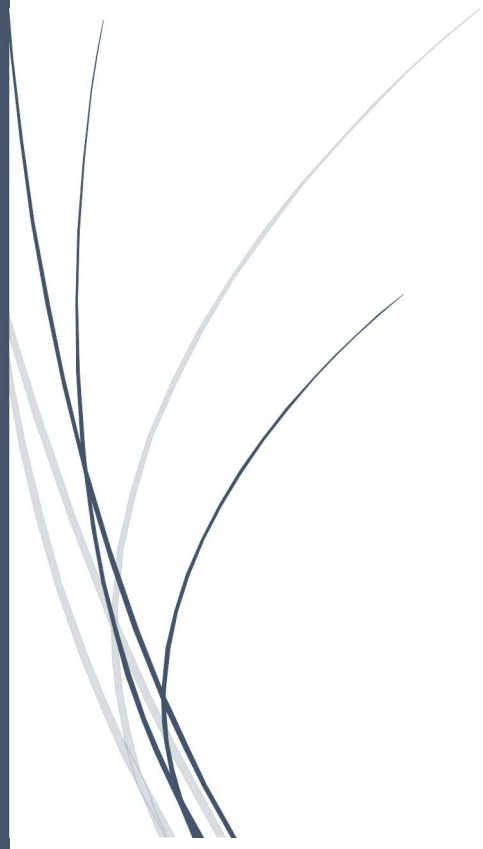
10/26/2025

Assignment 03

Identified Vulnerabilities
and Fixes



Submitted By:
EMAN TARIQ 21I-2773
SUMERA MALIK 21I-1579

Introduction:

This report covers ten critical API security areas remediated in the OWASP API Lab. For each control, the report explains the vulnerability, risk, the fix with rationale, the key code changes, and verification steps with screenshots.

Covered Security Controls:

1) Password Security (BCrypt)
2) Access Control (SecurityFilterChain & RBAC)
3) Resource Ownership Enforcement (IDOR prevention)
4) Data Exposure Control (DTOs, least data)
5) Rate Limiting (Bucket4j)
6) Mass Assignment Prevention (explicit DTOs)
7) JWT Hardening (secret/issuer/audience/TTL)
8) Error Handling & Logging (no leaks)
9) Input Validation (Jakarta validation)
10) Testing & Verification (integration tests)

1) Password Security (BCrypt):

Vulnerability & Context

Plaintext or weakly protected passwords enable immediate credential compromise.

Risk/Impact

DB exfiltration yields reusable passwords; offline cracking is trivial if not salted & stretched.

Fix Approach & Rationale

Store BCrypt hashes; verify with PasswordEncoder.matches(); seed demo users with hashed passwords only.

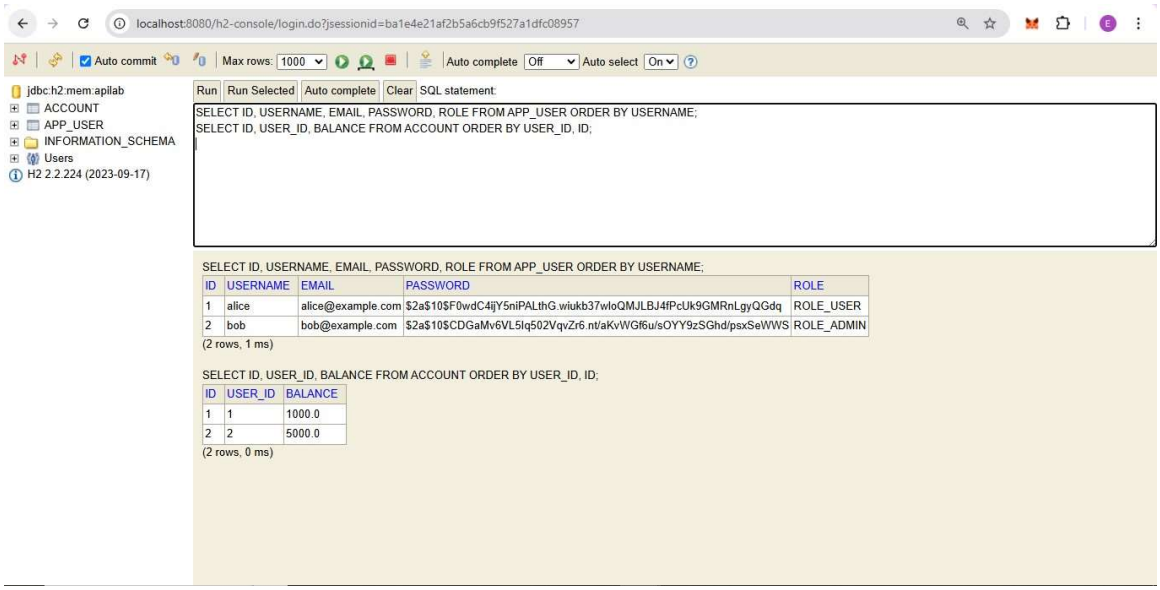
Key Code Changes

- config/SecurityConfig.java — BCryptPasswordEncoder bean
- web/AuthController.java — encode on signup; matches on login config/DataSeeder.java — seed users with BCrypt

Validation / Test Steps

1. Signup, then inspect DB: password column is a BCrypt hash ($2...)
2. Wrong password → 401; no verbose error

Evidence (Screenshots)



Bcrypt-hashes.PNG

```java
 * - Passwords are hashed with BCrypt via PasswordEncoder (no plaintext).
 * - Uses upsert-style logic: creates records only if missing.
 */
@Component
public class DataSeeder implements CommandLineRunner {

    private final AppUserRepository users;
    private final AccountRepository accounts;
    private final PasswordEncoder passwordEncoder;

    public DataSeeder(AppUserRepository users,
                      AccountRepository accounts,
                      PasswordEncoder passwordEncoder) {
        this.users = users;
        this.accounts = accounts;
        this.passwordEncoder = passwordEncoder;
    }

    @Override
    public void run(String... args) {
        // --- Ensure ALICE exists ---
        AppUser alice = upsertUser(
                "alice",
                "alice@example.com",
                "alice123",         // NOTE: will be encoded with BCrypt
                "ROLE_USER"         // normal user
        );

        // --- Ensure BOB exists ---
        AppUser bob = upsertUser(
                "bob",
                "bob@example.com",
                "bob123",           // NOTE: will be encoded with BCrypt
                "ROLE_ADMIN"        // admin (needed for /api/admin/** endpoints)
        );

        // --- Ensure each has at least one account (simple balances for demo) ---
        upsertAccountForUser(alice.getId(), 1000.00);
        upsertAccountForUser(bob.getId(),   5000.00);
    }

    /**
     * Create user if missing; if present, leave as-is (idempotent).
     * Passwords are hashed with BCrypt (no plaintext storage).
     */
    private AppUser upsertUser(String username, String email, String rawPassword, String role) {
        Optional<AppUser> existing = users.findByUsername(username);
        if (existing.isPresent()) {
            return existing.get();
        }
        AppUser u = new AppUser();
        u.setUsername(username);
        u.setEmail(email);
        // BCrypt hash via injected PasswordEncoder bean from SecurityConfig
        u.setPassword(passwordEncoder.encode(rawPassword)); // <-- BCrypt hash, not plaintext
        u.setRole(role);                                    // <-- app uses hasRole("ADMIN") in SecurityConfig
        return users.save(u);
```

code.PNG

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $API = "http://localhost:8080"
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $JsonHeader = @{ "Content-Type" = "application/json" }
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $aliceLoginBody = @{ username="alice"; password="alice123" } | ConvertTo-Json
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $aliceResp = Invoke-RestMethod -Method POST -Uri "$API/api/auth/login" -Headers $
JsonHeader -Body $aliceLoginBody
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $hdrAlice = @{ Authorization = "Bearer " + $aliceResp.token }
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $aliceResp | Select-Object token    # shows token

token
-----
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGljZSIsImlzcyI6ImFwaWxhYiIsImF1ZCI6ImFwaWxhYi1jbGllbnRzIiwicm9sZSI6IlJPTEVfVVNFUiIsImlhdCI6MTc2MDg4OTAwMSwiZXhwIjoxNzYwODg5OTAxfQ....


PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab>
```

login-token.PNG

2) Access Control (SecurityFilterChain & RBAC):

Vulnerability & Context

Overly-permissive permitAll exposed sensitive endpoints.

Risk/Impact

Anonymous access allows enumeration and unauthorized actions.

Fix Approach & Rationale

Deny-by-default; only /api/auth/login & /api/auth/signup are public; RBAC on /api/admin/**; stateless JWT auth.

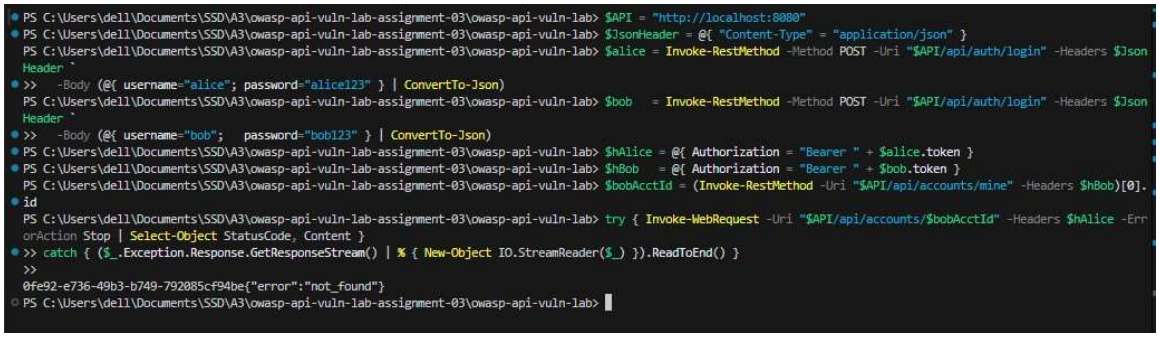
Key Code Changes

 config/SecurityConfig.java — strict authorizeHttpRequests rules security/JwtAuthFilter.java — token verification

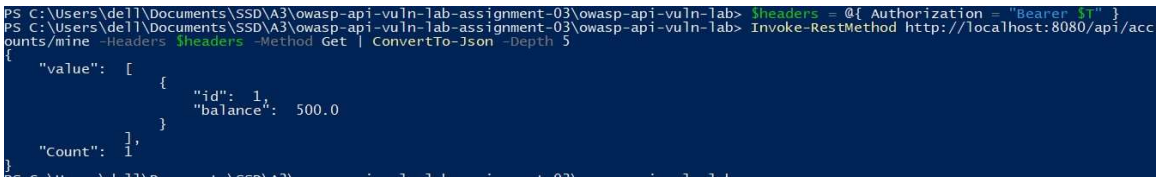
Validation / Test Steps

3. Unauthenticated /api/accounts/mine → 401
4. Non-admin accessing /api/admin/users → 403

Evidence (Screenshots)



Access-control.PNG



Accesscontrol-200.PNG

```
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $res | ConvertTo-Json -Depth 5
{
    "token":  "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGljZSIsImlzcyI6Im93YXNwLWxhYiIsImF1ZCI6Im93YXNwLWNsaWVudHMiLCJyb2xlIjoiUk9MRV9VU0VSIiwiaWF0IjoxNzYwODcxNTQ5LCJleHAiOjE3NjA4NzI0NDl9.dsp2dPa1Tx-VB-gyONQvuEP2d-aOOvZ-7ORsp84YMkY"
}
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $T = $res.token
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $T
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGljZSIsImlzcyI6Im93YXNwLWxhYiIsImF1ZCI6Im93YXNwLWNsaWVudHMiLCJyb2xlIjoiUk9MRV9VU0VSIiwiaWF0IjoxNzYwODcxNTQ5LCJleHAiOjE3NjA4NzI0NDl9.dsp2dPa1Tx-VB-gyONQvuEP2d-aOOvZ-7ORsp84YMkY
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> try { Invoke-RestMethod http://localhost:8080/api/accounts/mine -Method Get -ErrorAction Stop } catch { $_.Exception.Message }
>>
The remote server returned an error: (403) Forbidden.
```

Accesscontrol-403.PNG

```java
/**
 * Central helper to map the authenticated subject (username) -> AppUser / userId.
 * Controllers call this to enforce ownership checks (BOLA).
 */
@Component
public class CurrentUserService {

    private final AppUserRepository users;

    public CurrentUserService(AppUserRepository users) {
        this.users = users;
    }

    /** Returns the authenticated AppUser (or empty if unauthenticated/not found). */
    public Optional<AppUser> current(Authentication auth) {
        if (auth == null || auth.getName() == null) return Optional.empty();
        return users.findByUsername(auth.getName());
    }

    /** Returns the authenticated user's id (or empty). */
    public Optional<Long> currentUserId(Authentication auth) {
        return current(auth).map(AppUser::getId);
    }

    /** Convenience: true if the authenticated user has the given role (e.g., "ADMIN"). */
    public boolean hasRole(Authentication auth, String role) {
        if (auth == null || auth.getAuthorities() == null) return false;
        final String expected = "ROLE_" + role.toUpperCase();
        return auth.getAuthorities().stream().anyMatch(a -> expected.equals(a.getAuthority()));
    }
}
```

code.PNG

3) Resource Ownership Enforcement (IDOR):

Vulnerability & Context

ID-only checks allowed cross-user data access.

Risk/Impact

Attackers could read/modify other users' resources by guessing IDs.

Fix Approach & Rationale

Derive subject userId from token; repository methods like findByIdAndUserId; never trust client userId.

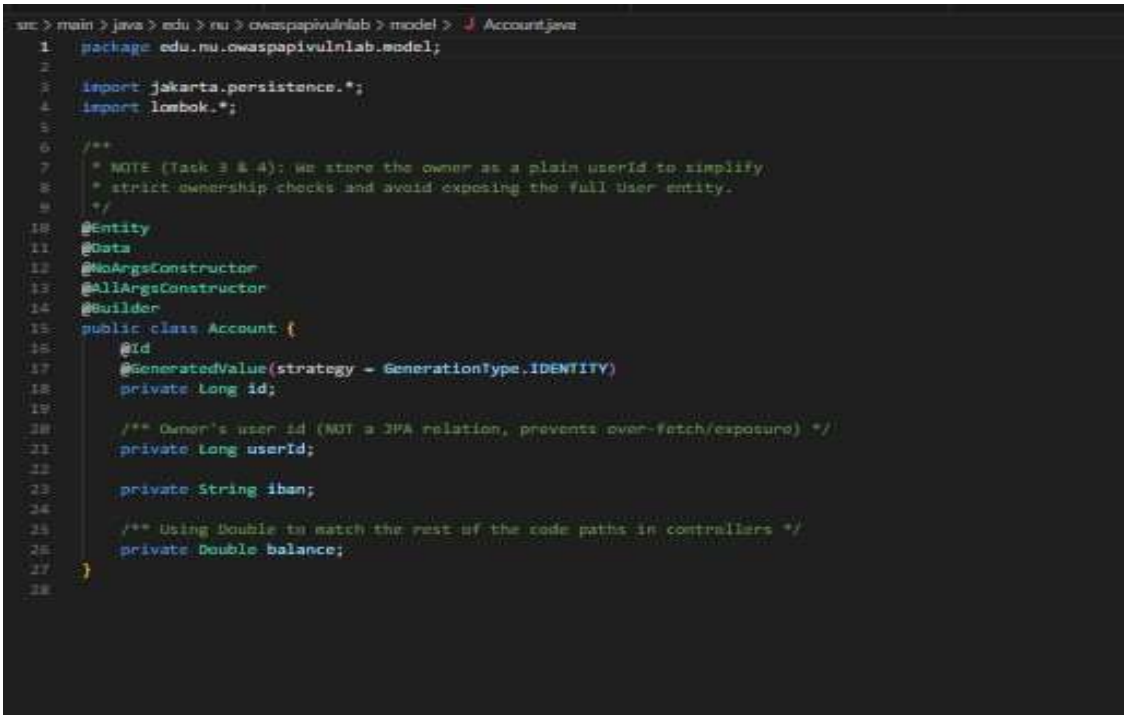
Key Code Changes

- web/AccountController.java — use subject userId everywhere
- repository/AccountRepository.java — findByIdAndUserId

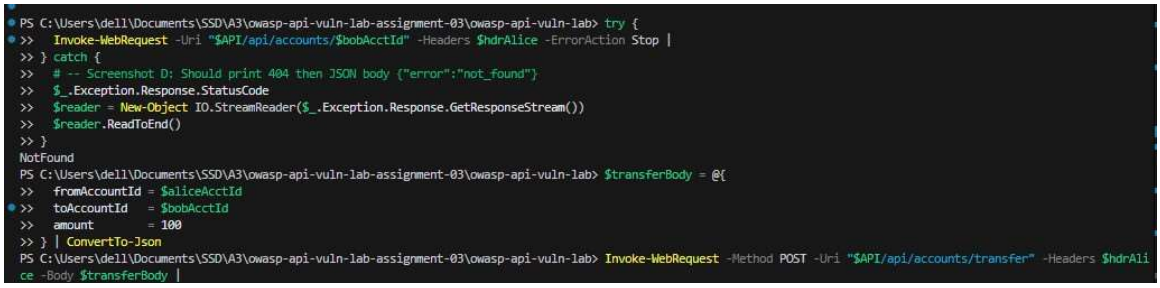
Validation / Test Steps

5. User A requesting B's account id → 404
6. Transfers require ownership of source account

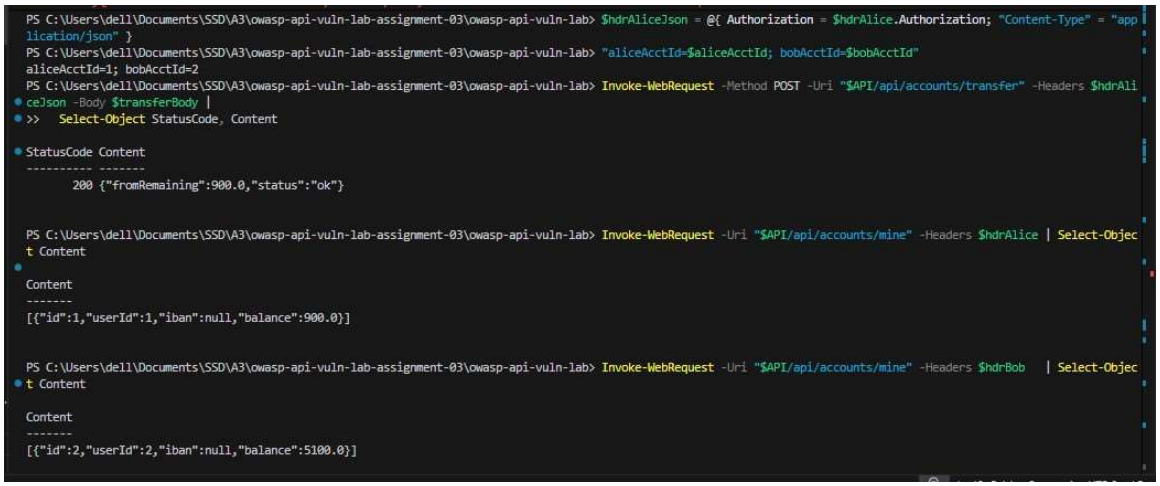
Evidence (Screenshots)



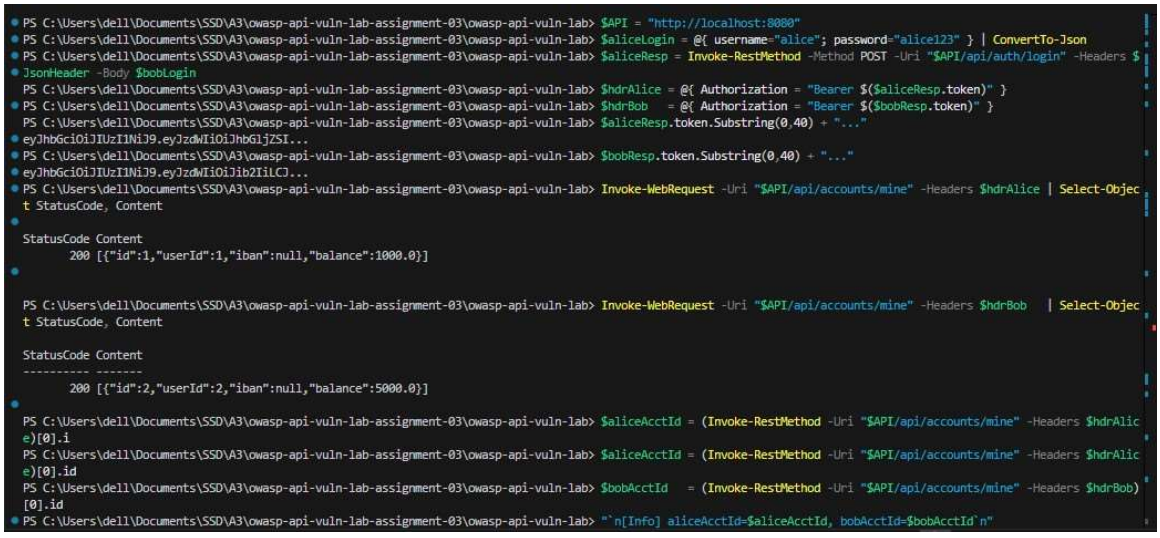
code.PNG



resource-2.PNG



resource-3.PNG



resource-owner.PNG

4) Data Exposure Control (DTOs):

Vulnerability & Context

Returning entities leaked sensitive/internal fields.

Risk/Impact

Leaked fields accelerate privilege escalation and data harvesting.

Fix Approach & Rationale

Return narrow response DTOs; exclude password/roles/flags by construction.

Key Code Changes

- web/dto/*.java — AccountDto, UserDto, TokenResponse
- web/AccountController.java — entity→DTO mapping

Validation / Test Steps

7. GET /api/accounts/mine returns no sensitive fields

Evidence (Screenshots)

```java
package edu.nu.owaspapivulnlab.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.*;
import jakarta.validation.constraints.*;
import lombok.*;

/**
 * User entity.
 * SECURITY: Mark sensitive fields with @JsonIgnore so they never appear in JSON responses,
 * even if someone accidentally returns the entity. Prefer returning DTOs (see later tasks).
 */
@Entity @Data @NoArgsConstructor @AllArgsConstructor @Builder
public class AppUser {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank
    private String username;

    @JsonIgnore              // <-- prevent password from being serialized
    @NotBlank
    private String password;

    @JsonIgnore              // <-- prevent role/isAdmin leaking
    private String role;     // "ROLE_USER" or "ROLE_ADMIN"

    @JsonIgnore
    private boolean isAdmin;

    @Email
    private String email;
}
```

code 2.PNG

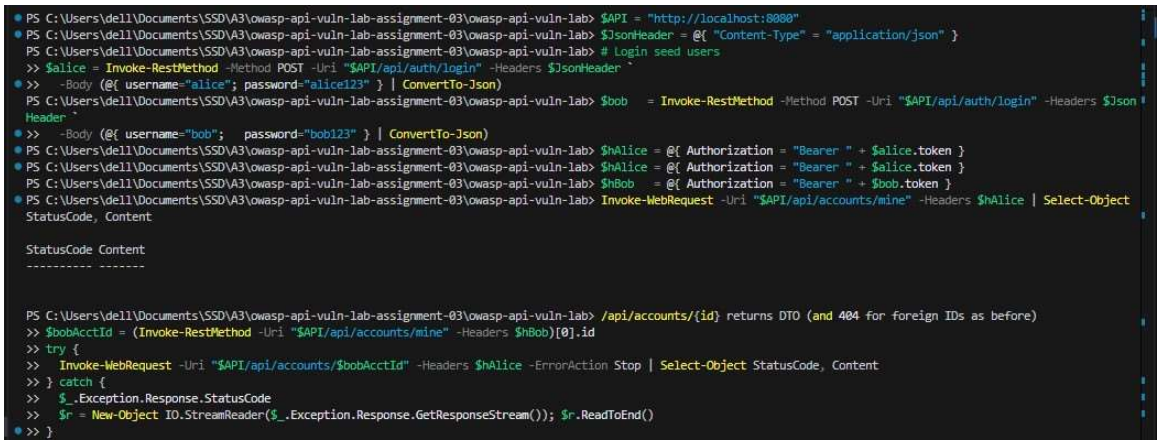
```java
private final AppUserRepository users;
private final CurrentUserService current;
private final PasswordEncoder encoder;

public UserController(AppUserRepository users, CurrentUserService current, PasswordEncoder encoder) {
    this.users = users;
    this.current = current;
    this.encoder = encoder;
}

private static UserDto toDto(AppUser u) {
    return new UserDto(u.getId(), u.getUsername(), u.getEmail()); // Task 4 DTO (no password/role/isAdmin)
}

@GetMapping("/me")
public ResponseEntity<?> me(Authentication auth) {
    return current.current(auth)
            .<ResponseEntity<?>>map(u -> ResponseEntity.ok(toDto(u)))
            .orElseGet(() -> ResponseEntity.status(401).body(Map.of("error","unauthenticated")));
}

@GetMapping("/{id}")
public ResponseEntity<?> get(@PathVariable("id") Long id, Authentication auth) {
    Long callerId = current.currentUserId(auth).orElse(null);
    if (callerId == null) return ResponseEntity.status(401).body(Map.of("error","unauthenticated"));

    boolean isAdmin = current.hasRole(auth, "ADMIN");
    if (!isAdmin && !id.equals(callerId)) {
        return ResponseEntity.status(403).body(Map.of("error","forbidden"));
    }

    return users.findById(id)
            .<ResponseEntity<?>>map(u -> ResponseEntity.ok(toDto(u)))
            .orElseGet(() -> ResponseEntity.status(404).body(Map.of("error","not_found")));
}
```

code 3.PNG

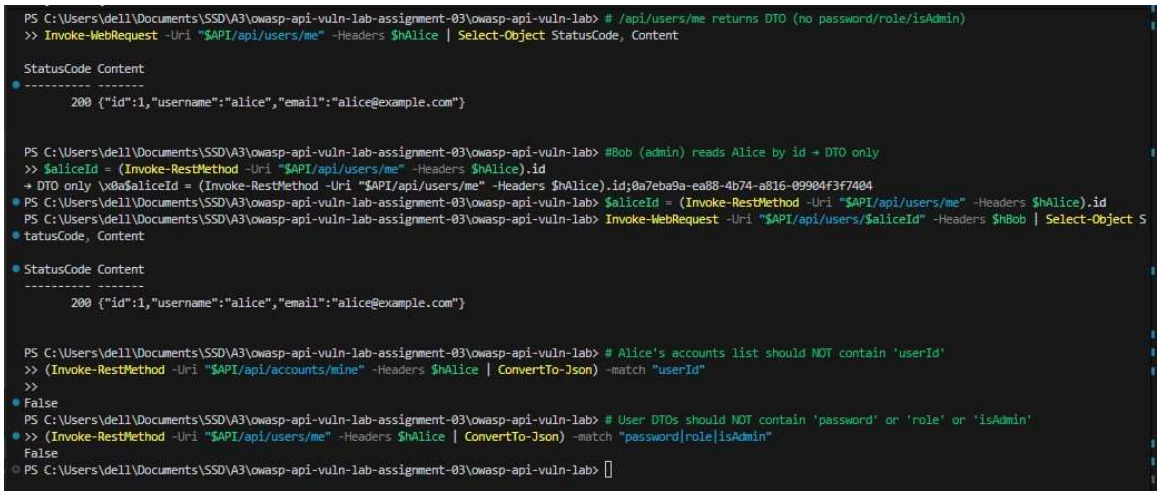
src > main > java > edu > nu > owaspapivulnlab > model > Account.java

```java
package edu.nu.owaspapivulnlab.model;

import jakarta.persistence.*;
import lombok.*;

/**
 * NOTE (Task 3 & 4): We store the owner as a plain userId to simplify
 * strict ownership checks and avoid exposing the full User entity.
 */
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Account {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    /** Owner's user id (NOT a JPA relation, prevents over-fetch/exposure) */
    private Long userId;

    private String iban;

    /** Using Double to match the rest of the code paths in controllers */
    private Double balance;
}
```

code.PNG



data exp 1.PNG



data exp 2.PNG

5) Rate Limiting (Bucket4j):

Vulnerability & Context

No throttling allowed brute force and abusive actions.

Risk/Impact

Credential stuffing or transfer spamming possible without limits.

Fix Approach & Rationale

Apply per-IP/per-user buckets on login/signup/transfer; on exceed → 429.

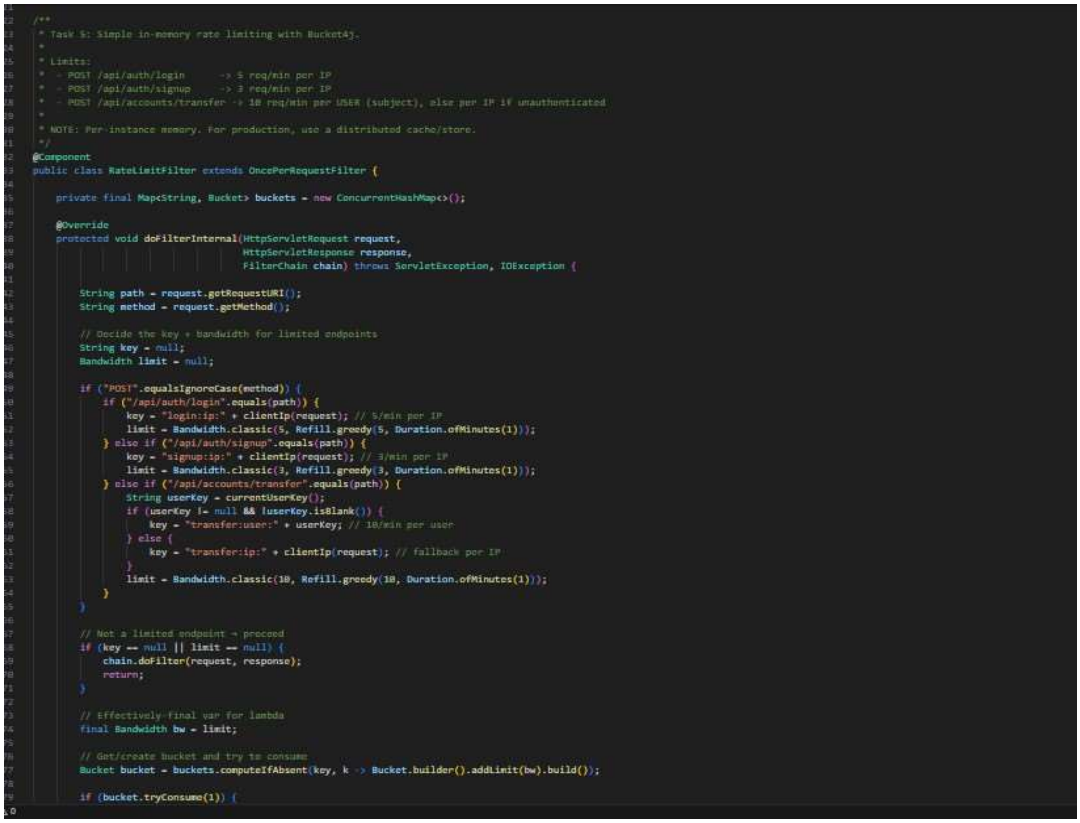
Key Code Changes

- security/RateLimitFilter.java — Bucket4j implementation
- config/SecurityConfig.java — register filter early

Validation / Test Steps

8. Blast login requests → after threshold, 429

Evidence (Screenshots)



```java
/**
 * Task 5: Simple in-memory rate limiting with Bucket4j.
 *
 * Limits:
 *   - POST /api/auth/login      -> 5 req/min per IP
 *   - POST /api/auth/signup     -> 3 req/min per IP
 *   - POST /api/accounts/transfer -> 10 req/min per USER (subject), else per IP if unauthenticated
 *
 * NOTE: Per-instance memory. For production, use a distributed cache/store.
 */
@Component
public class RateLimitFilter extends OncePerRequestFilter {

    private final Map<String, Bucket> buckets = new ConcurrentHashMap<>();

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain chain) throws ServletException, IOException {

        String path = request.getRequestURI();
        String method = request.getMethod();

        // Decide the key + bandwidth for limited endpoints
        String key = null;
        Bandwidth limit = null;

        if ("POST".equalsIgnoreCase(method)) {
            if ("/api/auth/login".equals(path)) {
                key = "login:ip:" + clientIp(request); // 5/min per IP
                limit = Bandwidth.classic(5, Refill.greedy(5, Duration.ofMinutes(1)));
            } else if ("/api/auth/signup".equals(path)) {
                key = "signup:ip:" + clientIp(request); // 3/min per IP
                limit = Bandwidth.classic(3, Refill.greedy(3, Duration.ofMinutes(1)));
            } else if ("/api/accounts/transfer".equals(path)) {
                String userKey = currentUserKey();
                if (userKey != null && !userKey.isBlank()) {
                    key = "transfer:user:" + userKey; // 10/min per user
                } else {
                    key = "transfer:ip:" + clientIp(request); // fallback per IP
                }
                limit = Bandwidth.classic(10, Refill.greedy(10, Duration.ofMinutes(1)));
            }
        }

        // Not a limited endpoint → proceed
        if (key == null || limit == null) {
            chain.doFilter(request, response);
            return;
        }

        // Effectively final var for lambda
        final Bandwidth bw = limit;

        // Get/create bucket and try to consume
        Bucket bucket = buckets.computeIfAbsent(key, k -> Bucket.builder().addLimit(bw).build());

        if (bucket.tryConsume(1)) {
```

code 1.PNG

```java
        }
    }

    // Not a limited endpoint → proceed
    if (key == null || limit == null) {
        chain.doFilter(request, response);
        return;
    }

    // Effectively-final var for lambda
    final Bandwidth bw = limit;

    // Get/create bucket and try to consume
    Bucket bucket = buckets.computeIfAbsent(key, k -> Bucket.builder().addLimit(bw).build());

    if (bucket.tryConsume(1)) {
        chain.doFilter(request, response);
        return;
    }

    // Too many requests → HTTP 429 + Retry-After
    long nanosToWait = bucket.getAvailableTokens() == 0
            ? Objects.requireNonNull(bucket.estimateAbilityToConsume(1).getNanosToWaitForRefill())
            : 0L;

    long retryAfterSeconds = Duration.ofNanos(nanosToWait).toSeconds();
    if (retryAfterSeconds < 1) retryAfterSeconds = 1;

    response.setStatus(429);
    response.setHeader("Retry-After", String.valueOf(retryAfterSeconds));
    response.setContentType(MediaType.APPLICATION_JSON_VALUE);
    response.getWriter().write("{\"error\":\"rate_limited\",\"retryAfterSeconds\":" + retryAfterSeconds + "}");
}

/** Resolve client IP using X-Forwarded-For (if present) or remote address. */
private String clientIp(HttpServletRequest req) {
    String xff = req.getHeader("X-Forwarded-For");
    if (xff != null && !xff.isBlank()) {
        int comma = xff.indexOf(',');
        return (comma > 0) ? xff.substring(0, comma).trim() : xff.trim();
    }
    return req.getRemoteAddr();
}

/** Current authenticated username (JWT subject); null if unauthenticated. */
private String currentUserKey() {
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    if (auth == null) return null;
    String name = auth.getName();
    return (name == null || name.isBlank()) ? null : name;
}
}
```

code 2.PNG

```
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $API = "http://localhost:8080"
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $JsonHeader = @{ "Content-Type" = "application/json" }
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # --- LOGIN to get tokens ---
>> $alice = Invoke-RestMethod -Method POST -Uri "$API/api/auth/login" -Headers $JsonHeader `
>>   -Body (@{ username="alice"; password="alice123" } | ConvertTo-Json)
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $bob   = Invoke-RestMethod -Method POST -Uri "$API/api/auth/login" -Headers $Json
>>   -Body (@{ username="bob";   password="bob123" } | ConvertTo-Json)
>>
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $hAlice = @{ Authorization = "Bearer " + $alice.token }
>>
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $hBob   = @{ Authorization = "Bearer " + $bob.token }
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # ===== Rate limit /api/auth/login (5 per minute per IP) =====
>> # Make 6 login attempts quickly from the same host to trigger 429.
>> 1..6 | % {
>>   try {
>>     Invoke-WebRequest -Method POST -Uri "$API/api/auth/login" -Headers $JsonHeader `
>>       -Body (@{ username="alice"; password="alice123" } | ConvertTo-Json) -ErrorAction Stop |
>>       Select-Object StatusCode, Content
>>     $_.Exception.Response.StatusCode
>>   }

StatusCode Content
---------- -------
       200 {"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGljZSIsImlzcyI6ImFwaWxhYiIsImF1ZCI6ImFwaWxhYi1jbGllbnRzIiwicm9sZSI6IlJPTEVfVVNFUiIsImlhdCI6MTc2MDg5NzA4NSwiZX...
       200 {"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGljZSIsImlzcyI6ImFwaWxhYiIsImF1ZCI6ImFwaWxhYi1jbGllbnRzIiwicm9sZSI6IlJPTEVfVVNFUiIsImlhdCI6MTc2MDg5NzA4NSwiZX...
       200 {"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGljZSIsImlzcyI6ImFwaWxhYiIsImF1ZCI6ImFwaWxhYi1jbGllbnRzIiwicm9sZSI6IlJPTEVfVVNFUiIsImlhdCI6MTc2MDg5NzA4NSwiZX...
       200 {"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGljZSIsImlzcyI6ImFwaWxhYiIsImF1ZCI6ImFwaWxhYi1jbGllbnRzIiwicm9sZSI6IlJPTEVfVVNFUiIsImlhdCI6MTc2MDg5NzA4NSwiZX...
       200 {"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGljZSIsImlzcyI6ImFwaWxhYiIsImF1ZCI6ImFwaWxhYi1jbGllbnRzIiwicm9sZSI6IlJPTEVfVVNFUiIsImlhdCI6MTc2MDg5NzA4NSwiZX...
429
```

login flood.PNG

```
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # =====Rate limit /api/auth/signup (3 per minute per IP) =====
>> 1..4 | % {
>>   $u = "u$($_)lim"
>>   try {
>>     Invoke-WebRequest -Method POST -Uri "$API/api/auth/signup" -Headers $JsonHeader `
>>       -Body (@{ username=$u; email="$u@example.com"; password="StrongP@ssw0rd!" } | ConvertTo-Json) -ErrorAction Stop |
>>       Select-Object StatusCode, Content
>>   } catch {
>>     $_.Exception.Response.StatusCode
>>     $r = New-Object IO.StreamReader($_.Exception.Response.GetResponseStream()); $r.ReadToEnd()
>>   }
>> }

StatusCode Content
---------- -------
       200 {"message":"User registered successfully"}
       200 {"message":"User registered successfully"}
       200 {"message":"User registered successfully"}
429
```

sign up flood.PNG

```
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # ===== Test C: Rate limit /api/accounts/transfer (10/min per USER) =====
>> # Discover dynamic account IDs:
>> $aliceAcctId = (Invoke-RestMethod -Uri "$API/api/accounts/mine" -Headers $hAlice)[0].id
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $bobAcctId   = (Invoke-RestMethod -Uri "$API/api/accounts/mine" -Headers $hBob)[0
].id
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $hAliceJson  = @{ Authorization = $hAlice.Authorization; "Content-Type" = "applic
ation/json" }
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # Send 11 transfers as Alice to exceed limit
>> 1..11 | % {
>>   $body = @{ fromAccountId=$aliceAcctId; toAccountId=$bobAcctId; amount=1 } | ConvertTo-Json
>>   try {
>>     Invoke-WebRequest -Method POST -Uri "$API/api/accounts/transfer" -Headers $hAliceJson -Body $body -ErrorAction Stop |
>>       Select-Object StatusCode, Content
>>   } catch {
>>     $_.Exception.Response.StatusCode
>>     $r = New-Object IO.StreamReader($_.Exception.Response.GetResponseStream()); $r.ReadToEnd()
>>   }
>> }

StatusCode Content
---------- -------
       200 {"toAccount":{"id":2,"balance":5001.0},"fromAccount":{"id":1,"balance":999.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5002.0},"fromAccount":{"id":1,"balance":998.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5003.0},"fromAccount":{"id":1,"balance":997.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5004.0},"fromAccount":{"id":1,"balance":996.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5005.0},"fromAccount":{"id":1,"balance":995.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5006.0},"fromAccount":{"id":1,"balance":994.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5007.0},"fromAccount":{"id":1,"balance":993.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5008.0},"fromAccount":{"id":1,"balance":992.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5009.0},"fromAccount":{"id":1,"balance":991.0},"status":"ok"}
       200 {"toAccount":{"id":2,"balance":5010.0},"fromAccount":{"id":1,"balance":990.0},"status":"ok"}
429
```

transfer flood.PNG

6) Mass Assignment Prevention:

Vulnerability & Context

Clients could set privileged fields (role/isAdmin).

Risk/Impact

Privilege escalation via crafted JSON.

Fix Approach & Rationale

Use explicit request DTOs; ignore/override privileged fields server-side.

Key Code Changes

 web/dto/SignupRequest.java, UpdateProfileRequest.java — safe fields only web/AuthController.java — enforce default role server-side

Validation / Test Steps

9. Signup with {"role":"ADMIN"} still persists non-admin user

Evidence (Screenshots)

```
    /**
     * Task 6: Safe self-update (no role/isAdmin allowed).
     * Accepts only email/password via UpdateProfileRequest; validation via annotations.
     */
    @PutMapping("/me")
    public ResponseEntity<?> updateMe(@Valid @RequestBody UpdateProfileRequest req, Authentication auth) {
        AppUser me = current.current(auth).orElse(null);
        if (me == null) return ResponseEntity.status(401).body(Map.of("error","unauthenticated"));

        // Whitelisted updates only:
        if (req.getEmail() != null) {
            me.setEmail(req.getEmail().trim());
        }
        if (req.getPassword() != null && !req.getPassword().isBlank()) {
            me.setPassword(encoder.encode(req.getPassword())); // server-side encode (no plaintext)
        }

        users.save(me);
        return ResponseEntity.ok(Map.of("status", "updated", "user", toDto(me)));
    }
}
```

code.PNG

```
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $API = "http://localhost:8080"
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $JsonHeader = @{ "Content-Type" = "application/json" }
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # Login as alice (seed)
>> $alice = Invoke-RestMethod -Method POST -Uri "$API/api/auth/login" -Headers $JsonHeader `
>>   -Body (@{ username="alice"; password="alice123" } | ConvertTo-Json)
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> $hAlice = @{ Authorization = "Bearer " + $alice.token }
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # 1) Attempt mass assignment at signup (should 400 due to unknown fields)
>> $badSignup = @{
>>   username = "eviladmin";
>>   email    = "evil@example.com";
>>   password = "StrongP@ssw0rd!";
>>   role     = "ROLE_ADMIN";   # not allowed
>>   isAdmin  = $true           # not allowed
>> } | ConvertTo-Json
>> try {
>>   Invoke-WebRequest -Method POST -Uri "$API/api/auth/signup" -Headers $JsonHeader -Body $badSignup -ErrorAction Stop |
>> } catch {
>>   $_.Exception.Response.StatusCode
>>   $r = New-Object IO.StreamReader($_.Exception.Response.GetResponseStream()); $r.ReadToEnd()
BadRequest
{"error":"bad_request","message":"Unrecognized field: isAdmin"}
```

mass 1.PNG

```
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # 1) Attempt mass assignment at signup (should 400 due to unknown fields)
>> $badSignup = @{
>>   username = "eviladmin";
>>   email    = "evil@example.com";
>>   password = "StrongP@ssw0rd!"; …
>> $okSignup = @{ username="uSafe1"; email="usafe1@example.com"; password="Str0ngP@ss!" } | ConvertTo-Json
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> Invoke-WebRequest -Method POST -Uri "$API/api/auth/signup" -Headers $JsonHeader -Body $okSignup |
>>   Select-Object StatusCode, Content

StatusCode Content
---------- -------
       200 {"message":"User registered successfully"}


PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # 3) Try to escalate privileges via profile update (should 400: unknown fields)
>> $badUpdate = @{
>>   email   = "alice+new@example.com";
>>   role    = "ROLE_ADMIN";     # not allowed
>>   isAdmin = $true             # not allowed
>> } | ConvertTo-Json
>> try {
>>   Invoke-WebRequest -Method PUT -Uri "$API/api/users/me" -Headers @{ Authorization=$hAlice.Authorization; "Content-Type"="application/json"} `
>>     -Body $badUpdate -ErrorAction Stop | Select-Object StatusCode, Content
>> } catch {
>>   $_.Exception.Response.StatusCode
>>   $r = New-Object IO.StreamReader($_.Exception.Response.GetResponseStream()); $r.ReadToEnd()
>> }
BadRequest
{"error":"bad_request","message":"Unrecognized field: isAdmin"}
```

mass 2.PNG

```
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> # 4) Legit profile update (email/password only)
>> $okUpdate = @{ email = "alice+ok@example.com"; password="NewP@ssw0rd!" } | ConvertTo-Json
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> Invoke-WebRequest -Method PUT -Uri "$API/api/users/me" -Headers @{ Authorization=$hAlice.Authorization; "Content
-Type"="application/json"} `
>>   -Body $okUpdate | Select-Object StatusCode, Content

StatusCode Content
---------- -------
       200 {"user":{"id":1,"username":"alice","email":"alice+ok@example.com"},"status":"updated"}
```

mass 3.PNG

7) JWT Hardening:

Vulnerability & Context

Weak secret/missing iss/aud/short TTL make tokens forgeable/replayable.

Risk/Impact

Forged/long-lived tokens allow persistent unauthorized access.

Fix Approach & Rationale

Strong secret from env; validate iss/aud/exp; short TTL; strict signature verification.

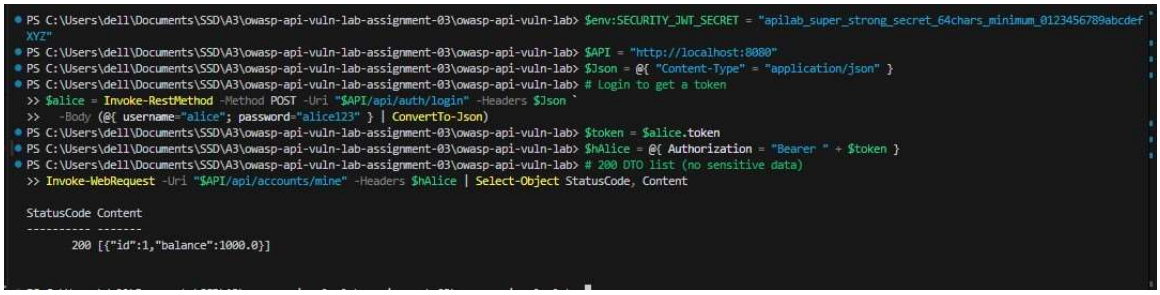
Key Code Changes

- service/JwtService.java — iss/aud/exp & HMAC key
- application.yaml — security.jwt.* config

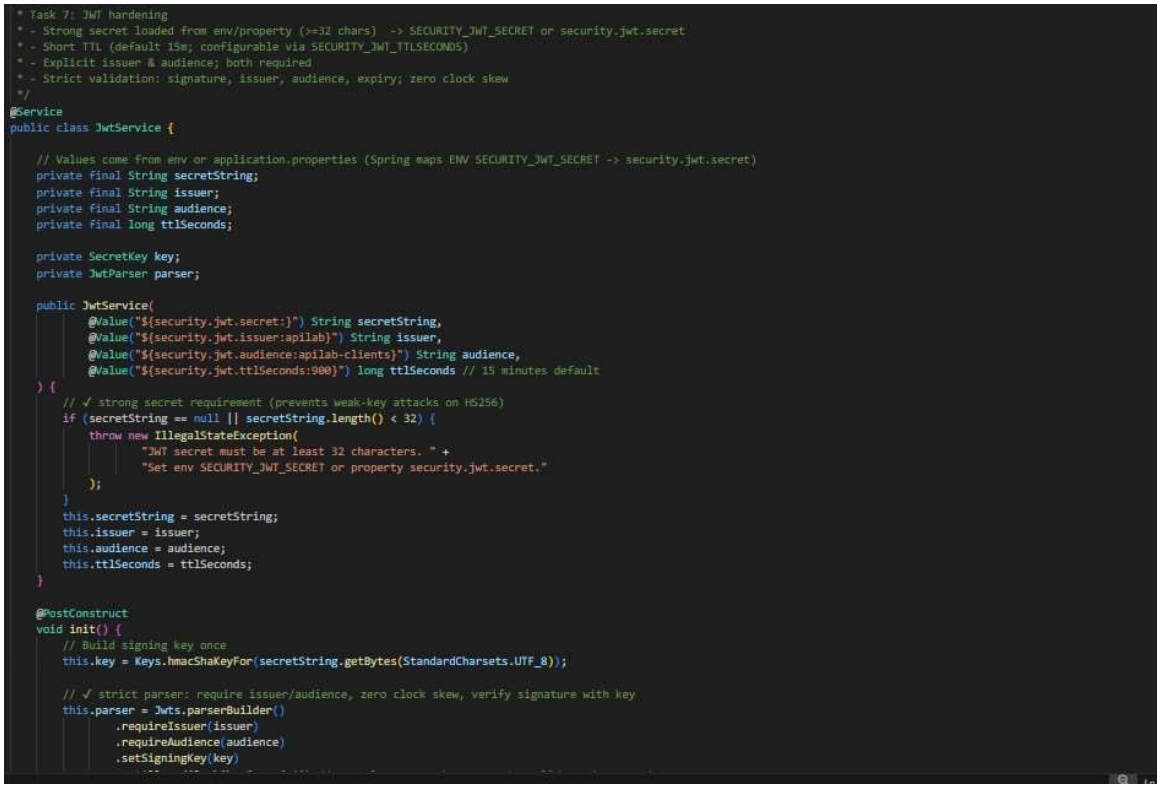
Validation / Test Steps

10. Expired token → 401; wrong audience → 401; valid token → 200

Evidence (Screenshots)



200 DTO list (no sensitive data).PNG

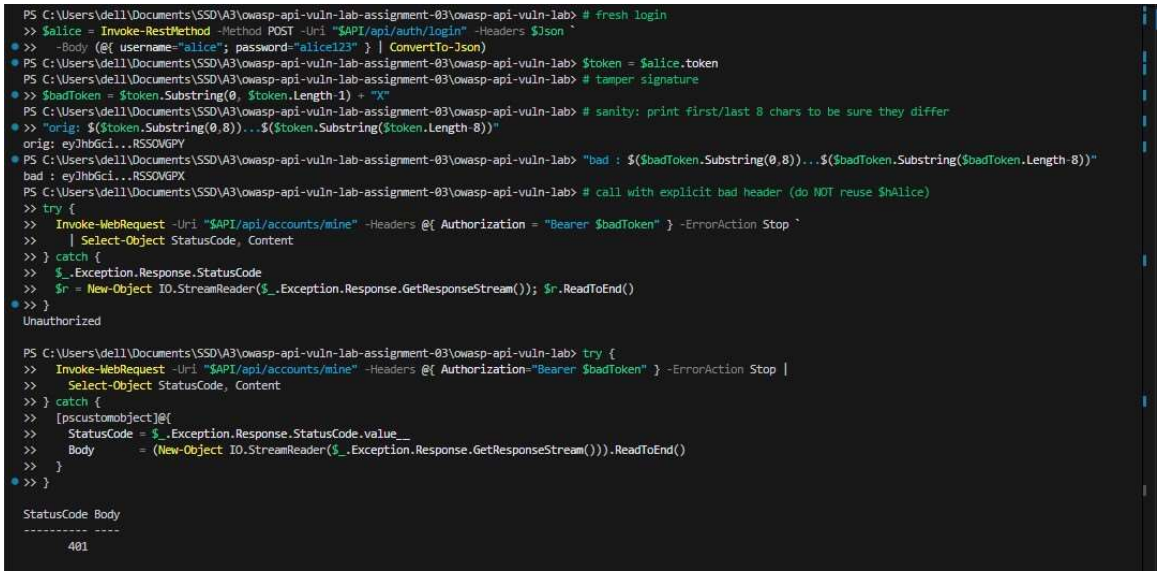


```
 * Task 7: JWT hardening
 * - Strong secret loaded from env/property (>=32 chars)  -> SECURITY_JWT_SECRET or security.jwt.secret
 * - Short TTL (default 15m; configurable via SECURITY_JWT_TTLSECONDS)
 * - Explicit issuer & audience; both required
 * - Strict validation: signature, issuer, audience, expiry; zero clock skew
 */
@Service
public class JwtService {

    // Values come from env or application.properties (Spring maps ENV SECURITY_JWT_SECRET -> security.jwt.secret)
    private final String secretString;
    private final String issuer;
    private final String audience;
    private final long ttlSeconds;

    private SecretKey key;
    private JwtParser parser;

    public JwtService(
            @Value("${security.jwt.secret:}") String secretString,
            @Value("${security.jwt.issuer:apilab}") String issuer,
            @Value("${security.jwt.audience:apilab-clients}") String audience,
            @Value("${security.jwt.ttlSeconds:900}") long ttlSeconds // 15 minutes default
    ) {
        // ✓ strong secret requirement (prevents weak-key attacks on HS256)
        if (secretString == null || secretString.length() < 32) {
            throw new IllegalStateException(
                    "JWT secret must be at least 32 characters. " +
                    "Set env SECURITY_JWT_SECRET or property security.jwt.secret."
            );
        }
        this.secretString = secretString;
        this.issuer = issuer;
        this.audience = audience;
        this.ttlSeconds = ttlSeconds;
    }

    @PostConstruct
    void init() {
        // Build signing key once
        this.key = Keys.hmacShaKeyFor(secretString.getBytes(StandardCharsets.UTF_8));

        // ✓ strict parser: require issuer/audience, zero clock skew, verify signature with key
        this.parser = Jwts.parserBuilder()
                .requireIssuer(issuer)
                .requireAudience(audience)
                .setSigningKey(key)
```

code 1.PNG

```
    }

    @PostConstruct
    void init() {
        // Build signing key once
        this.key = Keys.hmacShaKeyFor(secretString.getBytes(StandardCharsets.UTF_8));

        // ✓ strict parser: require issuer/audience, zero clock skew, verify signature with key
        this.parser = Jwts.parserBuilder()
                .requireIssuer(issuer)
                .requireAudience(audience)
                .setSigningKey(key)
                .setAllowedClockSkewSeconds(0) // no tolerance; tokens must be valid to the second
                .build();
    }

    /** Issue compact JWT with subject=username and "role" claim. */
    public String issueToken(String username, String role) {
        Date now = new Date();
        Date exp = new Date(now.getTime() + (ttlSeconds * 1000));

        return Jwts.builder()
                .setSubject(username)
                .setIssuer(issuer)     // ✓ include issuer
                .setAudience(audience)// ✓ include audience
                .claim("role", role)
                .setIssuedAt(now)
                .setExpiration(exp)    // ✓ short TTL enforced
                .signWith(key, SignatureAlgorithm.HS256) // ✓ HS256 with strong secret
                .compact();
    }

    /** Validate token and return subject+role; throws IllegalArgumentException on any failure. */
    public SubjectAndRole validateAndGetSubjectAndRole(String token) {
        try {
            Jws<Claims> jws = parser.parseClaimsJws(token); // ✓ verifies signature, issuer, audience, expiry
            Claims claims = jws.getBody();

            String username = claims.getSubject();
            Object roleObj = claims.get("role");
            String role = (roleObj == null) ? null : String.valueOf(roleObj);

            if (username == null || username.isBlank() || role == null || role.isBlank()) {
                throw new IllegalArgumentException("invalid_or_expired_token");
            }
            return new SubjectAndRole(username, role);
        } catch (JwtException | IllegalArgumentException e) {
            // Any issue (bad sig, wrong iss/aud, expired, malformed) -> invalid
            throw new IllegalArgumentException("invalid_or_expired_token");
        }
    }

    /** Small DTO used by security filter to set Authentication authorities. */
    public record SubjectAndRole(String username, String role) {}
}
```

code 2.PNG



tampered signature (401).PNG

8) Error Handling & Logging:

Vulnerability & Context

Stack traces and framework details leaked to clients.

Risk/Impact

Information disclosure aids targeted exploitation.

Fix Approach & Rationale

Map to minimal client messages; correlate logs with request ID; keep details server-side.

Key Code Changes

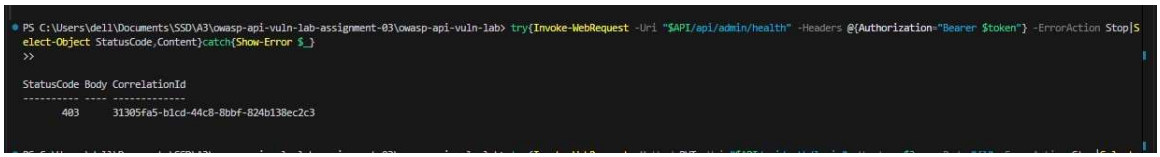
 web/advices/GlobalExceptionHandler.java — safe mapping
filters/CorrelationIdFilter.java — X-Correlation-ID

Validation / Test Steps

11. Trigger validation error → concise client message; detailed server log with correlation ID

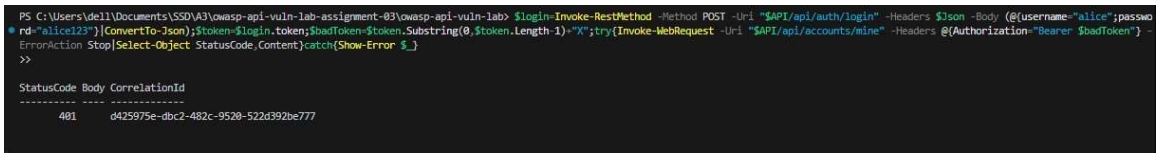
Evidence (Screenshots)

Bad JSON (400).PNG

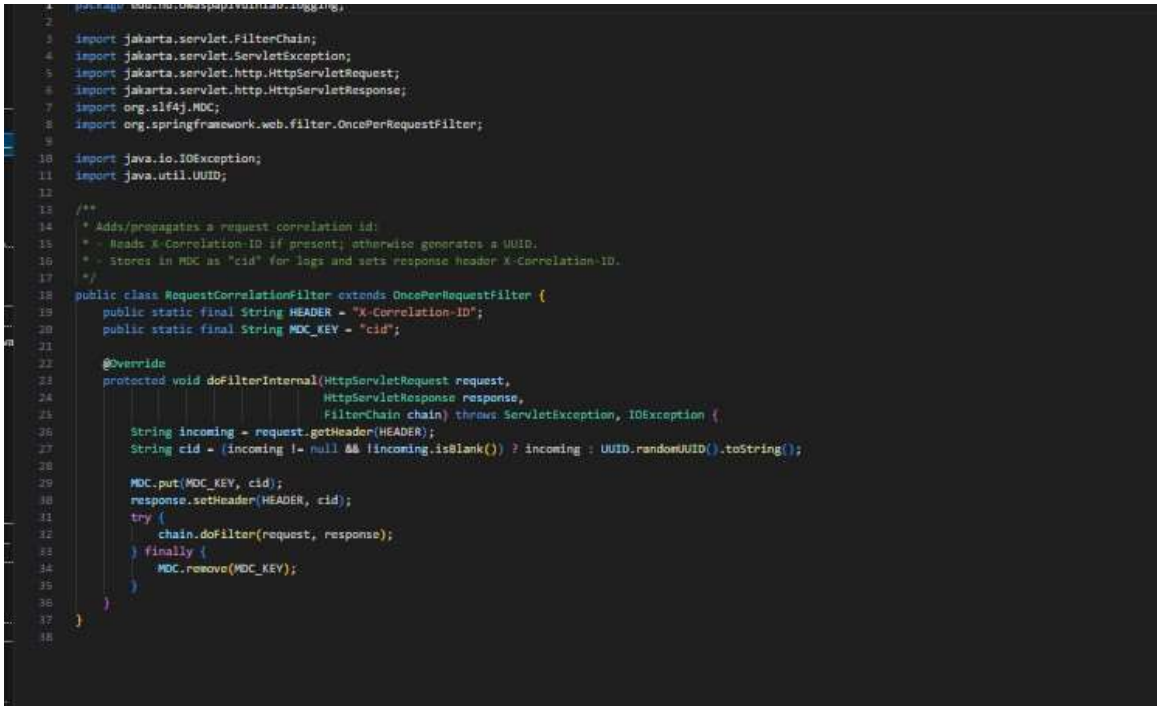


Forbidden (user hitting admin) 403.PNG

Method and not found error.PNG



Unauthorized (tampered token) 401.PNG



```
package edu.nu.owaspapivulnlab.logging;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.slf4j.MDC;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;
import java.util.UUID;

/**
 * Adds/propagates a request correlation id:
 * - Reads X-Correlation-ID if present; otherwise generates a UUID.
 * - Stores in MDC as "cid" for logs and sets response header X-Correlation-ID.
 */
public class RequestCorrelationFilter extends OncePerRequestFilter {
    public static final String HEADER = "X-Correlation-ID";
    public static final String MDC_KEY = "cid";

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain chain) throws ServletException, IOException {
        String incoming = request.getHeader(HEADER);
        String cid = (incoming != null && !incoming.isBlank()) ? incoming : UUID.randomUUID().toString();

        MDC.put(MDC_KEY, cid);
        response.setHeader(HEADER, cid);
        try {
            chain.doFilter(request, response);
        } finally {
            MDC.remove(MDC_KEY);
        }
    }
}
```

code.PNG

9) Input Validation:

Vulnerability & Context

Negative or malformed inputs allowed state corruption/DoS.

Risk/Impact

Unbounded or invalid data breaks business logic and stability.

Fix Approach & Rationale

Jakarta annotations & explicit numeric/format checks; reject invalid inputs early.

Key Code Changes

- web/dto/TransferRequest.java — @NotNull, @Positive, @Digits
- validators/ValidationRules.java — custom checks

Validation / Test Steps

12. POST transfer amount=-10 → 400; valid → 200

Evidence (Screenshots)

Invalid signup rejected (short password).PNG

Invalid transfer rejected (negative amount).PNG

```java
/**
 * Task 9: Input Validation error responses.
 * Maps bean validation errors to a consistent 400 JSON payload,
 */
@ControllerAdvice
public class ValidationErrorHandler {

    private String cid() {
        String cid = MDC.get(RequestCorrelationFilter.MDC_KEY);
        return cid == null ? "n/a" : cid;
    }

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<Map<String, Object>> onInvalidBody(MethodArgumentNotValidException ex) {
        List<Map<String, String>> details = ex.getBindingResult()
                .getFieldErrors()
                .stream()
                .map(this::toDetail)
                .collect(Collectors.toList());

        Map<String, Object> body = new HashMap<>();
        body.put("error", "validation_error");
        body.put("correlationId", cid());
        body.put("details", details);

        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(body);
    }

    @ExceptionHandler(ConstraintViolationException.class)
    public ResponseEntity<Map<String, Object>> onConstraintViolation(ConstraintViolationException ex) {
        List<Map<String, String>> details = ex.getConstraintViolations()
                .stream().map(this::toDetail)
                .collect(Collectors.toList());

        Map<String, Object> body = new HashMap<>();
        body.put("error", "validation_error");
        body.put("correlationId", cid());
        body.put("details", details);
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(body);
    }

    private Map<String, String> toDetail(FieldError fe) {
        Map<String, String> d = new HashMap<>();
        d.put("field", fe.getField());
        d.put("message", fe.getDefaultMessage());
        return d;
    }
```

code 1.PNG

```java
package edu.nu.owaspapivulnlab.web.validators;

/**
 * Task 9: Centralized input validation rules and ranges.

 */
public final class ValidationRules {
    private ValidationRules() {}

    // Numeric bounds
    public static final long MIN_ID = 1L;
    public static final long MAX_ID = Long.MAX_VALUE / 4; // defensive upper-bound

    // Money (demo): reject negative or excessively large transfers
    public static final String MIN_AMOUNT_STR = "0.01";
    public static final String MAX_AMOUNT_STR = "1000000"; // 1M cap for lab

    // Username / password
    public static final int USERNAME_MIN = 3;
    public static final int USERNAME_MAX = 50;
    public static final int PASSWORD_MIN = 8;
    public static final int PASSWORD_MAX = 128;
}
```

code 2.PNG

10) Testing & Verification:

Vulnerability & Context

Lack of tests risks regression and security drift.

Risk/Impact

Undetected re-introduction of vulnerabilities.

Fix Approach & Rationale

Integration tests for auth, authorization, ownership, rate-limits, and DTO exposure.

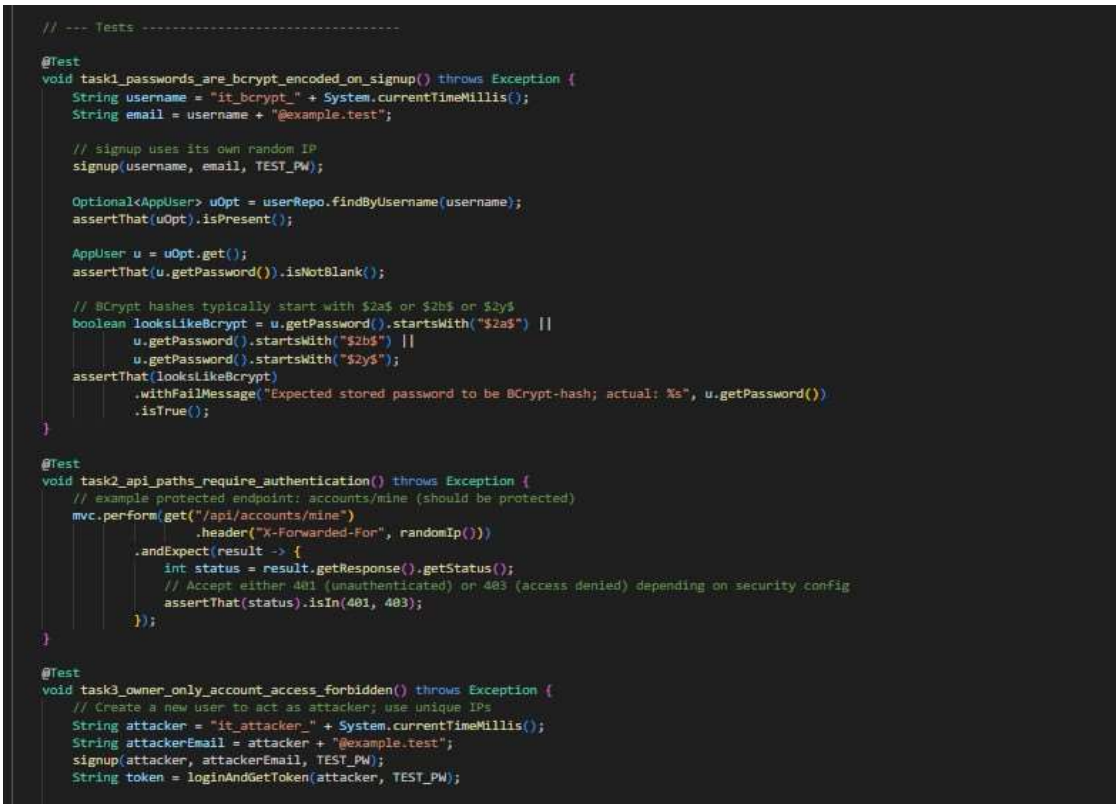
Key Code Changes

 test/java/.../it — verify 401/403, DTO fields, rate-limit 429, transfer invariants

Validation / Test Steps

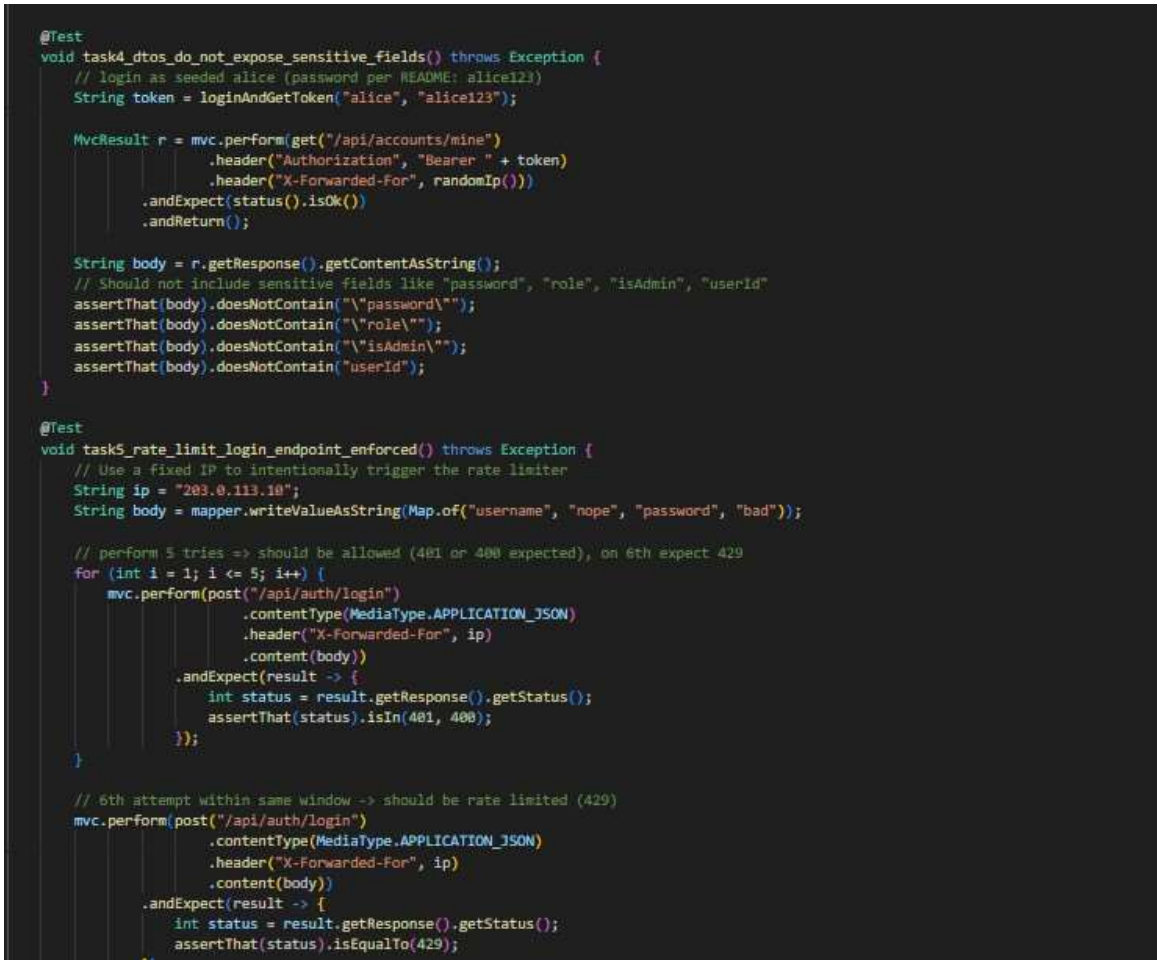
13. Run: mvn clean test → all pass

Evidence (Screenshots)



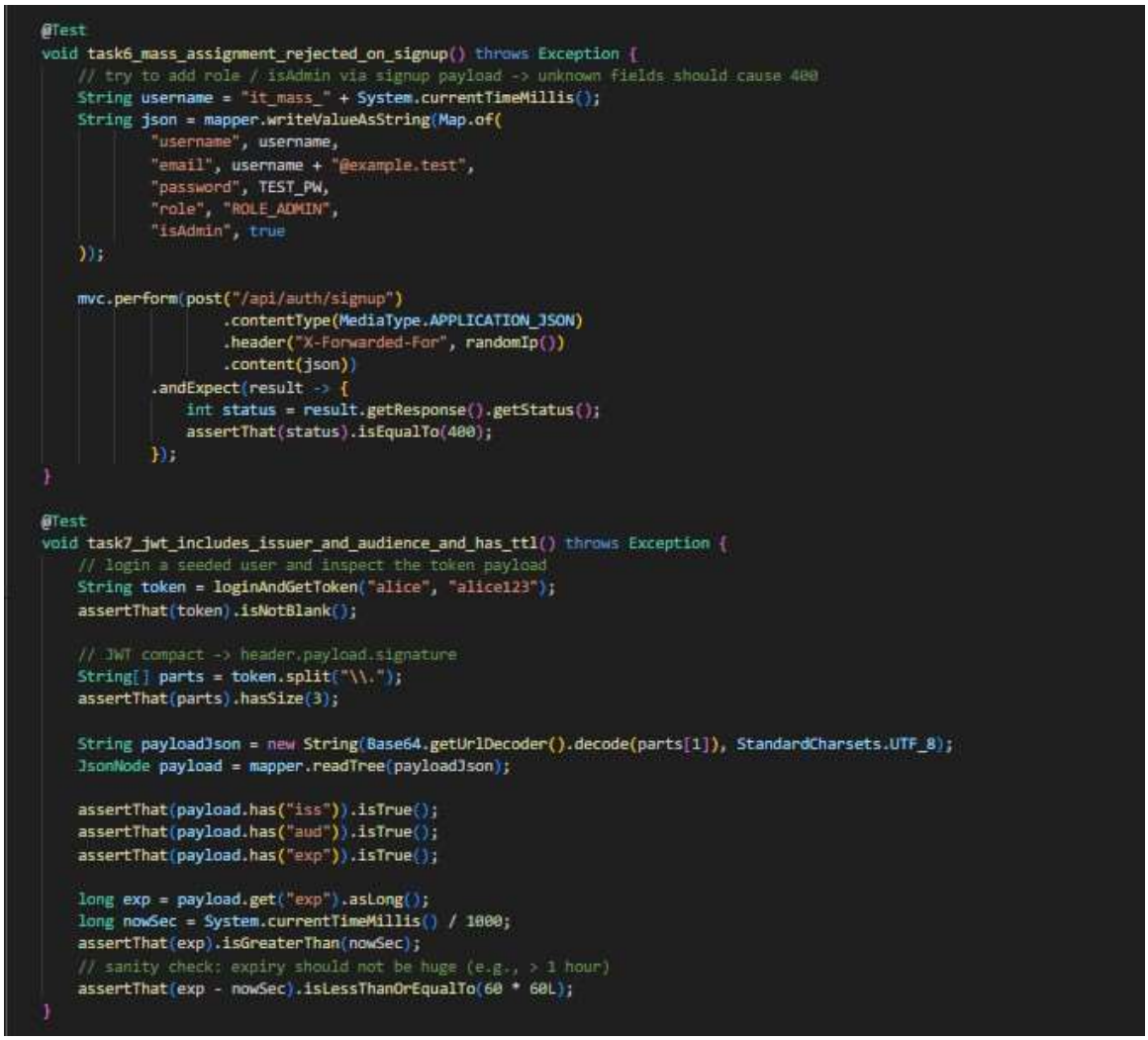
```
// --- Tests --------------------------------

@Test
void task1_passwords_are_bcrypt_encoded_on_signup() throws Exception {
    String username = "it_bcrypt_" + System.currentTimeMillis();
    String email = username + "@example.test";

    // signup uses its own random IP
    signup(username, email, TEST_PW);

    Optional<AppUser> uOpt = userRepo.findByUsername(username);
    assertThat(uOpt).isPresent();

    AppUser u = uOpt.get();
    assertThat(u.getPassword()).isNotBlank();

    // BCrypt hashes typically start with $2a$ or $2b$ or $2y$
    boolean looksLikeBcrypt = u.getPassword().startsWith("$2a$") ||
            u.getPassword().startsWith("$2b$") ||
            u.getPassword().startsWith("$2y$");
    assertThat(looksLikeBcrypt)
            .withFailMessage("Expected stored password to be BCrypt-hash; actual: %s", u.getPassword())
            .isTrue();
}

@Test
void task2_api_paths_require_authentication() throws Exception {
    // example protected endpoint: accounts/mine (should be protected)
    mvc.perform(get("/api/accounts/mine")
                    .header("X-Forwarded-For", randomIp()))
            .andExpect(result -> {
                int status = result.getResponse().getStatus();
                // Accept either 401 (unauthenticated) or 403 (access denied) depending on security config
                assertThat(status).isIn(401, 403);
            });
}

@Test
void task3_owner_only_account_access_forbidden() throws Exception {
    // Create a new user to act as attacker; use unique IPs
    String attacker = "it_attacker_" + System.currentTimeMillis();
    String attackerEmail = attacker + "@example.test";
    signup(attacker, attackerEmail, TEST_PW);
    String token = loginAndGetToken(attacker, TEST_PW);
```

code 1.PNG



```java
@Test
void task4_dtos_do_not_expose_sensitive_fields() throws Exception {
    // login as seeded alice (password per README: alice123)
    String token = loginAndGetToken("alice", "alice123");

    MvcResult r = mvc.perform(get("/api/accounts/mine")
                    .header("Authorization", "Bearer " + token)
                    .header("X-Forwarded-For", randomIp()))
            .andExpect(status().isOk())
            .andReturn();

    String body = r.getResponse().getContentAsString();
    // Should not include sensitive fields like "password", "role", "isAdmin", "userId"
    assertThat(body).doesNotContain("\"password\"");
    assertThat(body).doesNotContain("\"role\"");
    assertThat(body).doesNotContain("\"isAdmin\"");
    assertThat(body).doesNotContain("userId");
}

@Test
void task5_rate_limit_login_endpoint_enforced() throws Exception {
    // Use a fixed IP to intentionally trigger the rate limiter
    String ip = "203.0.113.10";
    String body = mapper.writeValueAsString(Map.of("username", "nope", "password", "bad"));

    // perform 5 tries => should be allowed (401 or 400 expected), on 6th expect 429
    for (int i = 1; i <= 5; i++) {
        mvc.perform(post("/api/auth/login")
                        .contentType(MediaType.APPLICATION_JSON)
                        .header("X-Forwarded-For", ip)
                        .content(body))
                .andExpect(result -> {
                    int status = result.getResponse().getStatus();
                    assertThat(status).isIn(401, 400);
                });
    }

    // 6th attempt within same window -> should be rate limited (429)
    mvc.perform(post("/api/auth/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .header("X-Forwarded-For", ip)
                    .content(body))
            .andExpect(result -> {
                int status = result.getResponse().getStatus();
                assertThat(status).isEqualTo(429);
```

code 2.PNG



```java
@Test
void task6_mass_assignment_rejected_on_signup() throws Exception {
    // try to add role / isAdmin via signup payload -> unknown fields should cause 400
    String username = "it_mass_" + System.currentTimeMillis();
    String json = mapper.writeValueAsString(Map.of(
            "username", username,
            "email", username + "@example.test",
            "password", TEST_PW,
            "role", "ROLE_ADMIN",
            "isAdmin", true
    ));

    mvc.perform(post("/api/auth/signup")
                    .contentType(MediaType.APPLICATION_JSON)
                    .header("X-Forwarded-For", randomIp())
                    .content(json))
            .andExpect(result -> {
                int status = result.getResponse().getStatus();
                assertThat(status).isEqualTo(400);
            });
}

@Test
void task7_jwt_includes_issuer_and_audience_and_has_ttl() throws Exception {
    // login a seeded user and inspect the token payload
    String token = loginAndGetToken("alice", "alice123");
    assertThat(token).isNotBlank();

    // JWT compact -> header.payload.signature
    String[] parts = token.split("\\.");
    assertThat(parts).hasSize(3);

    String payloadJson = new String(Base64.getUrlDecoder().decode(parts[1]), StandardCharsets.UTF_8);
    JsonNode payload = mapper.readTree(payloadJson);

    assertThat(payload.has("iss")).isTrue();
    assertThat(payload.has("aud")).isTrue();
    assertThat(payload.has("exp")).isTrue();

    long exp = payload.get("exp").asLong();
    long nowSec = System.currentTimeMillis() / 1000;
    assertThat(exp).isGreaterThan(nowSec);
    // sanity check: expiry should not be huge (e.g., > 1 hour)
    assertThat(exp - nowSec).isLessThanOrEqualTo(60 * 60L);
}
```

code 3.PNG

```java
@Test
void task8_error_responses_do_not_leak_stacktrace_or_internal_details() throws Exception {
    // Send malformed JSON which should trigger a 400 (or 429 if rate-limited)
    String badJson = "{ this-is-not: valid-json }";
    MvcResult r = mvc.perform(post("/api/auth/signup")
                    .contentType(MediaType.APPLICATION_JSON)
                    .header("X-Forwarded-For", randomIp())
                    .content(badJson))
            .andReturn();

    int status = r.getResponse().getStatus();
    // Prefer 400; if 429 appears it indicates rate-limiter still in play.
    assertThat(status).isIn(400, 429);

    String body = r.getResponse().getContentAsString();
    // If we got 400, ensure we didn't leak internals; if 429, it's rate-limited response and safe.
    if (status == 400) {
        assertThat(body.toLowerCase()).doesNotContain("stacktrace");
        assertThat(body).doesNotContain("Exception");
        assertThat(body).contains("error");
    } else {
        // rate-limited response should have structured JSON
        assertThat(body).contains("rate_limited");
    }
}

@Test
void task9_transfer_negative_or_excessive_amount_validated() throws Exception {
    // sign up user and attempt a negative transfer
    String user = "it_transfer_" + System.currentTimeMillis();
    signup(user, user + "@x.test", TEST_PW);
    String token = loginAndGetToken(user, TEST_PW);

    String neg = "{\"fromAccountId\":1,\"toAccountId\":2,\"amount\":-50}";
    mvc.perform(post("/api/accounts/transfer")
                    .contentType(MediaType.APPLICATION_JSON)
                    .header("Authorization", "Bearer " + token)
                    .header("X-Forwarded-For", randomIp())
                    .content(neg))
            .andExpect(result -> {
                int status = result.getResponse().getStatus();
                assertThat(status).isEqualTo(400);
            });

    String huge = String.format("{\"fromAccountId\":1,\"toAccountId\":2,\"amount\":%d}", 10_000_000);
    mvc.perform(post("/api/accounts/transfer")
                    .contentType(MediaType.APPLICATION_JSON)
```

code 4.PNG

```
PS C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab> mvn -DskipTests=false test
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running edu.nu.owaspapivulnlab.AdditionalSecurityExpectationsTests
11:57:07.011 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect default configuration classes for test class [edu.nu.owaspapivulnlab.A
dditionalSecurityExpectationsTests]: AdditionalSecurityExpectationsTests does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
11:57:07.671 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Found @SpringBootConfiguration edu.nu.owaspapivulnlab.OwaspApiVulnLabApplication for test c
lass edu.nu.owaspapivulnlab.AdditionalSecurityExpectationsTests

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.4)

2025-10-21T11:57:10.182+05:00  INFO 9892 --- [           main] .n.o.AdditionalSecurityExpectationsTests : Starting AdditionalSecurityExpectationsTests using Java 21.0.8 with PID 9892 (started
 by dell in C:\Users\dell\Documents\SSD\A3\owasp-api-vuln-lab-assignment-03\owasp-api-vuln-lab)
2025-10-21T11:57:10.198+05:00  INFO 9892 --- [           main] .n.o.AdditionalSecurityExpectationsTests : No active profile set, falling back to 1 default profile: "default"
2025-10-21T11:57:14.252+05:00  INFO 9892 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-10-21T11:57:14.648+05:00  INFO 9892 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 343 ms. Found 2 JPA repository interfaces
.
2025-10-21T11:57:17.360+05:00  INFO 9892 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-10-21T11:57:17.795+05:00  INFO 9892 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.5.3.Final
2025-10-21T11:57:18.013+05:00  INFO 9892 --- [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-10-21T11:57:20.041+05:00  INFO 9892 --- [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-10-21T11:57:20.281+05:00  INFO 9892 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-10-21T11:57:21.284+05:00  INFO 9892 --- [           main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:apilab user=SA
```

test 1.PNG

How to Build, Run, and Test:

- Build: mvn -q -DskipTests=false clean package
- Run: mvn spring-boot:run
- Public endpoints: /api/auth/login, /api/auth/signup
- Use Bearer token for authenticated endpoints (Accounts, Transfers)
- Flow: signup → get token → GET /api/accounts/mine → POST transfer → verify responses.

Conclusion:

The implemented security controls effectively strengthened the API by addressing key OWASP vulnerabilities. Each fix improved authentication, authorization, data protection, and overall resilience, ensuring a more secure and reliable application.

Repository Link:

https://github.com/emantariq634-bot/owasp-api-vuln-lab