ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΑΝΑΠΤΥΞΗ & ΑΣΦΑΛΕΙΑ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ασφάλεια Λογισμικού και Δικτύων

Τίμια Ψηφιακά Ζάρια

Ευγένιος Γκρίτσης, F3312306

Ηλίας Παναγόπουλος, F3312409

Κρυπτογραφικό Πρωτόκολλο BitCommitment

OpenSSL

Για την δημιουργία CA, CSR και SSL Certificate με Openssl ακολουθούν τα εξής βήματα:

1. Δημιουργία Ιδιωτικού Κλειδιού

openssl genrsa -out domain.key

2. Δημιουργία Αιτήματος Υπογραφής Πιστοποιητικού (CSR)

openssl req -key domain.key -new -out domain.csr

Ρυθμίσεις:

Country Name (2 letter code) [AU]:GR

State or Province Name (full name) [Some-State]:Attica

Locality Name (eg, city) []:Athens

Organization Name (eg, company) [Internet Widgits Pty Ltd]:aueb-casino

Organizational Unit Name (eg, section) []:cybersecurity

Common Name (e.g. server FQDN or YOUR name) []:localhost

Email Address []:example@aueb.gr

3. Δημιουργία Root CA (Αρχής Πιστοποίησης)

openssl req -x509 -sha256 -days 1825 -newkey rsa:2048 -keyout rootCA.key -out rootCA.crt

PEM pass phrase: aueb123

4. Προσθήκη του CA Certificate στα Trusted Certificates

sudo apt install -y ca-certificates

sudo cp rootCA.crt /usr/local/share/ca-certificates

sudo update-ca-certificates

5. Υπογραφή του CSR με την Root CA

Δημιουργήσαμε ένα configuration file (domain.ext) με τις παρακάτω ρυθμίσεις

```
ilias@ubuntu-vm:~/Desktop/netSec$ cat domain.ext
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
subjectAltName = @alt_names
[alt_names]
DNS.1 = localhost
```

6. Υπογραφή του CSR με το Root CA και το ιδιωτικό της κλειδί

openssl x509 -req -CA rootCA.crt -CAkey rootCA.key -in domain.csr -out domain.crt -days 365 -CAcreateserial -extfile domain.ext

6. Επαλήθευση πιστοποιητικού

openssl x509 -text -noout -in domain.crt

7. Μεταφορά Πιστοποιητικών στον NGINX Server

sudo mkdir -p /etc/nginx/ssl

sudo cp domain.crt domain.key rootCA.crt /etc/nginx/ssl/

8. Δημιουργία Full Chain Certificate

Ορισμένοι browsers και clients απαιτούν την πλήρη αλυσίδα πιστοποιητικών. Για αυτόν τον λόγο, συνενώσαμε το domain.crt με το rootCA.crt:

cat /etc/nginx/ssl/domain.crt /etc/nginx/ssl/rootCA.crt > /etc/nginx/ssl/full_chain.crt

9. Αλλαγή Δικαιωμάτων Πρόσβασης

sudo chmod 644 /etc/nginx/ssl/*

10. Ρύθμιση HTTPS & Ανακατεύθυνση HTTP -> HTTPS

Για να σερβίρει ο NGINX server το πιστοποιητικό μας σε HTTPS και να ανακατευθύνει τα HTTP σε HTTPS προστέθηκαν οι εξής αλλαγές στο /etc/nginx/sites-available/default:

```
# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        #Redirect all HTTP requests to HTTPS
        return 301 https://$host$request_uri/;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }
```

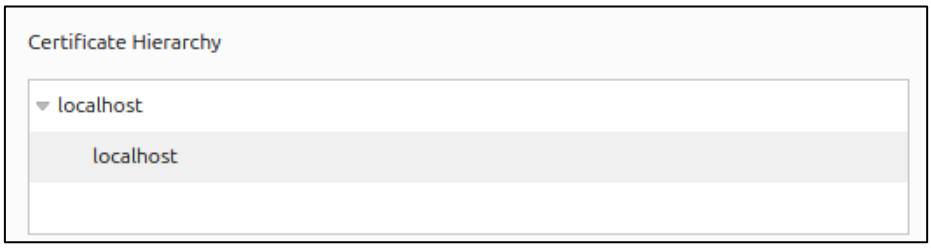
```
#Server HTTPS with SSL
server {
        listen 443 ssl http2;
        listen [::]:443 ssl http2;
        server_name localhost;

        # SSL Certificates
        ssl_certificate /etc/nginx/ssl/full_chain.crt;
        ssl_certificate_key /etc/nginx/ssl/domain.key;
        ssl_trusted_certificate /etc/nginx/ssl/rootCA.crt;

        # Secure SSL settings
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;

        # Root directory and default index file
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        location / {
            try_files $uri $uri/ =404;
        }
}
```

11. Έλεγχος της Αλυσίδας Πιστοποιητικών

openssl s_client -connect localhost:443 -CAfile /etc/nginx/ssl/rootCA.crt

```
ilias@ubuntu-vm:~/Desktop/netSec$ openssl s_client -connect localhost:443 -CAfile /etc/nginx/ssl/rootCA.crt
CONNECTED(00000003)
Can't use SSL_get_servername
depth=1 C = GR, ST = Attica, L = Athens, O = aueb-casino, OU = cybersecurity, CN = localhost, emailAddress = example@aueb.gr
verify return:1
depth=0 C = GR, ST = Attica, L = Athens, O = aueb-casino, OU = cybersecurity, CN = localhost, emailAddress = example@aueb.gr
verify return:1
---
Certificate chain
 0 s:C = GR, ST = Attica, L = Athens, O = aueb-casino, OU = cybersecurity, CN = localhost, emailAddress = example@aueb.gr
   i:C = GR, ST = Attica, L = Athens, O = aueb-casino, OU = cybersecurity, CN = localhost, emailAddress = example@aueb.gr
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Mar  9 11:29:31 2025 GMT; NotAfter: Mar  9 11:29:31 2026 GMT
 1 s:C = GR, ST = Attica, L = Athens, O = aueb-casino, OU = cybersecurity, CN = localhost, emailAddress = example@aueb.gr
   i:C = GR, ST = Attica, L = Athens, O = aueb-casino, OU = cybersecurity, CN = localhost, emailAddress = example@aueb.gr
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Mar  9 11:22:40 2025 GMT; NotAfter: Mar  8 11:22:40 2030 GMT
---
```

Πηγαίνοντας στο http://localhost μας μεταφέρει αυτόματα στο https όπου μπορούμε να επιβεβαιώσουμε πως το SSL Certificate chain εμφανίζεται σωστά.



Αυθεντικοποίηση και ΒΔ

Η βάση η οποία χρησιμοποιήθηκε κατά την υλοποίηση της εργασίας, βρίσκεται τοπικά στον ubuntu server που στήθηκε και φαίνεται στα παρακάτω screenshots.

Username: postgres

Password: password

DB name: GDPR

Ακολουθούν οι εντολές σε PostgreSQL για την δημιουργία της βάσης GDPR, του πίνακα users και των 2 χρηστών:

```
# Create database "gdpr"
CREATE DATABASE GDPR;
# Connect to gdpr
\c gdpr

# Create table users
CREATE TABLE users (
   id SERIAL PRIMARY KEY,       -- Auto-incrementing ID for each user
   first_name VARCHAR(100) NOT NULL,   -- First name
   last_name VARCHAR(100) NOT NULL,    -- Last name
   username VARCHAR(100) UNIQUE NOT NULL,  -- Username must be unique
   password VARCHAR(255) NOT NULL      -- Password field to store encoded password
);

# Insert user admin
GDPR=# INSERT INTO users (first_name, last_name, username, password) VALUES ('Admin',
'Admin', 'admin', '$2a$12$4uWoFJokQGVs8sEhxY3GQ.FEtg/PuU1jKnk6.bcLfRPekEnk66atK');
```

 # Insert another user

GDPR=# INSERT INTO users (first_name, last_name, username, password) VALUES ('F3312306', 'F3312409', 'F3312306F3312409', '$2a$12$Bc7kocAYSOfzMK.86SSV1u4SnDrJZcH509cfkqWYFrlzCUYIfkrbS');

```
GDPR=# select * from users;
 id | first_name | last_name |     username     |                          password
----+------------+-----------+------------------+------------------------------------------------------------
  1 | Admin      | Admin     | admin            | $2a$12$4uWoFJokQGVs8sEhxY3GQ.FEtg/PuU1jKnk6.bcLfRPekEnk66atK
  2 | F3312306   | F3312409  | F3312306F3312409 | $2a$12$Bc7kocAYSOfzMK.86SSV1u4SnDrJZcH509cfkqWYFrlzCUYIfkrbS
(2 rows)
```

Η ενδεδειγμένη λύση για την αποθήκευση των κωδικών χρηστών στη βάση δεδομένων είναι η χρήση ισχυρής κρυπτογραφικής συνάρτησης κατακερματισμού (hashing) με salt. Αυτό εξασφαλίζει ότι ακόμα και αν η βάση δεδομένων παραβιαστεί, οι κωδικοί δεν θα είναι ανακτήσιμοι σε απλή μορφή (plaintext).

Προτεινόμενη Λύση: Χρήση του BCrypt

Το **BCrypt** είναι ένας αλγόριθμος hashing που προσφέρει προσαρμοσμένη πολυπλοκότητα (cost factor) και ενσωματωμένο salt, κάνοντάς τον ανθεκτικό σε επιθέσεις brute force και rainbow tables.

Στο SecurityConfig.java έχει δημιουργηθεί το Bean που θα περιέχει το BCrypt:

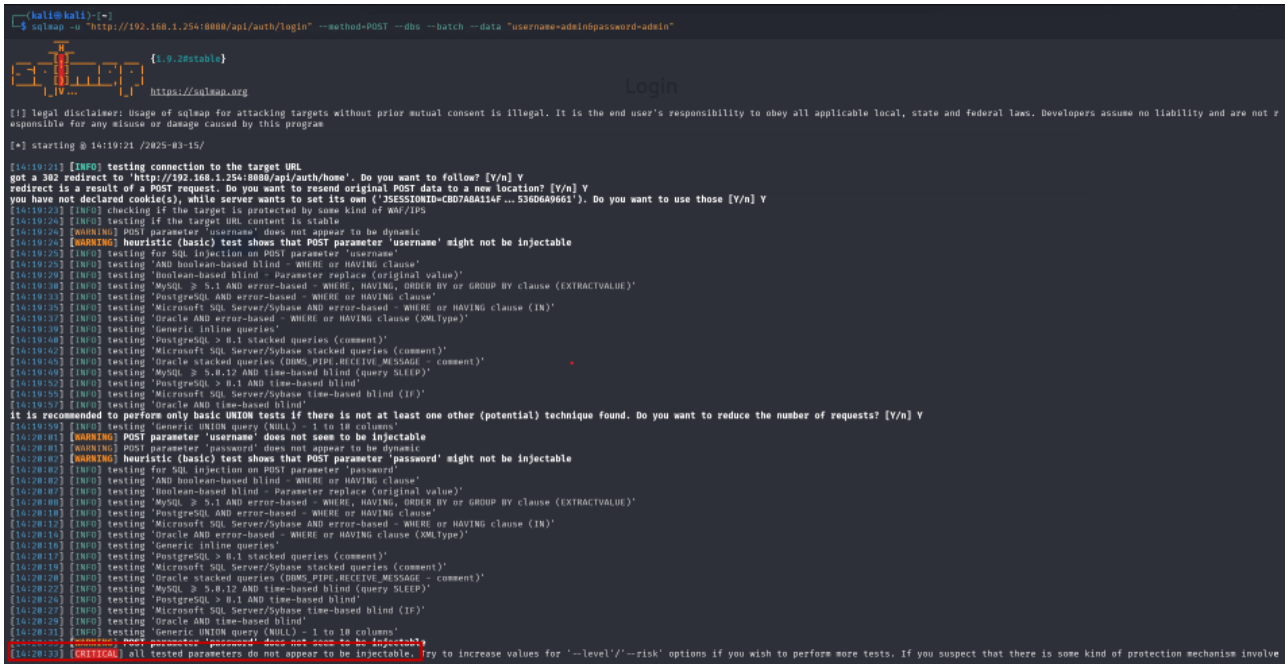
```
44      @Bean
45      public PasswordEncoder passwordEncoder() {
46          return new BCryptPasswordEncoder();
47      }
```

Στο UserServiceImpl.java που είναι υπεύθυνο για την αποθήκευση του νέου χρήστη, γίνεται χρήση του BCrypt για να γίνει η αποθήκευση του κωδικού στη βάση:

```
23      @Override
24      public User save(UserDto userDto) {
25          User user = new User(userDto.getFirst_name(),
26                               userDto.getLast_name(),
27                               userDto.getUsername(),
28                               passwordEncoder.encode(userDto.getPassword()));
29          return userRepository.save(user);
```

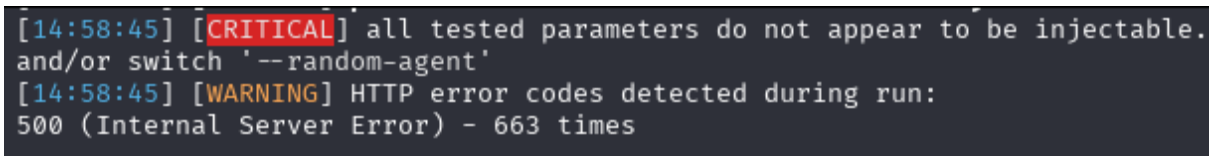
SQL INJECTION

LOGIN FORM:

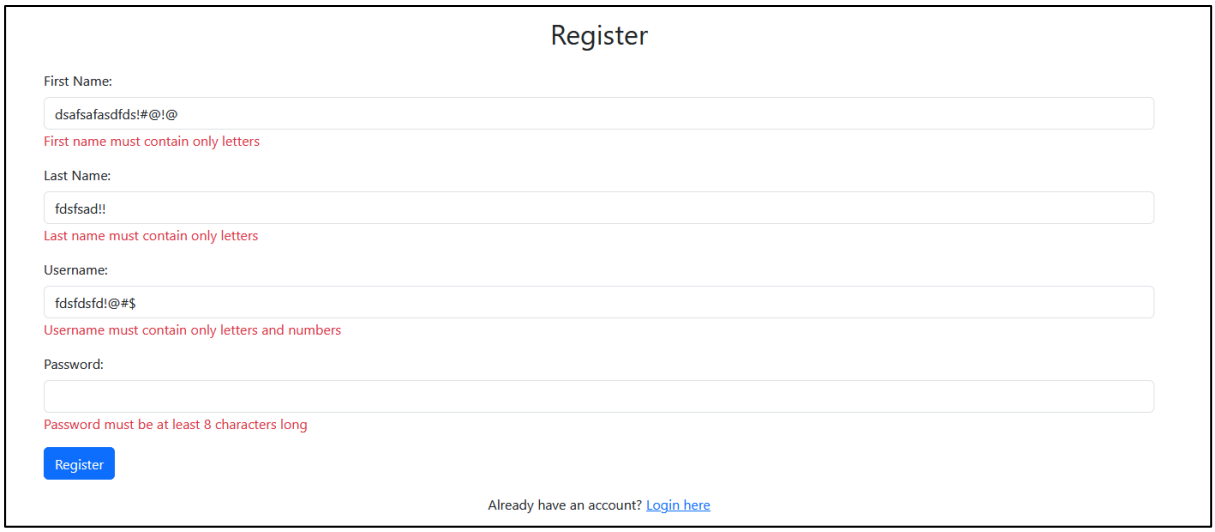


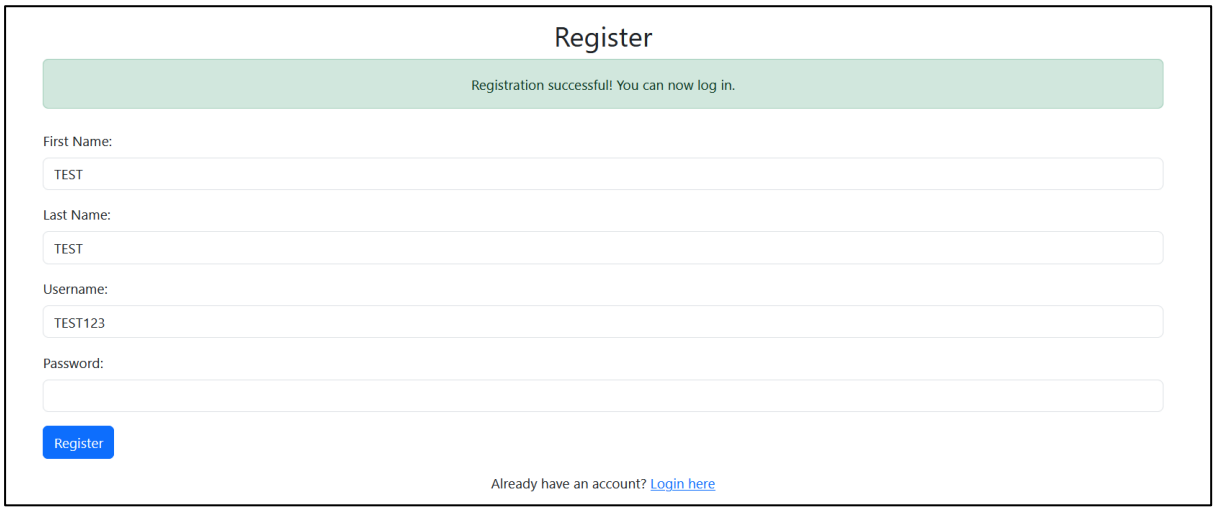
REGISTRATION FORM:

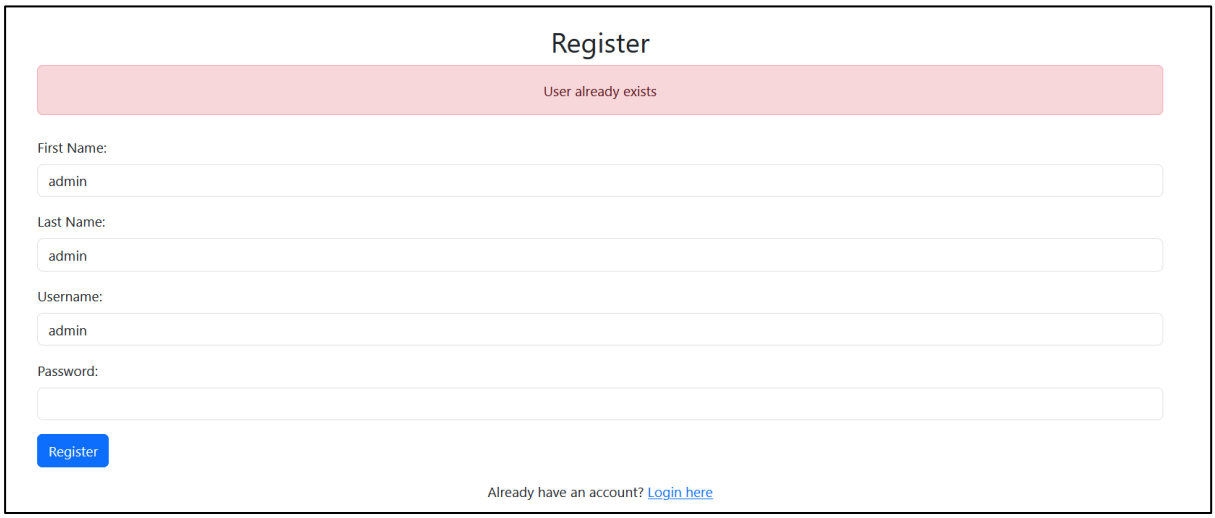
- **--level=5**: Increases the test depth.
- **--risk=3**: Higher risk of testing aggressive payloads.
- **--technique=BEUSTQ**: Specifies which techniques to test for (e.g., Boolean-based, Error-based, Union-based, etc.).



User Input Sanitation:







```java
public class UserDto {  8 usages  eGkritsis *

    @NotBlank(message = "Password is required")  3 usages
    @Size(min = 8, message = "Password must be at least 8 characters long")
    private String password;

    @NotBlank(message = "Username is mandatory")  3 usages
    @Size(min = 3, max = 20, message = "Username must be between 3 and 20 characters")
    @Pattern(regexp = "^[a-zA-Z0-9]+$", message = "Username must contain only letters and numbers")
    private String username;

    @NotBlank(message = "First name is required")  3 usages
    @Pattern(regexp = "^[A-Za-z]+$", message = "First name must contain only letters")
    @Size(min = 2, max = 50, message = "First name must be between 2 and 50 characters")
    private String first_name;

    @NotBlank(message = "Last name is required")  3 usages
    @Pattern(regexp = "^[A-Za-z]+$", message = "Last name must contain only letters")
    @Size(min = 2, max = 50, message = "Last name must be between 2 and 50 characters")
    private String last_name;
```

DEPLOYMENT:

mvn clean package -> Δημιουργείται το .jar

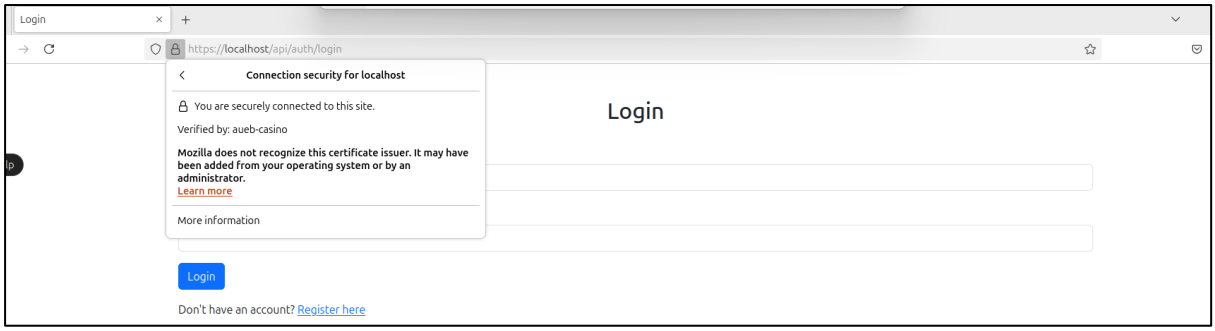
java -jar demo-0.0.1-SNAPSHOT.jar -> Τρέχουμε το Spring Boot application

Τρέχει σωστά:

```
2025-03-16T17:54:34.237+02:00  INFO 10789 --- [demo] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-03-16T17:54:34.343+02:00  INFO 10789 --- [demo] [           main] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 16.8
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-03-16T17:54:35.886+02:00  INFO 10789 --- [demo] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform availab
latform' to enable JTA platform integration)
2025-03-16T17:54:35.968+02:00  INFO 10789 --- [demo] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFacto
2025-03-16T17:54:36.080+02:00  WARN 10789 --- [demo] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled
ries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-03-16T17:54:36.826+02:00  INFO 10789 --- [demo] [           main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager confi
ith name customUserDetailsService
2025-03-16T17:54:37.679+02:00 DEBUG 10789 --- [demo] [           main] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with filte
ManagerIntegrationFilter, SecurityContextHolderFilter, HeaderWriterFilter, LogoutFilter, JwtRequestFilter, UsernamePasswordAuthenticationFilter, Req
tHolderAwareRequestFilter, AnonymousAuthenticationFilter, ExceptionTranslationFilter, AuthorizationFilter
2025-03-16T17:54:38.309+02:00  INFO 10789 --- [demo] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http)
2025-03-16T17:54:38.323+02:00  INFO 10789 --- [demo] [           main] c.a.casino.netsec.demo.DemoApplication   : Started DemoApplication in 9.654 s
Hello World
```

Στο /etc/nginx/sites-available/default προσθέτουμε τις κατάλληλες ρυθμίσεις για να γίνονται forward τα requests στο Spring Boot App.

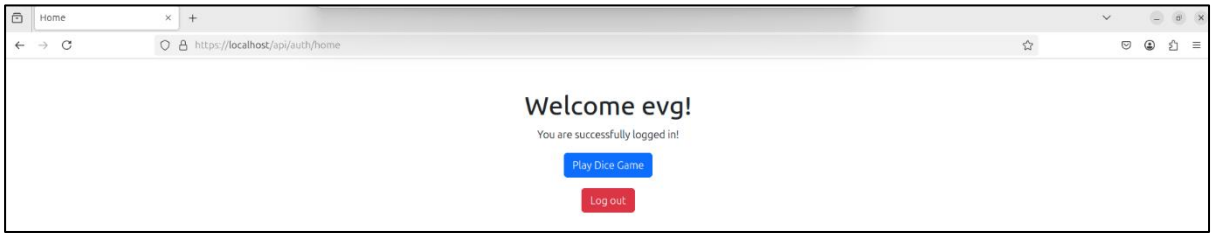
```
location / {
    proxy_pass http://localhost:8080/; # Forward requests to Spring Boot App
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    #try_files $uri $uri/ =404;
}
```



Η εφαρμογή ανοίγει σε https υποχρεωτικά

SCREENSHOTS:

Η σελίδα /home, αφού συνδεθεί επιτυχώς ο χρήστης. Οι επιλογές του χρήστη είναι να προχωρήσει στο παιχνίδι του ζαριού ή να αποσυνδεθεί από την εφαρμογή. Και οι δύο λειτουργίες μπορούν να γίνουν μέσω των κατάλληλα διαμορφωμένων κουμπιών "Play Dice Game" και "Log out" αντίστοιχα.



Εάν ο χρήστης επιλέξει να παίξει το παιχνίδι τότε γίνεται ανακατεύθυνση στο /game/play όπου εμφανίζεται το τελικό αποτέλεσμα και το τι νούμερο "έπαιξε" ο υπολογιστής και ο χρήστης. Για λόγους επαλήθευσης ότι το πρωτόκολλο εφαρμόστηκε επιτυχώς σύμφωνα με τις οδηγίες της εκφώνησης, εμφανίζονται και οι τιμές των ενδιάμεσων βημάτων. Τέλος ο χρήστης έχει την επιλογή να ξανά παίξει ή να πάει στην αρχική σελίδα της εφαρμογής. Αυτές οι λειτουργίες επιτυχάνονται μέσω του "Play Again" και του "Back to Home". Ανάλογα το τις τιμές των ζαριών και το τελικό απότελεσμα, θα εμφανιστεί στο χρήστη το κατάλληλο γραφικό.

Game Results

Intermediate Steps

Category	Details
User's Random String (rA)	SNrFLtbJ4r
Server's Random String (rS)	5muKPdjehM
User's Number	2
Server's Number	4
Commitment Hash (hCommit)	`023f3e58cfe902fff4821ad612d99b6a64b5c6a6823d1d6e960af6e4c1098bf7`
Revealed Hash (h2)	`023f3e58cfe902fff4821ad612d99b6a64b5c6a6823d1d6e960af6e4c1098bf7`

User loses!

Play Again

Back to Home

Game Results

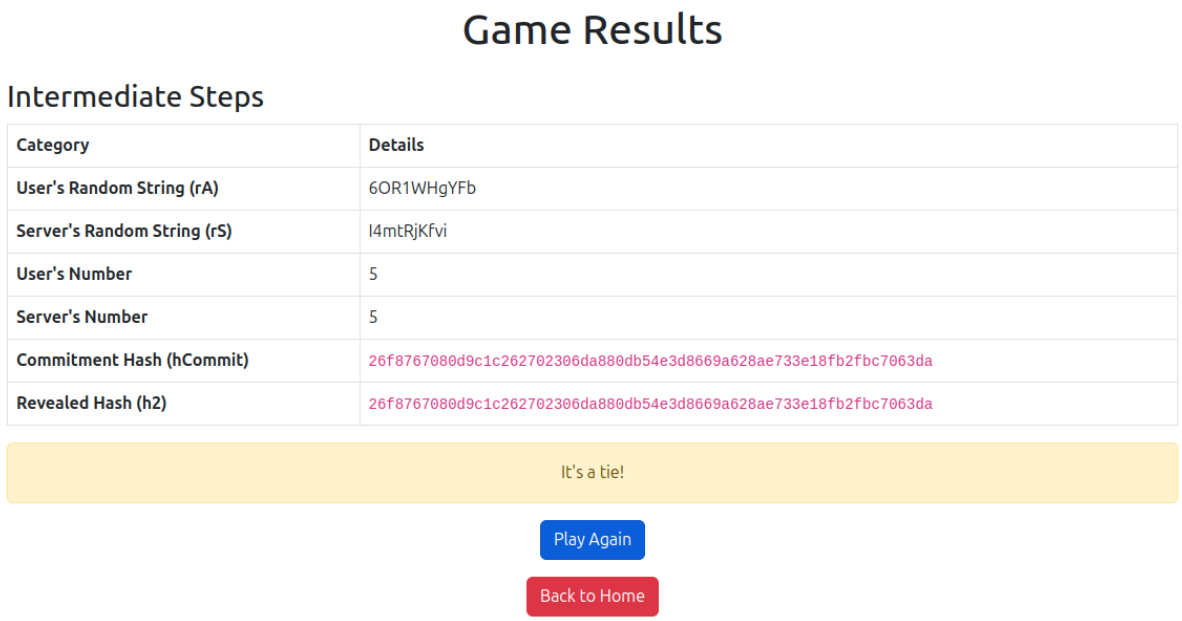
Intermediate Steps

Category	Details
User's Random String (rA)	ZtbLHdDO3z
Server's Random String (rS)	DmQdMYxbZQ
User's Number	6
Server's Number	4
Commitment Hash (hCommit)	`c7f375ed6f891d1d43a2ca0f55c7712185b74a6182109b59e0be8a7746347b2a`
Revealed Hash (h2)	`c7f375ed6f891d1d43a2ca0f55c7712185b74a6182109b59e0be8a7746347b2a`

User wins!

Play Again

Back to Home



Ανάλυση του Πρωτοκόλλου

Στόχος του Πρωτοκόλλου

Το πρωτόκολλο επιτρέπει σε έναν παίκτη (Anna) και έναν server να παίξουν ένα δίκαιο παιχνίδι με ένα ζάρι (1-6), εξασφαλίζοντας ότι κανείς δεν μπορεί να αλλάξει την επιλογή του μετά την αποκάλυψή της.

Βήματα του Πρωτοκόλλου

Step 1: Γεννήτρια τυχαίων στοιχείων

- Παράγεται μία τυχαία συμβολοσειρά για τον παίκτη και για τον server:
    - rA (τυχαία συμβολοσειρά του παίκτη)
    - rS (τυχαία συμβολοσειρά του server)

Αυτό διασφαλίζει ότι κάθε εκτέλεση του πρωτοκόλλου είναι μοναδική και μη επαναλήψιμη.

```
// Step 1: Generate random strings
String rA = gameService.generateRandomString();
String rS = gameService.generateRandomString();
```

```
public String generateRandomString() {
    return RandomStringUtils.randomAlphanumeric(count:10);
}
```

Step 2: Επιλογή αριθμού από τον παίκτη

- Επιλέγεται ένας τυχαίος αριθμός από 1 έως 6

```
// Step 2: User chooses a number
int playerChoice = gameService.generateRandomNumber();
```

```
19      public int generateRandomNumber() {
20          return secureRandom.nextInt(6)+1;
21      }
```

Step 3: Ο server αποκαλύπτει το rS στον παίκτη

- Ο **server** στέλνει την τυχαία συμβολοσειρά του (rS) στον παίκτη.

Αυτό γίνεται ώστε να διασφαλιστεί ότι το rS δεν μπορεί να αλλάξει αργότερα.

Step 4: Ο παίκτης υπολογίζει το Commitment

- Ο παίκτης δημιουργεί μια δέσμευση (commitment) με βάση τα παρακάτω στοιχεία:
  - **τον αριθμό που επέλεξε** (playerChoice)
  - **τη δική του τυχαία συμβολοσειρά** (rA)
  - **την τυχαία συμβολοσειρά του server** (rS)

Δηλαδή:

hCommit=SHA-256(playerChoice||rA||rS)

Αυτό το hash εξασφαλίζει ότι ο παίκτης δεν μπορεί να αλλάξει την επιλογή του αργότερα.

```
// Step 3: Server sends rS to User
// Step 4: User computes the commitment
String commitmentInput = playerChoice + rA + rS;
String hCommit = gameService.computeHash(commitmentInput);
```

```
public String computeHash(String input) {
    return DigestUtils.sha256Hex(input); //
}
```

Step 5: Ο server επιλέγει έναν αριθμό

- Ο server επιλέγει και αυτός έναν αριθμό από 1 έως 6.

```
// Step 5: Server chooses its number
int serverChoice = gameService.generateRandomNumber();
```

Step 6: Ο παίκτης αποκαλύπτει την επιλογή του

- Ο παίκτης στέλνει στον server τα **πραγματικά στοιχεία** που χρησιμοποίησε για τη δέσμευση:
    - playerChoice
    - rA
    - rS

```
// Step 6: User reveals her number (not shown to User)
String revealedInput = playerChoice + rA + rS;
String h2 = gameService.computeHash(revealedInput);
```

Step 7: Ο server ελέγχει την εγκυρότητα της δέσμευσης

- Ο server **υπολογίζει το hash** των αποκαλυφθέντων στοιχείων: $h2=SHA-256(playerChoice||rA||rS)h2 = \text{SHA-256}(\text{playerChoice} || rA || rS)$h2=SHA-256(playerChoice||rA||rS)
- Αν **h2 ≠ hCommit**, τότε σημαίνει ότι ο παίκτης προσπάθησε να αλλάξει την επιλογή του και το παιχνίδι απορρίπτεται ως **άκυρο**.

```
// Step 7: Server verifies the commitment
if (!h2.equals(hCommit)) {
    model.addAttribute(attributeName:"error", attributeValue:"Invalid commitment!");
    return "error"; // Render an error page
}
```

Step 8: Ανακοίνωση αποτελέσματος

- Αν το commitment είναι έγκυρο, συγκρίνονται οι αριθμοί των δύο παικτών (**παίκτης και server**):
    - Αν **playerChoice > serverChoice**, ο παίκτης **κερδίζει**.
    - Αν **playerChoice < serverChoice**, ο server **κερδίζει**.
    - Αν **playerChoice == serverChoice**, υπάρχει **ισοπαλία**.

```
// Step 8: Determine the result
String result = gameService.determineResult(playerChoice, serverChoice);
```

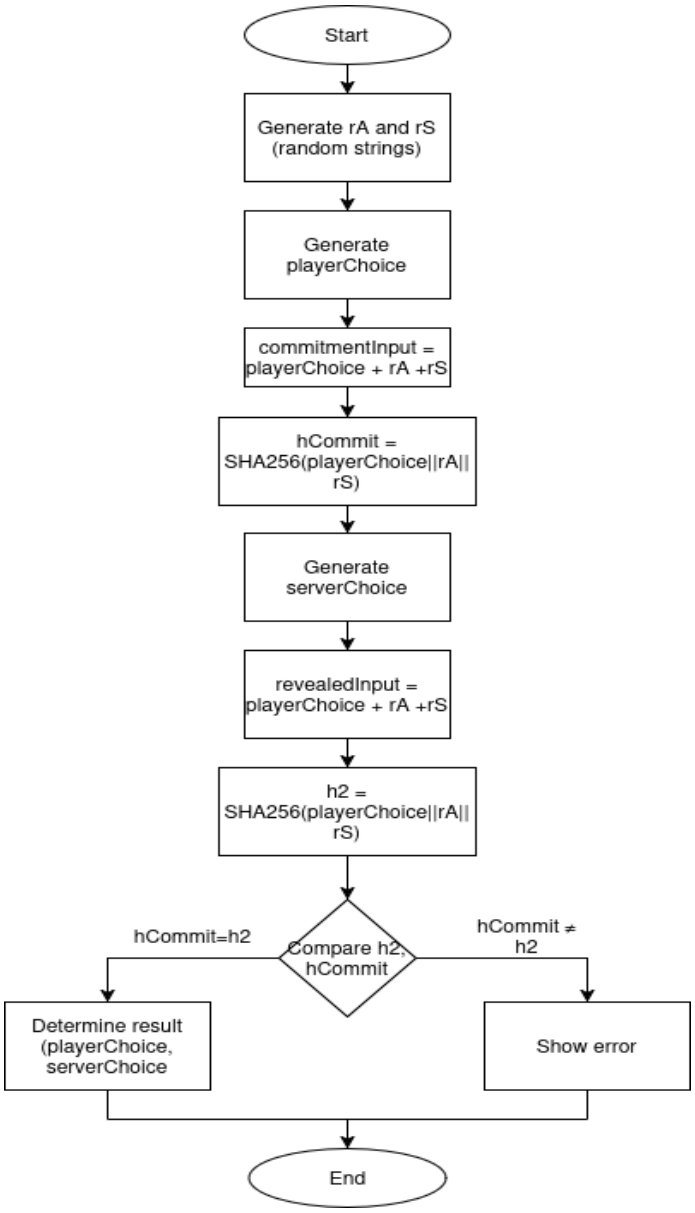
```
public String determineResult(int playerChoice, int serverChoice) {
    if (playerChoice > serverChoice) {
        return "User wins!";
    } else if (playerChoice < serverChoice) {
        return "User loses!";
    } else {
        return "It's a tie!";
    }
}
```

Ασφάλεια & Δικαιοσύνη του Πρωτοκόλλου

Αυτό το πρωτόκολλο **εγγυάται ένα δίκαιο παιχνίδι**, επειδή:

1. **Ο παίκτης δεσμεύεται στην επιλογή της πριν μάθει τον αριθμό του server**, οπότε **δεν μπορεί να αλλάξει** τον αριθμό της εκ των υστέρων.
2. **Ο server δεν μπορεί να επηρεάσει την επιλογή του παίκτη**, γιατί το rS αποκαλύπτεται πριν από τη δέσμευση (commitment).
3. **Η χρήση του SHA-256 διασφαλίζει ότι η δέσμευση είναι μη αναστρέψιμη**, δηλαδή ο παίκτης δεν μπορεί να "πειράξει" το hash ώστε να αλλάξει την επιλογή του.

Διάγραμμα ροής πρωτοκόλλου



Επεξήγηση Διαγράμματος Ροής

1. **Start**
   Η διαδικασία ξεκινά.
2. **Generate rA and rS (random strings)**
   Δημιουργούνται δύο τυχαίες συμβολοσειρές:
   a. rA: προσωπική συμβολοσειρά του χρήστη
   b. rS: συμβολοσειρά του διακομιστή

3. **Generate playerChoice**
   Ο χρήστης επιλέγει έναν αριθμό (1 έως 6) — σε αυτήν την περίπτωση γίνεται τυχαία από τον υπολογιστή.
4. **commitmentInput = playerChoice + rA + rS**
   Ορίζεται η συμβολοσειρά δέσμευσης, συνδυάζοντας την επιλογή του χρήστη με τα δύο random strings.
5. **hCommit = SHA256(playerChoice||rA||rS)**
   Υπολογίζεται το hash της συμβολοσειράς, το οποίο λειτουργεί ως "κλείδωμα" για την επιλογή του χρήστη.
6. **Generate serverChoice**
   Ο server επιλέγει επίσης έναν τυχαίο αριθμό (1 έως 6).
7. **revealedInput = playerChoice + rA + rS**
   Ο χρήστης "αποκαλύπτει" τα στοιχεία που είχε χρησιμοποιήσει αρχικά.
8. **h2 = SHA256(playerChoice||rA||rS)**
   Υπολογίζεται ξανά το hash με βάση τα “αποκαλυφθέντα” δεδομένα για να γίνει επαλήθευση.
9. **Compare h2 with hCommit**
   Γίνεται σύγκριση του νέου hash (h2) με το αρχικό (hCommit):
   a. Αν **είναι ίσα**: η δέσμευση θεωρείται έγκυρη.
   b. Αν **είναι διαφορετικά**: σημαίνει παραποίηση ή σφάλμα και εμφανίζεται σφάλμα.
10. **Ανάλογα το αποτέλεσμα της σύ**
    a. **Determine result (playerChoice, serverChoice)**
       Αν η δέσμευση είναι έγκυρη, συγκρίνεται η επιλογή του χρήστη με αυτή του διακομιστή και βγαίνει αποτέλεσμα. Ο χρήστης κερδίζει / χάνει / βγαίνει ισοπαλία
    b. **Show error**
       Αν το hash δεν ταιριάζει, εμφανίζεται σελίδα σφάλματος.
11. **End**
    Τέλος της διαδικασίας.