

datavane / datavines Public

&lt;&gt; Code Issues 73 Pull requests 1 Discussions Actions Projects

# [Fix][Security] Fix JWT authentication bypass via hardcoded secret and self-comparison logic #579



jackieya wants to merge 3 commits into datavane:dev from

jackieya:fix/jwt-auth-bypass



Conversation 6



Commits 3



Checks 4



Files changed 4



jackieya commented 3 weeks ago

## What is the purpose of the change

Fix a critical JWT authentication bypass vulnerability caused by two flaws:

- Hardcoded JWT secret:** `TokenManager` uses `@Value("${jwt.token.secret:asdqwe}")` with a hardcoded default, and `application.yaml` has no `jwt.token.secret` entry — all default deployments share the same secret.
- Self-comparison in token validation:** `AuthenticationInterceptor` passes `tokenManager.getPassword(token)` (extracted from the token itself) as the expected password to `validateToken`, which then compares it with itself — always returns true.

Combined, an attacker can forge a valid JWT for any user (e.g. `admin`) without knowing the actual password.

## Brief change log

- `TokenManager.java`** : Remove hardcoded default secret `asdqwe`, add `@PostConstruct` startup validation to auto-generate a random secret if not configured, and change `validateToken` to use `BCrypt.checkpw` instead of `String.equals` for password comparison.
- `AuthenticationInterceptor.java`** : Change `tokenManager.getPassword(token)` to `user.getPassword()` so the token's password is validated against the database `BCrypt` hash, not against itself.
- `application.yaml`** : Add `jwt.token.secret` configuration entry so users can discover and configure a custom secret.

## Verify this change

- Forged tokens are now rejected (BCrypt check fails for fake passwords).
- Normal login and token refresh continue to work correctly.
- Tested locally with both forged and legitimate tokens.

  [\[Fix\]\[Security\] Fix JWT authentication bypass via hardcoded secret an...](#)  [@bf9681](#)

  **Copilot** (AI) review requested due to automatic review settings [3 weeks ago](#)

 **Copilot** [started reviewing](#) on behalf of **jackieya** [3 weeks ago](#)

[View session](#)

  **jackieya** mentioned this pull request [3 weeks ago](#)

## **[Bug] [Security] JWT Authentication Bypass via Hardcoded Secret and Self-Comparison Logic #580**

 Open

 2 tasks

 **Copilot** (AI) reviewed [3 weeks ago](#)

[View reviewed changes](#)

 **Copilot** (AI) left a comment

## Pull request overview

This PR addresses a critical JWT authentication bypass by removing the hardcoded fallback secret, fixing token validation to compare against the stored user password hash (instead of self-comparing token-derived values), and surfacing JWT configuration in `application.yaml`.


### Changes:

- Updated `AuthenticationInterceptor` to validate the token against the database-stored password hash (`user.getPassword()`), not the token's own embedded password.
- Updated `TokenManager` to remove the hardcoded default secret, add startup secret handling, and switch token password comparison to `BCrypt.checkpw`.
- Added a `jwt.token.secret` entry to `application.yaml` for discoverability/configuration.

## Reviewed changes

Copilot reviewed 3 out of 3 changed files in this pull request and generated 5 comments.

File	Description
datavines-server/src/main/resources/application.yaml	Adds JWT configuration keys (secret/timeout/algorithm).
datavines-server/src/main/java/io/datavines/server/api/inteceptor/AuthenticationInterceptor.java	Fixes validation to use the DB password hash instead of token-derived password.
datavines-core/src/main/java/io/datavines/core/utils/TokenManager.java	Removes hardcoded default secret, adds secret initialization logic, and updates validation to use BCrypt.

 [Add Copilot custom instructions](#) for smarter, more guided reviews. [Learn how to get started](#).



▼ datavines-server/src/main/resources/application.yaml

```

83 +
84 + jwt:
85 +   token:
86 +     secret: PLEASE_CHANGE_THIS_TO_A_RANDOM_STRING

```



**Copilot** AI 3 weeks ago

The default `jwt.token.secret` value committed here is a predictable shared secret. If a deployment uses this file without changing it, attackers can forge JWTs (the original issue still exists, just moved from code to config). Prefer leaving the secret unset and requiring an environment/secret-manager provided value (or at least treat this placeholder as invalid at startup).

Suggested change

```

86 -   secret: PLEASE_CHANGE_THIS_TO_A_RANDOM_STRING
86 +   secret: ${JWT_TOKEN_SECRET}

```

  [Copilot](#) uses AI. Check for mistakes.

▼ datavines-core/src/main/java/io/datavines/core/utils/TokenManager.java

Comment on lines +56 to +60

```

56 +         if (StringUtils.isEmpty(tokenSecret) || tokenSecret.length() < 16) {
57 +             log.warn("jwt.token.secret is not configured or too short, gener
58 +                 tokenSecret = java.util.UUID.randomUUID().toString().replace("-"
59 +                     + java.util.UUID.randomUUID().toString().replace("-"
60 +         }

```

**Copilot** AI 3 weeks ago

Auto-generating a random JWT secret at startup has significant operational impact: tokens become invalid after restart, and in a multi-instance/clustered deployment each node will generate a different secret so tokens may fail validation depending on which node receives the request. Consider failing fast when `jwt.token.secret` is missing/weak and requiring a consistent externally configured secret (and/or explicitly documenting the single-node-only behavior).

Suggested change

```

56 -         if (StringUtils.isEmpty(tokenSecret) || tokenSecret.length() <
      16) {
57 -             log.warn("jwt.token.secret is not configured or too short,
      generating random secret");
58 -             tokenSecret =
      java.util.UUID.randomUUID().toString().replace("-", "")
59 -                 +
      java.util.UUID.randomUUID().toString().replace("-", "");
60 -         }

56 +         if (StringUtils.isBlank(tokenSecret)) {
57 +             throw new IllegalStateException("jwt.token.secret must be
      configured and must not be empty");
58 +         }
59 +         if (tokenSecret.length() < 16) {
60 +             throw new IllegalStateException("jwt.token.secret must be at
      least 16 characters long");
61 +         }

```

[Copilot](#) uses AI. Check for mistakes.

▼ datavines-core/src/main/java/io/datavines/core/utils/TokenManager.java

```

164 | 176 |         String tokenUsername = getUsername(token);
165 | 177 |         String tokenPassword = getPassword(token);
166 |     |         return (username.equals(tokenUsername) && password.equals(tokenPassw
178 |     |         return (username.equals(tokenUsername) && BCrypt.checkpw(tokenPasswo

```

**Copilot** AI 3 weeks ago

`BCrypt.checkpw(tokenPassword, password)` will likely throw if `tokenPassword` is null/empty (e.g., token missing the `up` claim, or `getPassword()` failed). Previously `password.equals(tokenPassword)` would safely return false. Consider guarding for missing/blank `tokenUsername` / `tokenPassword` and returning false instead of letting an exception bubble up as a 500.

## Suggested change

```

181 -     return (username.equals(tokenUsername) &&
      BCrypt.checkpw(tokenPassword, password) && !(isExpired(token)));
181 +     if (StringUtils.isBlank(username) ||
      StringUtils.isBlank(password)
182 +         || StringUtils.isBlank(tokenUsername) ||
      StringUtils.isBlank(tokenPassword)) {
183 +         return false;
184 +     }
185 +     return (StringUtils.equals(username, tokenUsername)
186 +         && BCrypt.checkpw(tokenPassword, password)
187 +         && !(isExpired(token)));

```

  [Copilot](#) uses AI. Check for mistakes.

▼ `datavines-core/src/main/java/io/datavines/core/utils/TokenManager.java`

Comment on lines 175 to +178

163	175		<code>public boolean validateToken(String token, String username, String passw</code>
164	176		<code>String tokenUsername = getUsername(token);</code>
165	177		<code>String tokenPassword = getPassword(token);</code>
166		-	<code>return (username.equals(tokenUsername) &amp;&amp; <u>password.equals(tokenPassw</u></code>
	178	+	<code>return (username.equals(tokenUsername) &amp;&amp; <u>BCrypt.checkpw(tokenPasswo</u></code>

 **Copilot** AI [3 weeks ago](#)

This validation approach still depends on the JWT carrying the user's plaintext password (`tokenPassword` comes from the token claims). That creates a high-impact credential exposure risk if a token is leaked, and it also makes per-request validation expensive (BCrypt on every request). Consider removing password from JWT claims entirely and validating via standard JWT claims (subject/userId) + signature/expiry, optionally with a server-side token revocation/version check.

  [Copilot](#) uses AI. Check for mistakes.

▼ `datavines-core/src/main/java/io/datavines/core/utils/TokenManager.java`

Comment on lines 175 to 176

163	175	public boolean validateToken(String token, String username, String passw
164	176	String tokenUsername = getUsername(token);



**Copilot** AI 3 weeks ago

validateToken(..., String password) now expects the stored BCrypt hash (from user.getPassword() ), not a plaintext password. Renaming this parameter (e.g., passwordHash ) would reduce confusion and help prevent accidental misuse in future call sites.

👍 🗨️ [Copilot](#) uses AI. Check for mistakes.

**jackieya** added 2 commits 3 weeks ago

[Fix][Security] Add null guard for tokenPassword in validateToken to ... [e540d6d](#)

[Fix][Build] Add missing jodd-core dependency to datavines-core modul... [30d7a77](#)

[Sign up for free](#) to join this conversation on **GitHub**. Already have an account? [Sign in to comment](#)

**Reviewers**

**Copilot**



**Assignees**

No one assigned

**Labels**

None yet

**Projects**

None yet

**Milestone**

No milestone

**Development**

Successfully merging this pull request may close these issues.

4/26/26, 4:56 AM

[Fix][Security] Fix JWT authentication bypass via hardcoded secret and self-comparison logic by jackieya · Pull Re...

None yet

---

2 participants

