# **PREDECT Project**

# **Security Vulnerabilities**

## **1. Environment variable injection risk (Low risk)**

**Issue**: Environment variable placeholders might be vulnerable to injection.

"${ENV}" in 202206221918-mq-config-carriers-dml.sql (lines 7, 11)

"${ENV}" in 202208031010-predict-feed2.sql (line 14)

**Remediation**:

* Use prepared statements with variable binding rather than string concatenation

Implement a secure environment variable substitution mechanism:  
 sql  
*-- Instead of string template*

INSERT INTO public.mq\_config (..., s3\_dirty, ...)

VALUES (..., 'dsa-cdl-s3-borderforce-predict-dirty-${ENV}', ...);

*-- Use a parameter binding approach*

INSERT INTO public.mq\_config (..., s3\_dirty, ...)

* VALUES (..., ?, ...); *-- Bind parameter securely*
* Validate environment variable values against a specific pattern before use

## **2. Lack of statement parameterization (Low risk)**

**Issue**: SQL statements use direct value insertion rather than parameterization.

**Files affected**: All files with INSERT/UPDATE statements

**Remediation**:

Use prepared statements with parameter binding:  
 sql  
*-- Instead of direct insertion*

INSERT INTO public.cert\_config (id, description, ...)

VALUES (nextval('cert\_config\_id\_seq'), 'Test config', ...);

*-- Use parameterized queries (implementation depends on your database access library)*

INSERT INTO public.cert\_config (id, description, ...)

* VALUES (?, ?, ...); *-- Parameters bound securely*
* If using Liquibase, follow their parameter substitution patterns
* Consider using ORM frameworks that handle parameterization automatically

## **3. Insufficient data validation (Low Risk)**

**Issue**: Tables lack constraints for data validation.

**Files affected**:

* 202206221737-cert-config-ddl.sql
* 202308241405-response-audit-ddl.sql

**Remediation**:

* Add appropriate CHECK constraints for data validation
* Specify length limits for VARCHAR columns
* Add NOT NULL constraints for required fields

Implement table and column constraints:  
 sql  
*-- Instead of unconstrained columns*

CREATE TABLE IF NOT EXISTS cert\_config (

id BIGINT NOT NULL PRIMARY KEY,

description VARCHAR,

filer\_carrier\_id VARCHAR NOT NULL,

...

);

*-- Add proper constraints*

CREATE TABLE IF NOT EXISTS cert\_config (

id BIGINT NOT NULL PRIMARY KEY,

description VARCHAR(255),

filer\_carrier\_id VARCHAR(50) NOT NULL CHECK (filer\_carrier\_id ~ '^[A-Za-z0-9]+$'),

...

* );
* Consider using domain types for specialized data formats

## **4. Exposed internal infrastructure details (Low risk)**

**Issue**: Queue names and directory structures are exposed in the scripts.

**Files affected**:

* 202206221918-mq-config-carriers-dml.sql
* 202207011045-update-data-feed-filename-pattern.sql
* 202305191505-data-feed-fix.sql

**Remediation**:

* Store infrastructure details in external configuration
* Use reference keys instead of actual infrastructure paths

Implement a configuration lookup pattern:  
 sql  
*-- Instead of hard-coded queue names*

INSERT INTO public.mq\_config (..., inbound\_mq\_queue, outbound\_mq\_queue, ...)

VALUES (..., 'PREDICT.IN.CARA', 'PREDICT.OUT.CARA', ...);

*-- Use configuration references*

INSERT INTO public.mq\_config (..., inbound\_mq\_queue, outbound\_mq\_queue, ...)

* VALUES (..., get\_config('QUEUE\_CARA\_IN'), get\_config('QUEUE\_CARA\_OUT'), ...);
* Consider encrypting sensitive infrastructure details

## **5. Limited column constraints (Low Risk)**

Many columns in the created tables have no length restrictions or other constraints

**Issue**: VARCHAR columns lack length limits.

**Files affected**:

* 202206221737-cert-config-ddl.sql (lines 6-12)
* 202308241405-response-audit-ddl.sql (lines 9-18)

**Remediation**:

* Add appropriate length constraints to all VARCHAR columns

Implement data type constraints:  
 sql  
*-- Instead of unlimited VARCHAR*

CREATE TABLE IF NOT EXISTS cert\_config (

id BIGINT NOT NULL PRIMARY KEY,

description VARCHAR,

filer\_carrier\_id VARCHAR NOT NULL,

...

);

*-- Add length constraints*

CREATE TABLE IF NOT EXISTS cert\_config (

id BIGINT NOT NULL PRIMARY KEY,

description VARCHAR(255),

filer\_carrier\_id VARCHAR(50) NOT NULL,

...

* );
* Consider using specialized data types for specific formats (e.g., UUID)

## 

6. **SWEET32 Attack on 3DES (Low Risk)**

**Description:**

The SWEET32 attack (CVE-2016-2183) is a "birthday attack" targeting 64-bit block ciphers like Triple-DES (3DES). By capturing a large volume of encrypted data (approx. 32GB) from a long-lived session using the same key, an attacker can find "collisions" (identical encrypted blocks). These collisions can allow a man-in-the-middle attacker to slowly recover small portions of plaintext, such as session cookies.

**Impact:**

The primary impact is a **loss of confidentiality**, potentially leading to the decryption of sensitive data like authentication tokens or session cookies. This could result in **session hijacking**. It undermines the security of encrypted communications, even though the attack is complex and requires capturing substantial data.

**Remediation:**

The core solution is to **disable 3DES and other 64-bit block ciphers**. Instead, configure your server to **use strong, modern ciphers like AES (128-bit or 256-bit block size)**. This involves updating server configurations (e.g., web server, VPN) and potentially operating system settings (like Windows Registry for Schannel). Ensure systems are patched and regularly audited.

**Patching related vulnerabilities includes:**

**Curl Vulnerabilities**

**1. CVE-2022-32207: Curl Improper Permission Preservation**

* **Description:** When curl saves cookies, alt-svc, and hsts data to local files, it uses a rename operation from a temporary name. This operation could unintentionally widen the permissions for the target file, making it accessible to more users than intended.
* **Impact:** Potential information disclosure if sensitive data within these files (e.g., cookies) becomes accessible to unauthorised local users.
* **CVSS Rating:** Medium (Severity as per curl project). NVD may provide a specific numerical score.
* **Remediation:**
  + Upgrade curl to version 7.84.0 or later.
  + Apply patches if upgrading is not immediately possible.
  + Implement strict umask settings to protect saved files.

**2. CVE-2023-23914: Curl HSTS Check Bypass**

* **Description:** Curl's HSTS (HTTP Strict Transport Security) functionality could fail when multiple URLs are requested serially on the same command line. The HSTS state from a previous transfer might not be correctly applied to subsequent HTTP requests, causing them to proceed over insecure HTTP instead of being upgraded to HTTPS. This is a command-line tool issue and does not affect libcurl.
* **Impact:** Limited confidentiality and integrity impact, as sensitive information could be transmitted in cleartext if an HTTP URL is used instead of HTTPS, despite HSTS normally preventing this.
* **CVSS Rating:** Low (Severity as per curl project and Red Hat).
* **Remediation:**
  + Upgrade curl to version 7.88.0 or later.
  + Ensure all URLs are specified with HTTPS:// rather than HTTP:// in command lines.

**Zlib Vulnerability**

**3. CVE-2022-37434: Zlib Heap-Based Buffer Overflow**

* **Description:** A heap-based buffer overflow vulnerability exists in the inflate.c component of zlib. This can be triggered by processing a specially crafted gzip file with a large gzip header extra field, leading to improper bounds checking. Applications using zlib to handle such files (e.g., rsync) could be affected.
* **Impact:** Potential for denial of service (application crash) or arbitrary code execution with the privileges of the affected application. The impact on confidentiality and integrity might be limited in some contexts (like rsync as per Red Hat), but availability impact can be high.
* **CVSS Rating:** CVSS v3.1 scores vary: 9.8 (Critical - NVD), 7.3 (IBM), 7.0 (RHEL). The higher score suggests a significant potential impact.
* **Remediation:**
  + Update the zlib library to a patched version (e.g., zlib 1.2.12 had fixes, but check specific product advisories for the correct patched version for your software).
  + Apply updates and patches for affected software that bundles or uses the zlib library.

**Golang Crypto Vulnerability**

**4. CVE-2024-45337: Golang Crypto SSH Authorization Bypass**

* **Description:** A vulnerability in golang.org/x/crypto's SSH implementation, specifically related to the ServerConfig.PublicKeyCallback functionality. The callback might be called with multiple keys, and the order of keys cannot reliably indicate which key was used for authentication. An attacker could potentially send multiple public keys, authenticate with one, while the application incorrectly makes authorization decisions based on a different, attacker-controlled key.
* **Impact:** Potential authorization bypass in SSH server implementations using the vulnerable library. This could lead to unauthorised access, disclosure of sensitive information, and data modification.
* **CVSS Rating:** 9.1 (Critical - NVD CVSS v3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N).
* **Remediation:**
  + Update golang.org/x/crypto to version v0.31.0 or later.
  + Review and adjust application logic to correctly handle public key authentication, possibly using the Extensions field of the Permissions return value from authentication callbacks and retrieving successful authentication state via ServerConn.Permissions.

**Apache MINA Vulnerability**

**5. CVE-2024-52046: Apache MINA Deserialisation RCE**

* **Description:** A deserialisation of untrusted data vulnerability exists in the ObjectSerializationDecoder of Apache MINA. It uses Java's native deserialisation protocol without sufficient security checks when processing incoming serialised data via the IoBuffer#getObject() method, typically in conjunction with ProtocolCodecFilter and ObjectSerializationCodecFactory.
* **Impact:** Successful exploitation by sending specially crafted malicious serialised data could allow an attacker to execute arbitrary code remotely on the server.
* **CVSS Rating:** 9.8 (Critical - NVD CVSS v3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H). SOCRadar also notes a CVSSv3 of 9.8/10.
* **Remediation:**
  + Upgrade Apache MINA core to versions 2.0.27, 2.1.10, or 2.2.4 or later.
  + Crucially, after upgrading, explicitly configure the ObjectSerializationDecoder to whitelist only the expected classes for deserialisation using methods like accept(ClassNameMatcher), accept(Pattern), or accept(String... patterns). By default, after patching, the decoder will reject all classes if not explicitly allowed.
  + If upgrading is not immediately possible, implement strict input validation and filtering for serialised data and avoid using ObjectSerializationDecoder or limit its use to trusted sources.

**AsyncHttpClient Vulnerability**

**6. CVE-2024-53990: AsyncHttpClient CookieStore User Cookie Clobbering**

* **Description:** A flaw in the AsyncHttpClient (AHC) library where its self-managed CookieStore can silently replace explicitly defined cookies with those from the cookie jar if they share the same name.
* **Impact:** In services operating with multiple users, this could lead to one user's cookie being used for another user's requests. This could result in improper authentication, unauthorised access, session hijacking, or impersonation.
* **CVSS Rating:** 8.1 (High - Red Hat CVSS v3.1: AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H). Feedly also mentions a CVSS v3.1 base score of 8.1.
* **Remediation:**
  + Update to a patched version of the AsyncHttpClient library (check vendor advisories for specific version numbers).
  + If immediate patching is not possible:
    - Consider disabling the automatic CookieStore feature and managing cookies manually within the application.
    - Implement a custom CookieStore that does not replace cookies based on name.
    - Review and audit the use of AsyncHttpClient in multi-user environments.

**Apache Tomcat Vulnerability**

**7. CVE-2025-24813: Apache Tomcat Path Equivalence and Deserialisation Vulnerability**

* **Description:** This vulnerability involves path equivalence ('file.name' internal dot issues) and potential deserialisation of untrusted data in Apache Tomcat. It can occur under specific, non-default configurations. For example, if writes are enabled for the default servlet (disabled by default), support for partial PUT is enabled (enabled by default), and potentially if Tomcat's file-based session persistence is used with the default storage location and an application includes a library that can be leveraged in a deserialisation attack.
* **Impact:** An attacker could potentially achieve remote code execution (RCE), view security-sensitive files, or inject content into those files.
* **CVSS Rating:** Not explicitly stated in the snippets as a single NVD score, but NHS England Digital rates the threat severity as "Medium". However, the potential for RCE suggests it could be higher depending on the configuration.
* **Affected Versions (examples, check advisory for full list):**
  + Apache Tomcat 11.0.0-M1 to 11.0.2 (fixed in 11.0.3 or later)
  + Apache Tomcat 10.1.0-M1 to 10.1.34 (fixed in 10.1.35 or later)
  + Apache Tomcat 9.0.0.M1 to 9.0.98 (fixed in 9.0.99 or later)
* **Remediation:** 1

[1. Apache Tomcat CVE-2025-24813: What You Need to Know | Rapid7 Blog](https://www.rapid7.com/blog/post/2025/03/19/etr-apache-tomcat-cve-2025-24813-what-you-need-to-know/" \t "_blank)

[www.rapid7.com](https://www.rapid7.com/blog/post/2025/03/19/etr-apache-tomcat-cve-2025-24813-what-you-need-to-know/" \t "_blank)

* + Upgrade to the fixed versions of Apache Tomcat.
  + Ensure secure configurations:
    - Keep writes for the default servlet disabled (default state).
    - Review usage of partial PUT if not strictly necessary.
    - If using file-based session persistence, ensure the storage location is secure and consider alternatives if possible.
    - Avoid including libraries susceptible to deserialisation attacks if possible, or ensure data being deserialised is from trusted sources.
  + Monitor systems for suspicious activity, especially commands launched by the webserver process.