SQL Injection
Αναφορά Εργασίας

Φίλιππος Δουραχαλής, 3170045

1. Δημιουργούμε τη PSQL βάση δεδομένων GDPR και προσθέτουμε τον πίνακα users μέσω των εντολών που παρέχονται στα SQL αρχεία “create_database”, “create_users” και “insert_users”. Θεωρούμε ότι κάθε χρήστης έχει μοναδικό όνομα, επομένως το πεδίο username ορίζεται ως πρωτεύον κλειδί.

2. Οι κωδικοί που αποθηκεύονται στη βάση πρέπει να είναι κρυπτογραφημένοι, διαφορετικά ο καθένας μπορεί να τους δει ως plaintext, όπως είχαν καταχωρηθεί. Επιπλέον κατά την κρυπτογράφηση του κωδικού, μαζί με το plaintext που περιέχει τον κωδικό πρέπει να δώσουμε μια τυχαία τιμή salt η οποία θα χρησιμοποιηθεί κατά την κρυπτογράφηση. Αυτό γίνεται έτσι ώστε αν δύο ή παραπάνω χρήστες έχουν τον ίδιο κωδικό και κάποιος επιτιθέμενος αποκτήσει πρόσβαση στη βάση μας, να μην είναι δυνατή η εξαγωγή κάποιου συμπεράσματος για τους κωδικούς αφού αυτοί αν και πανομοιότυποι, με τη χρήση του salt θα έχουν διαφορετικό hash και άρα η σύγκρισή τους δεν θα οδηγήσει σε κάποιο στοιχείο για την τιμή τους.
   Οι κωδικοί που δημιουργούμε για κάθε χρήστη μπορούν να κρυπτογραφηθούν χρησιμοποιώντας την επέκταση pg_crypto που παρέχει η PostgreSQL. Προσθέτουμε αυτό το extension στη βάση με την εντολή:

   ```
   CREATE EXTENSION pg_crypto;
   ```

   Μόλις προσθέσουμε το συγκεκριμένο extension μπορούμε να χρησιμοποιήσουμε τη συνάρτηση crypt() για να παραγάγουμε το hash του κωδικού που επιλέγουμε. Με τη μέθοδο gen_salt() ορίζουμε τον αλγόριθμο κρυπτογράφησης που θα χρησιμοποιηθεί, ο οποίος είναι ο Blowfish.
   Τα ανωτέρω φαίνονται στο αρχείο SQL “insert_users”.
   Με την ίδια μέθοδο crypt() μπορούμε επίσης να ελέγξουμε αν ένας κωδικός που δόθηκε ταιριάζει με τον κρυπτογραφημένο κωδικό ενός δεδομένου χρήστη.

```
 username |                           password                           |   description    |
----------+--------------------------------------------------------------+------------------+--
 p3170045 | $2a$06$b5MKwpuFaaauo.YKQG65WubVT.QLM9eI9nesjwgSWac.qYixI8FLO | This is my user  |
 admin    | $2a$06$AlWboqalGeYilvPSWVkt3uAKTghqSiUdzLw2iW7O2ybVNjmPwHuD2 | Priviledged user |
(2 rows)
```

```
GDPR=# SELECT username, password, description FROM users WHERE username = 'p3170045' AND password = crypt('991015', password);
 username |                          password                          |   description
----------+------------------------------------------------------------+-----------------
 p3170045 | $2a$06$tKUNYMe.JnFlpEyrfXloy.9dZ5DVNDm5vUu0rBaQauwvURvjBepBi | This is my user
(1 row)
```

3. Για την υλοποίηση της web εφαρμογής χρησιμοποιήθηκε το framework Java Spring. Αρχικά χρησιμοποιήθηκε το Spring Initializr για την αρχικοποίηση του project και την εισαγωγή των κατάλληλων dependencies. Για τις ανάγκες της εργασίας σε αυτό το ερώτημα έχουν υλοποιηθεί οι παρακάτω κλάσεις:

   User: Αναπαριστά έναν χρήστη της εφαρμογής όπως έχει οριστεί στην σχέση users. Περιλαμβάνει κατάλληλα πεδία για την αποθήκευση και τη διαχείριση των δεδομένων που σχετίζονται με εκείνον.

   UsersRepository: Ένα JPA Repository που περιλαμβάνει μεθόδους για την αναζήτηση ενός χρήστη στη βάση και την ενημέρωση των πεδίων του.

   UserDetailsImpl: Υλοποιεί ένα αντικείμενο UserDetails, το οποίο ενθυλακώνει πληροφορίες σχετικές με τον χρήστη ώστε να χρησιμοποιηθεί κατά την αυθεντικοποίηση του.

   UserLoginService: Παρέχει πληροφορίες σχετικά με την αυθεντικοποίηση ενός χρήστη, όπως το μέγιστο πλήθος αποτυχημένων προσπαθειών σύνδεσης, το χρονικό όριο κλειδώματος ενός χρήστη μετά το πλήθος αυτό καθώς και την περίοδο ισχύος του κωδικού του. Επίσης περιλαμβάνει μεθόδους για την εύρεση ενός χρήστη που ζητά να συνδεθεί στο σύστημα, το κλείδωμα και το ξεκλείδωμα του λογαριασμού του και τον έλεγχο για τον αν ο κωδικός του είναι σε ισχύ.

```
public static final int MAX_FAILED_ATTEMPTS = 3;
private static final long LOCK_TIME_DURATION = 60 * 1000; // unlock user after a period of time (in msec)
private static final long PWD_PERIOD_VALIDITY = 60 * 60 * 1000; //period after which password change is required (in msec)
```

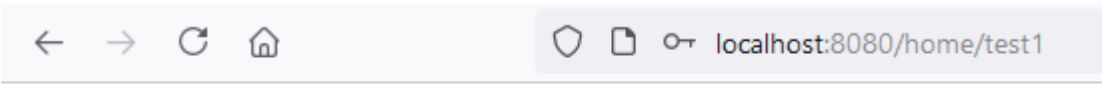
   WebSecurityConfig: Διαμορφώνει τις παραμέτρους ασφαλείας που θα χρησιμοποιήσει ο server. Εδώ ορίζουμε ποια endpoints είναι προσπελάσιμα από ποιους χρήστες, ποια μέθοδος εισόδου χρησιμοποιείται και ποια σελίδα θα εμφανιστεί στον χρήστη για την είσοδο, καθώς και τι θα συμβεί σε περίπτωση επιτυχούς ή ανεπιτυχούς login.

   Η διαδικασία της αυθεντικοποίησης, δηλαδή ο έλεγχος των στοιχείων του χρήστη και η διασταύρωσή (επιλογή) τους με τα στοιχεία της βάσης γίνεται εσωτερικά μέσω του Hibernate. Η προσέγγιση αυτή είναι ασφαλής ενάντια σε επιθέσεις SQL Injection καθώς η Hibernate δεν πραγματοποιεί συνένωση των στοιχείων εισόδου που δίνονται από το χρήστη, αλλά τα χρησιμοποιεί ως παραμέτρους για να συνθέσει ένα δικό της ερώτημα και να το υποβάλει στην βάση. Επομένως όχι μόνο δεν χρειάζεται να γράψουμε δικά μας SQL

ερωτήματα για το σκοπό αυτό, αλλά μπορούμε να είμαστε σίγουροι ότι ένας επιτιθέμενος δεν θα μπορέσει να αποκτήσει με οποιοδήποτε τρόπο πρόσβαση στη βάση δίνοντας ένα ειδικά διαμορφωμένο string. Παρακάτω βλέπουμε ένα ερώτημα που παράγεται σε μια απόπειρα σύνδεσης:

```
Hibernate:
    select
        user0_.username as username1_1_,
        user0_.description as descript2_1_,
        user0_.enabled as enabled3_1_,
        user0_.failed_attempts as failed_a4_1_,
        user0_.pwd_last_modified as pwd_last5_1_,
        user0_.lock_time as lock_tim6_1_,
        user0_.locked as locked7_1_,
        user0_.password as password8_1_
    from
        public.users user0_
    where
        user0_.username=?
```

Όλα τα endpoints πλην του login προστατεύονται και είναι διαθέσιμα μόνο στους χρήστες που έχουν αυθεντικοποιηθεί. Για παράδειγμα αν ζητήσουμε τη σελίδα που βρίσκεται στη διεύθυνση http://localhost:8080/home/test1, το σύστημα θα μας ανακατευθύνει στη σελίδα του login. Μετά από μια επιτυχή σύνδεση, το σύστημα μας εμφανίζει τον πόρο που ζητήθηκε:



Hello there! There's nothing here yet

Διαφορετικά εμφανίζει κατάλληλο μήνυμα λάθους:


```
{"exception":"Bad credentials","timestamp":1623520870925}
```

Επίσης κατά την αυθεντικοποίηση, ο κωδικός που εισήγαγε ο χρήστης κρυπτογραφείται. Το κομμάτι αυτό χειρίζεται η κλάση **BCryptPasswordEncoder**, η οποία δίνεται ως όρισμα στον Authentication Provider που χειρίζεται τη διαδικασία της αυθεντικοποίησης.

```java
@Bean
public DaoAuthenticationProvider daoAuthenticationProvider(){
    DaoAuthenticationProvider provider = new DaoAuthenticationProvider();
    provider.setPasswordEncoder(encoder);
    provider.setUserDetailsService(userLoginService);
    return provider;
}
```

Οι κανόνες σχετικά με τα endpoints, τα φίλτρα που εφαρμόζονται και τη ***configure****(HttpSecurity httpSecurity)* της κλάσης **WebSecurityConfig:**

```java
@Override
protected void configure(HttpSecurity httpSecurity) throws Exception{
    httpSecurity.cors().and().csrf().disable() //enable CORS and disable CSRF
            .exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and() //set exception handler for unauthorized requests
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
            .authorizeRequests()
            .antMatchers( ...antPatterns: "api/**").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .failureHandler(loginFailureHandler).successHandler(loginSuccessHandler).permitAll(); //allow users to access login form and set handlers for failed and successful login
    httpSecurity.addFilterBefore(authenticationJwtTokenFilter(), UsernamePasswordAuthenticationFilter.class);
}
```

Σημειώνεται ότι στην συγκεκριμένη μέθοδο μπορούμε να ορίσουμε και την δική μας σελίδα που θα χρησιμοποιηθεί κατά το login μέσω της μεθόδου *loginPage*(“/login”). Παρότι έχει δημιουργηθεί η σελίδα login.html στα αρχεία του project, δεν χρησιμοποιείται και αντ’ αυτής εμφανίζεται η default σελίδα της Spring boot security, καθώς σε διάφορες δοκιμές το σύστημα δεν ήταν σε θέση αν εντοπίσει την custom σελίδα (Error 404).

4. Για το συγκεκριμένο ερώτημα δημιουργείται ο πίνακας logging στη βάση. Στον συγκεκριμένο πίνακα καταγράφονται στη βάση όλες οι απόπειρες σύνδεσης των χρηστών και αν ήταν επιτυχημένες ή όχι. Επιπλέον, εμπλουτίζουμε τον πίνακα users με 5 νέα πεδία μέσω των οποίων παρακολουθούμε τις αποτυχημένες απόπειρες σύνδεσης ενός χρήστη (**failed_attempts**), το αν ο λογαριασμός ενός χρήστη είναι κλειδωμένος (**locked**), τη χρονική στιγμή του κλειδώματος (**lock_time**), το αν ο λογαριασμός είναι ενεργός (**enabled**) και το πότε τροποποιήθηκε τελευταία φορά ο κωδικός του χρήστη (**pwd_last_modified**).
   Οι παραπάνω εντολές φαίνονται στα αρχεία “create_logging”, “alter_users” και “update_users”.

   Στη συνέχεια ορίζουμε τις εξής νέες κλάσεις στο project:

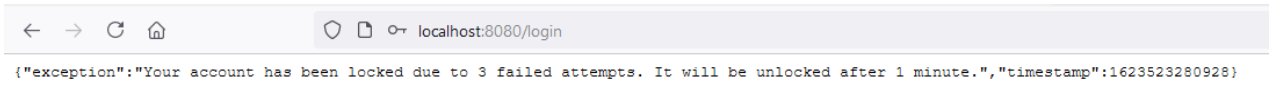
   Logging: Αναπαριστά τον πίνακα logging της βάσης. Κάθε αντικείμενό της είναι μια απόπειρα σύνδεσης

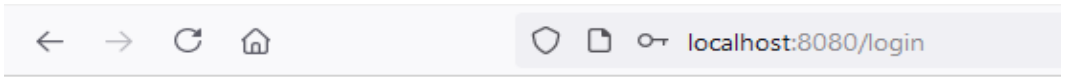
   LoggingCompositeKey: Το σύνθετο κλειδί του ανωτέρω πίνακα.

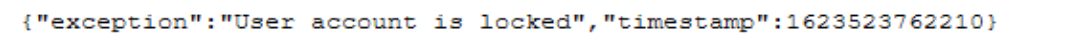
   LogRepository: Το JPA Repository που παρέχει μεθόδους ανάκτησης όλων των εγγραφών για έναν δεδομένο χρήστη και διαχείριση της σχέσης logging.

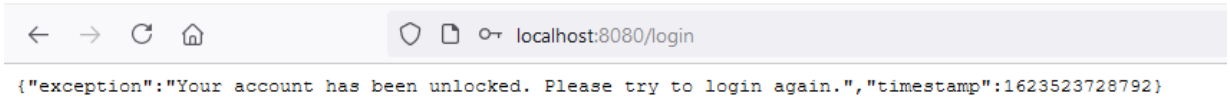
CustomAuthenticationFailureHandler: Η συγκεκριμένη κλάση καλείται στην περίπτωση μια αποτυχημένης προσπάθειας σύνδεσης του χρήστη και νέσω της μεθόδου *onAuthenticateFailure* είναι υπεύθυνη για να καταγράψει την αποτυχημένη προσπάθεια στον πίνακα logging και να ελέγξει αν απαιτείται κλείδωμα του χρήστη. Το πλήθος αποτυχημένων προσπαθειών πριν το κλείδωμα, όπως έχει οριστεί στην UserLoginService είναι 3. Μόλις ο χρήστης υπερβεί αυτό το όριο, το πεδίο locked του χρήστη γίνεται true, ενώ το lock_time ορίζεται στην τρέχουσα ημέρα και ώρα και εμφανίζεται κατάλληλο exception με μήνυμα λάθους. Η περίοδος κλειδώματος έχει οριστεί σε 1 λεπτό. Κάθε απόπειρα σύνδεσης σε αυτό το διάστημα οδηγεί σε μήνυμα λάθους που ενημερώνει τον χρήστη ότι πρέπει να περιμένει. Όταν παρέλθει αυτό το διάστημα, η μέθοδος πιάνει την επόμενη προσπάθεια σύνδεσης, βλέπει ότι ο λογαριασμός του χρήστης πρέπει να ξεκλειδώσει, αίρει το κλείδωμα και τον καλεί να προσπαθήσει ξανά.

Παρακάτω φαίνονται τα μηνύματα που εμφανίζονται στις περιπτώσεις που περιγράφηκαν.

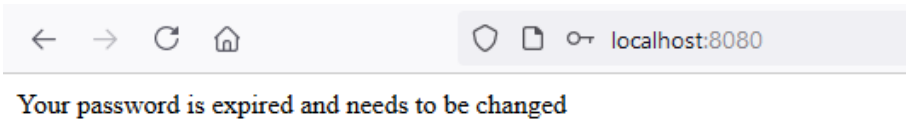








CustomAuthenticationSuccessHandler: Η κλάση αυτή χειρίζεται τις επιτυχημένες προσπάθειες σύνδεσης. Μόλις ένας χρήστης αυθεντικοποιηθεί από το σύστημα, η απόπειρα σύνδεσης καταγράφεται όπως και προηγουμένως και επιπροσθέτως γίνεται έλεγχος για το αν ο κωδικός του χρήστη είναι σε ισχύ. Αν όχι, εμφανίζει κατάλληλο μήνυμα που τον ενημερώνει πως πρέπει να πραγματοποιήσει αλλαγή κωδικού, διαφορετικά εμφανίζεται το τυπικό μήνυμα εισόδου.



5. Για το κομμάτι της αυθεντικοποίησης μέσω JWT Token έχουν υλοποιηθεί κάποιες κλάσεις που παράγουν και ελέγχουν τα tokens που διανέμονται στους χρήστες. Ωστόσο η λειτουργία αυτή **δεν** έχει ενσωματωθεί στην εφαρμογή καθώς θέτοντας το authentication entry point όπως φαίνεται στη συνέχεια προκαλεί unauthorized exception για όλα τα endpoints της εφαρμογής, ανεξαρτήτως κανόνων. Οι κλάσεις που έχουν δημιουργηθεί για αυτό τον σκοπό είναι:

AuthEntryPointJwt: Η κλάση που χειρίζεται exceptions σχετικά με άρνηση πρόσβασης στα endpoints της εφαρμογής, στέλνοντας κατάλληλη απάντηση.

AuthTokenFilter: Φίλτρο από το οποίο περνάει ένα αίτημα αυθεντικοποίησης. Μέσα στη μέθοδο `doFilterInternal` η εφαρμογή λαμβάνει το αίτημα, εξάγει το jwt token και επαληθεύει ότι είναι σωστό. Κατόπιν εξάγει τα στοιχεία του χρήστη και δημιουργεί το αντικείμενο προς αυθεντικοποίηση.

JWTUtils: Βοηθητική κλάση που περιλαμβάνει μεθόδους για την δημιουργία και επαλήθευση της εγκυρότητας ενός token. Επίσης περιλαμβάνει το μυστικό κλειδί του server και τον χρόνο ισχύος του token (ορίζονται στο αρχείο application.properties).

LoginController: Ορίζει ένα endpoint για το login του χρήστη. Λαμβάνει ένα αίτημα για σύνδεση (LoginRequest), δηλαδή το όνομα και τον κωδικό του χρήστη, παράγει το JWT token και πραγματοποιεί την αυθεντικοποίηση ή πετάει exception αν ο χρήστης έδωσε λάθος στοιχεία.

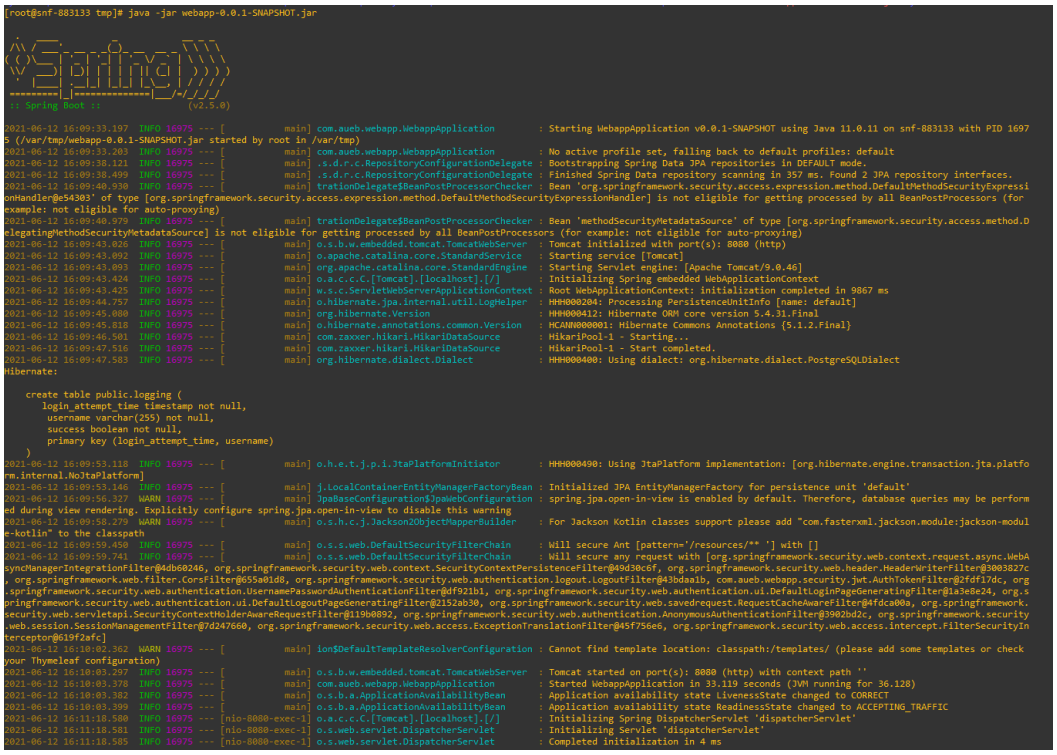
Για την ενσωμάτωση των ανωτέρω στην εφαρμογή αρκεί να κάνουμε uncomment τις εξής δύο γραμμές εντός της μεθόδου configure(HttpSecurity httpSecurity) στην WebSecurityConfig. Τονίζεται ότι όπως αναφέρθηκε, ενδέχεται η εφαρμογή να μην δουλεύει όπως πρέπει όσο οι γραμμές αυτές συμπεριλαμβάνονται στο configuration:

```
@Override
protected void configure(HttpSecurity httpSecurity) throws Exception{
    httpSecurity.cors().and().csrf().disable() //enable CORS and disable CSRF
//              .exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and() //set exception handler for unauthorized requests
//              .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
```

Τέλος, έχοντας ελέγξει ότι η εφαρμογή δουλεύει όπως πρέπει, παράγουμε ένα εκτελέσιμο jar μεταβαίνοντας στον φάκελο του project και πληκτρολογώντας την εντολή:

```
mvn package
```

Στη συνέχεια μεταφέρουμε το jar αρχείο στον server και το εκτελούμε.

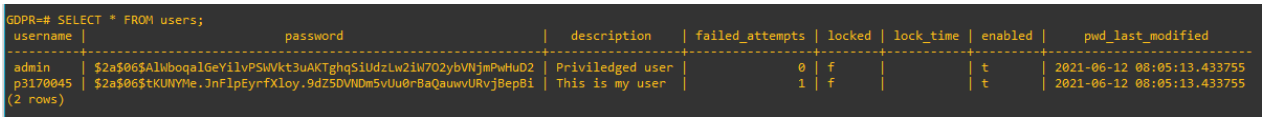


Ελέγχουμε και εδώ ότι η εφαρμογή παράγει τα αναμενόμενα αποτελέσματα τρέχοντας την εντολή

```
curl -X POST -c cookies ‘http://localhost:8080/login’ -d ‘username=p3170045&password=991015’
```

Η εντολή αυτή στέλνει ένα POST αίτημα στο login endpoint της εφαρμογής με τις παραμέτρους σύνδεσης.

Παράλληλα έχουμε ανοίξει δύο διαφορετικές συνεδρίες στον server: Μια για να παρακολουθούμε το output της εφαρμογής και μια όπου παρακολουθούμε τους πίνακες της βάσης ώστε να ελέγξουμε ότι τα περιεχόμενά τους ανακτώνται και ανανεώνονται σωστά καθώς σε μια άλλη συνεδρία τρέχουμε τις εντολές σύνδεσης. Παρακάτω βλέπουμε το output για μια σωστή και μια λανθασμένη προσπάθεια login τροποποιώντας κατάλληλα το password στην εντολή curl:



```
GDPR=# SELECT * FROM logging;
   login_attempt_time     | username | success
--------------------------+----------+---------
 2021-06-12 16:25:09.888 | p3170045 | t
 2021-06-12 16:29:59.009 | p3170045 | f
(2 rows)
```

Και μια ακόμη επιτυχή προσπάθεια σύνδεσης που μηδενίζει τους μετρητές στον πίνακα users:

```
GDPR=# SELECT * FROM users;
 username |                          password                          |   description    | failed_attempts | locked | lock_time | enabled |     pwd_last_modified
----------+------------------------------------------------------------+------------------+-----------------+--------+-----------+---------+----------------------------
 admin    | $2a$06$AlWboqalGeYilvPSWVkt3uAKTghqSiUdzLw2iW7O2ybVNjmPwHuD2 | Priviledged user |               0 | f      |           | t       | 2021-06-12 08:05:13.433755
 p3170045 | $2a$06$tKUNYMe.JnFlpEyrfXloy.9dZ5DVNDm5vUu0rBaQauwvURvjBepBi | This is my user  |               0 | f      |           | t       | 2021-06-12 08:05:13.433755
(2 rows)
```

```
GDPR=# SELECT * FROM logging;
   login_attempt_time     | username | success
--------------------------+----------+---------
 2021-06-12 16:25:09.888 | p3170045 | t
 2021-06-12 16:29:59.009 | p3170045 | f
 2021-06-12 16:34:52.884 | p3170045 | t
(3 rows)
```

Αν τώρα τρέξουμε την ίδια εντολή 3 φορές με λάθος κωδικό, παρατηρούμε ότι ο χρήστης μας κλειδώνει.

```
[root@snf-883133 ~]# curl -X POST -c cookies 'http://localhost:8080/login' -d 'username=p3170045&password=99101'
{"exception":"Bad credentials","timestamp":1623530264712}
[root@snf-883133 ~]# curl -X POST -c cookies 'http://localhost:8080/login' -d 'username=p3170045&password=99101'
{"exception":"Bad credentials","timestamp":1623530266854}
[root@snf-883133 ~]# curl -X POST -c cookies 'http://localhost:8080/login' -d 'username=p3170045&password=99101'
{"exception":"Your account has been locked due to 3 failed attempts. It will be unlocked after 1 minute.","timestamp":1623530268219}
[root@snf-883133 ~]#
```

```
GDPR=# SELECT * FROM users;
 username |                          password                          |   description    | failed_attempts | locked |       lock_time        | enabled |     pwd_last_modified
----------+------------------------------------------------------------+------------------+-----------------+--------+------------------------+---------+----------------------------
 admin    | $2a$06$AlWboqalGeYilvPSWVkt3uAKTghqSiUdzLw2iW7O2ybVNjmPwHuD2 | Priviledged user |               0 | f      |                        | t       | 2021-06-12 08:05:13.433755
 p3170045 | $2a$06$tKUNYMe.JnFlpEyrfXloy.9dZ5DVNDm5vUu0rBaQauwvURvjBepBi | This is my user  |               2 | t      | 2021-06-12 16:37:48.086 | t       | 2021-06-12 08:05:13.433755
(2 rows)
```

```
GDPR=# SELECT * FROM logging;
   login_attempt_time     | username | success
--------------------------+----------+---------
 2021-06-12 16:25:09.888 | p3170045 | t
 2021-06-12 16:29:59.009 | p3170045 | f
 2021-06-12 16:34:52.884 | p3170045 | t
 2021-06-12 16:37:44.615 | p3170045 | f
 2021-06-12 16:37:46.755 | p3170045 | f
 2021-06-12 16:37:48.028 | p3170045 | f
(6 rows)
```

Περιμένουμε ένα λεπτό μέχρι ο λογαριασμός μας να ξεκλειδώσει και ξαναδοκιμάζουμε:

```
[root@snf-883133 ~]# curl -X POST -c cookies 'http://localhost:8080/login' -d 'username=p3170045&password=991015'
{"exception":"Your account has been unlocked. Please try to login again.","timestamp":1623530406230}
[root@snf-883133 ~]# curl -X POST -c cookies 'http://localhost:8080/login' -d 'username=p3170045&password=991015'
[root@snf-883133 ~]#
```

```
GDPR=# SELECT * FROM users;
 username |                                                      password                                                      |   description    | failed_attempts | locked | lock_time | enabled |     pwd_last_modified
----------+--------------------------------------------------------------------------------------------------------------------+------------------+-----------------+--------+-----------+---------+----------------------------
 admin    | $2a$06$AlWboqalGeYilvPSWVkt3uAKTghqSiUdzLw2iW7O2ybVNjmPwHuD2 | Priviledged user |               0 | f      |           | t       | 2021-06-12 08:05:13.433755
 p3170045 | $2a$06$tKUNYMe.JnFlpEyrfXloy.9dZ5DVNDm5vUu0rBaQauwvURvjBepBi | This is my user  |               0 | f      |           | t       | 2021-06-12 08:05:13.433755
(2 rows)
```

```
GDPR=# SELECT * FROM logging;
   login_attempt_time    | username | success
-------------------------+----------+---------
 2021-06-12 16:25:09.888 | p3170045 | t
 2021-06-12 16:29:59.009 | p3170045 | f
 2021-06-12 16:34:52.884 | p3170045 | t
 2021-06-12 16:37:44.615 | p3170045 | f
 2021-06-12 16:37:46.755 | p3170045 | f
 2021-06-12 16:37:48.028 | p3170045 | f
 2021-06-12 16:40:05.68  | p3170045 | f
 2021-06-12 16:40:29.8   | p3170045 | t
(8 rows)
```

Στο παραπάνω output δεν φαίνονται τα μηνύματα που είδαμε προηγουμένως κατά το επιτυχές login, καθώς αυτά παρουσιάζονται απλά ως strings στον χρήστη και όχι ως responses στο αίτημα που υπέβαλε.

Το εκτελέσιμο αρχείο καθώς και ο (συμπιεσμένος) κώδικας της εφαρμογής με τα ερωτήματα SQL βρίσκονται στο directory /var/webapp