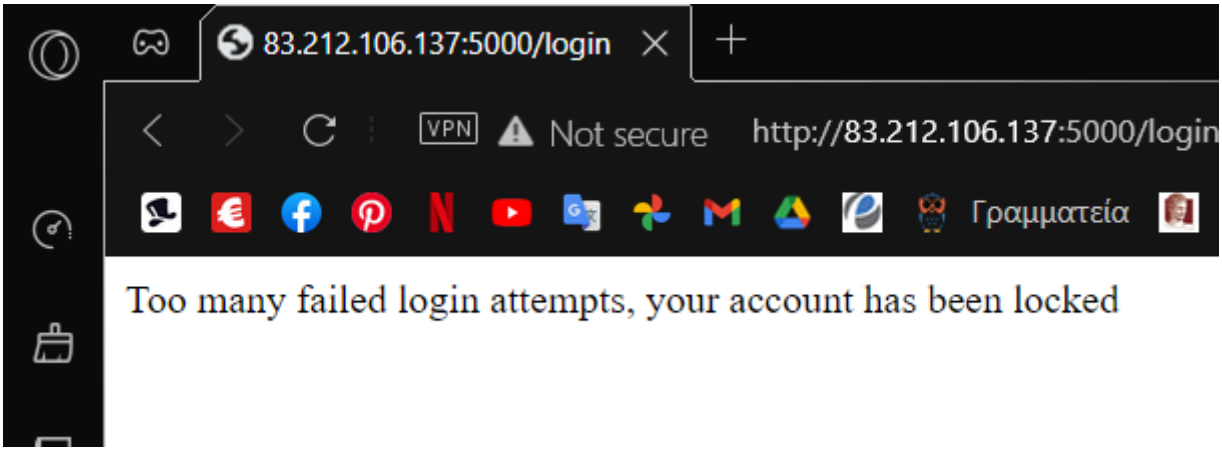
Άσκηση 4 Screenshots-παραδείγματα

Έχοντας τροποποιήσει τον κώδικα του **index.py**, τρέχοντας τον server και βάζοντας για admin και 3190095 από μία φορά σωστό password και ύστερα βάζοντας για τον καθένα 5 φορές λάθος password έχουμε τα εξής δεδομένα στον πίνακα logging της βάσης GDPR :

```
^C[root@snf-890528 html]# psql -U postgres -d GDPR
Password for user postgres:
psql (9.2.24)
Type "help" for help.

GDPR=# select * from logging;
 id | username |         timestamp          | success
----+----------+----------------------------+---------
 26 | admin    | 2023-01-24 09:15:26.968467 | t
 27 | 3190095  | 2023-01-24 09:16:00.307515 | t
 28 | 3190095  | 2023-01-24 09:16:04.271519 | f
 29 | 3190095  | 2023-01-24 09:16:04.315767 | f
 30 | 3190095  | 2023-01-24 09:16:06.54333  | f
 31 | 3190095  | 2023-01-24 09:16:08.603276 | f
 32 | 3190095  | 2023-01-24 09:16:10.483394 | f
 33 | admin    | 2023-01-24 09:16:12.842668 | f
 34 | admin    | 2023-01-24 09:16:19.022017 | f
 35 | admin    | 2023-01-24 09:16:21.537119 | f
 36 | admin    | 2023-01-24 09:16:23.333364 | f
 37 | admin    | 2023-01-24 09:16:25.483526 | f
(12 rows)
```

Και στο front-end πλέον όποτε ξαναπροσπαθεί να συνδεθεί είτε ο admin είτε ο 3190095, ακόμα και με σωστό κωδικό, του εμφανίζεται αυτό το μήνυμα :

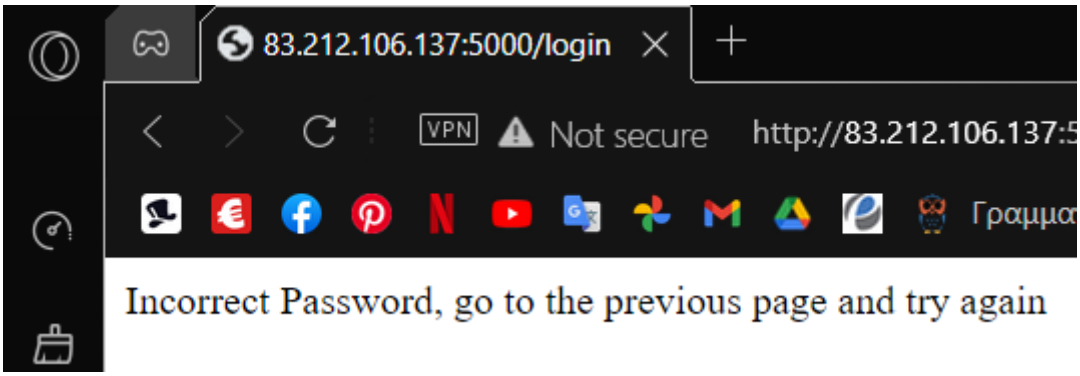


Ο κώδικας έχει διαμορφωθεί ώστε να επιτρέπει σε κάθε χρήστη να κάνει 4 λάθος απόπειρες σύνδεσης με λάθος κωδικό. Στην 5η φορά κλειδώνετε από τον server και σε κάθε επόμενη του προσπάθεια θα λαμβάνει αυτό το μήνυμα. Στις προηγούμενες τέσσερις λαμβάνει απλά μήνυμα ότι πληκτρολόγησε λάθος κωδικό και να προσπαθήσει ξανά. Βλέποντας ένα άλλο σενάριο όπου ο admin πραγματοποιεί 3 αποτυχημένες προσπάθειες και η 4η επιτυχής ενώ ο 3190095 2 αποτυχημένες προσπάθειες και η 3η επιτυχής, ο πίνακας logging έχει τα εξής δεδομένα :

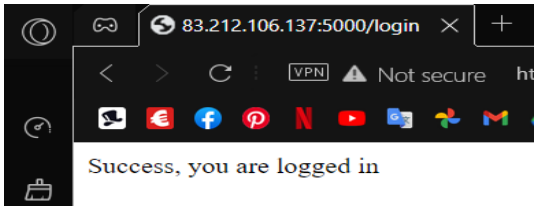
```
psql (9.2.24)
Type "help" for help.

GDPR=# select * from logging;
 id | username |          timestamp          | success
----+----------+-----------------------------+---------
 38 | admin    | 2023-01-24 09:23:28.213252 | f
 39 | admin    | 2023-01-24 09:23:32.34247  | f
 40 | admin    | 2023-01-24 09:23:35.240428 | f
 41 | admin    | 2023-01-24 09:23:37.462168 | t
 42 | 3190095  | 2023-01-24 09:23:47.813117 | f
 43 | 3190095  | 2023-01-24 09:23:47.982973 | f
 44 | 3190095  | 2023-01-24 09:23:50.729871 | t
(7 rows)
```

Σε κάθε αποτυχημένη προσπάθεια των 2 χρηστών, στο front-end τους εμφανίζεται το μήνυμα :



Ενώ όταν πλέον ο καθένας τους κάνει επιτυχής σύνδεση, εμφανίζεται στο front-end το μήνυμα :



ΠΛΗΡΗΣ ΑΝΑΠΑΡΑΣΤΑΣΗ-ΔΙΑΔΙΚΑΣΙΑ TERMINAL ΑΠΟ ΤΟ ΠΡΟΗΓΟΥΜΕΝΟ ΠΑΡΑΔΕΙΓΜΑ

```
root@snf-890528:/var/www/html
[root@snf-890528 html]# python index.py
 * Serving Flask app 'index' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://83.212.106.137:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 482-677-735
195.251.255.75 - - [24/Jan/2023 09:23:22] "GET / HTTP/1.1" 200 -
195.251.255.75 - - [24/Jan/2023 09:23:28] "POST /login HTTP/1.1" 200 -
195.251.255.75 - - [24/Jan/2023 09:23:32] "POST /login HTTP/1.1" 200 -
195.251.255.75 - - [24/Jan/2023 09:23:35] "POST /login HTTP/1.1" 200 -
195.251.255.75 - - [24/Jan/2023 09:23:37] "POST /login HTTP/1.1" 200 -
195.251.255.75 - - [24/Jan/2023 09:23:41] "POST /login HTTP/1.1" 200 -
195.251.255.75 - - [24/Jan/2023 09:23:47] "POST /login HTTP/1.1" 200 -
195.251.255.75 - - [24/Jan/2023 09:23:50] "POST /login HTTP/1.1" 200 -
195.251.255.75 - - [24/Jan/2023 09:24:15] "POST /login HTTP/1.1" 200 -
^C[root@snf-890528 html]# psql -U postgres -d GDPR
Password for user postgres:
psql: FATAL:  password authentication failed for user "postgres"
[root@snf-890528 html]# psql -U postgres -d GDPR
Password for user postgres:
psql (9.2.24)
Type "help" for help.

GDPR=# select * from logging;
 id | username |         timestamp          | success
----+----------+----------------------------+---------
 38 | admin    | 2023-01-24 09:23:28.213252 | f
 39 | admin    | 2023-01-24 09:23:32.34247  | f
 40 | admin    | 2023-01-24 09:23:35.240428 | f
 41 | admin    | 2023-01-24 09:23:37.462168 | t
 42 | 3190095  | 2023-01-24 09:23:47.813117 | f
 43 | 3190095  | 2023-01-24 09:23:47.982973 | f
 44 | 3190095  | 2023-01-24 09:23:50.729871 | t
(7 rows)
```

Ο κώδικας **index.py** έχει ρυθμιστεί ώστε να μετράει τις αποτυχημένες προσπάθειες μετά την τελευταία επιτυχή σύνδεση του χρήστη. Αυτό σημαίνει ότι αν ο χρήστης έχει στο πίνακα logging 2 αποτυχημένες, ύστερα μία επιτυχημένη και μετά 3 αποτυχημένες προσπάθειες, όταν πάει να συνδεθεί με σωστό κωδικό θα είναι επιτυχής η σύνδεση. Προκείμενου να του παρουσιαστεί μήνυμα ότι πραγματοποίησε πολλές αποτυχημένες προσπάθειες πρέπει να πραγματοποιήσει 5 αποτυχημένες προσπάθειες στη σειρά.

Σχετικά με το κομμάτι του κώδικα που ελέγχει πότε ήταν η τελευταία αλλαγή κωδικού του κάθε χρήστη, έχει υλοποιηθεί ώστε να απαιτεί από τον χρήστη αλλαγή κωδικού κάθε 6 μήνες (180 ημέρες). Υπάρχουν κατάλληλες μεταβλητές στον κώδικα **index.py** οι οποίες εξηγούνται και με σχόλια # που μετράνε κάθε φορά πριν γίνει η επιτυχής σύνδεση πόσες μέρες έχει «ζήσει» ο εκάστοτε κωδικός του χρήστη. Αυτό το κομμάτι του κώδικα βεβαιώθηκα ότι δουλεύει βάζοντας στην συνθήκη ***if time_passed.days > 180:*** αντί για 180 ημέρες, 0 ημέρες και πράγματι όποτε οι χρήστες προσπαθούσαν να συνδεθούν με τον σωστό τους κωδικό, στον πίνακα logging δεν προσθέτονταν δεδομένα και εμφανιζόταν στον κάθε χρήστη στο front-end το ανάλογο μήνυμα που φαίνεται και στον κώδικα. **Επιπλέον προστέθηκε στον πίνακα users νέα στήλη last_pwd_change όπως φαίνεται και στο αρχείο index.sql με timestamp.**

Τέλος θα ήθελα να σημειώσω πως έχοντας πλέον την τελική μορφή του πίνακα users, αν υποθετικά θέλαμε να προσθέσουμε καινούργιο χρήστη θα εκτελούσαμε sql εντολή με την παρακάτω δομή :

```
INSERT INTO users (username, description, password, last_pwd_change)
VALUES ('username', 'description', hash('password'), NOW());
```

Και αν θέλαμε να ενημερώσουμε κάποιο κωδικό χρήστη που άλλαξε θα εκτελούσαμε sql εντολή με την παρακάτω δομή :

```
UPDATE users SET password=hash('newpassword'), last_pwdchange=NOW()
WHERE username='username';
```