Databasir 1.01 has Remote Code Execution vulnerability.

Moderate **vran-dev** published **GHSA-5r2v-wcwh-7xmp** on Apr 18

Package

No package listed

Affected versions

<= 1.0.1

Patched versions

> 1.0.1 or latest

Description

Impact

Databasir is a team-oriented relational database model document management platform.
Databasir 1.01 has remote code execution vulnerability.
Remote code execution vulnerability is a Web security vulnerability, we can execute any command, such as `whoami`.

Patches

(https://github.com/vran-dev/databasir/blob/master/core/src/main/java/com/databasir/core/infrastructure/connection/CustomDatabaseConnectionFactory.java)

```
@Override
    public Connection getConnection(Context context) throws SQLException {
        DatabaseTypePojo type =
databaseTypeDao.selectByDatabaseType(context.getDatabaseType());
        File driverFile;
        try {
            driverFile = driverResources.download(context.getDatabaseType(),
type.getJdbcDriverFileUrl());
        } catch (IOException e) {
            log.error("download driver error " + context, e);
            throw DomainErrors.DOWNLOAD_DRIVER_ERROR.exception(e.getMessage());
        }
```

```
        URLClassLoader loader = null;
        try {
            loader = new URLClassLoader(
                    new URL[]{
                            driverFile.toURI().toURL()
                    },
                    this.getClass().getClassLoader()
            );
        } catch (MalformedURLException e) {
            log.error("load driver error " + context, e);
            throw DomainErrors.CONNECT_DATABASE_FAILED.exception(e.getMessage());
        }

        Class<?> clazz = null;
        Driver driver = null;
        try {
            clazz = Class.forName(type.getJdbcDriverClassName(), true, loader);
            driver = (Driver) clazz.getConstructor().newInstance();
        } catch (ClassNotFoundException e) {
            log.error("init driver error", e);
            throw DomainErrors.CONNECT_DATABASE_FAILED.exception("驱动初始化异常，请检查 Driver
name: " + e.getMessage());
        } catch (InvocationTargetException
                | InstantiationException
                | IllegalAccessException
                | NoSuchMethodException e) {
            log.error("init driver error", e);
            throw DomainErrors.CONNECT_DATABASE_FAILED.exception("驱动初始化异常: " +
e.getMessage());
        }

        String urlPattern = type.getUrlPattern();
        String jdbcUrl = urlPattern.replace("{{jdbc.protocol}}", type.getJdbcProtocol())
                .replace("{{db.url}}", context.getUrl())
                .replace("{{db.name}}", context.getDatabaseName())
                .replace("{{db.schema}}", context.getSchemaName());
        Properties info = new Properties();
        info.put("user", context.getUsername());
        info.put("password", context.getPassword());
        return driver.connect(jdbcUrl, info);
    }
```

Workarounds

Can will be affected by the first source (https://github.com/vran-dev/databasir/blob/master/core/src/main/java/com/databasir/core/infrastructure/connection/CustomDatabaseConnectionFactory.java), one of the 62 to 76 line commented out.

References

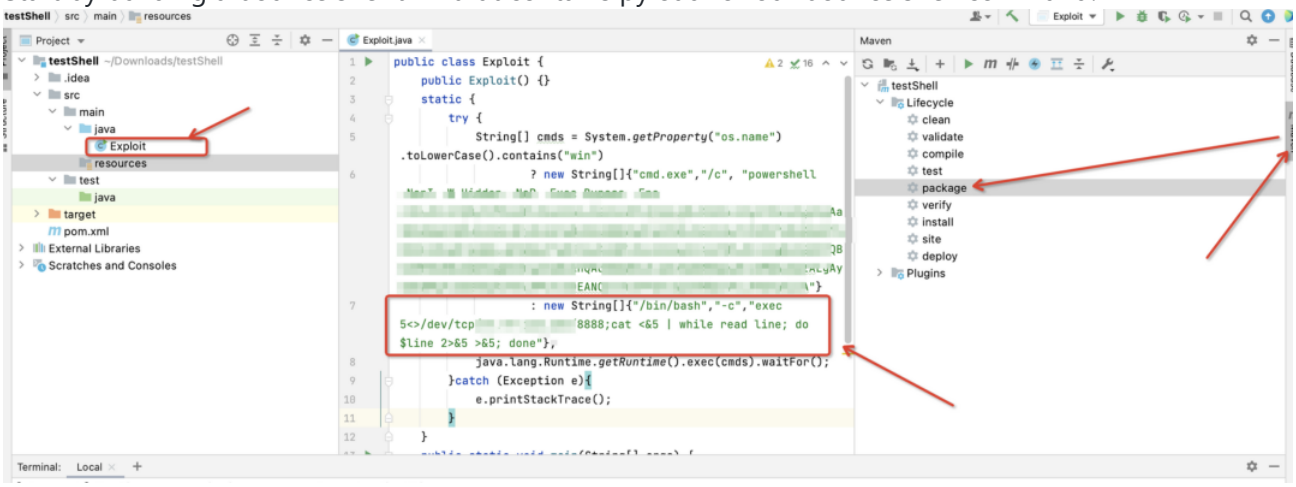
None

For more information

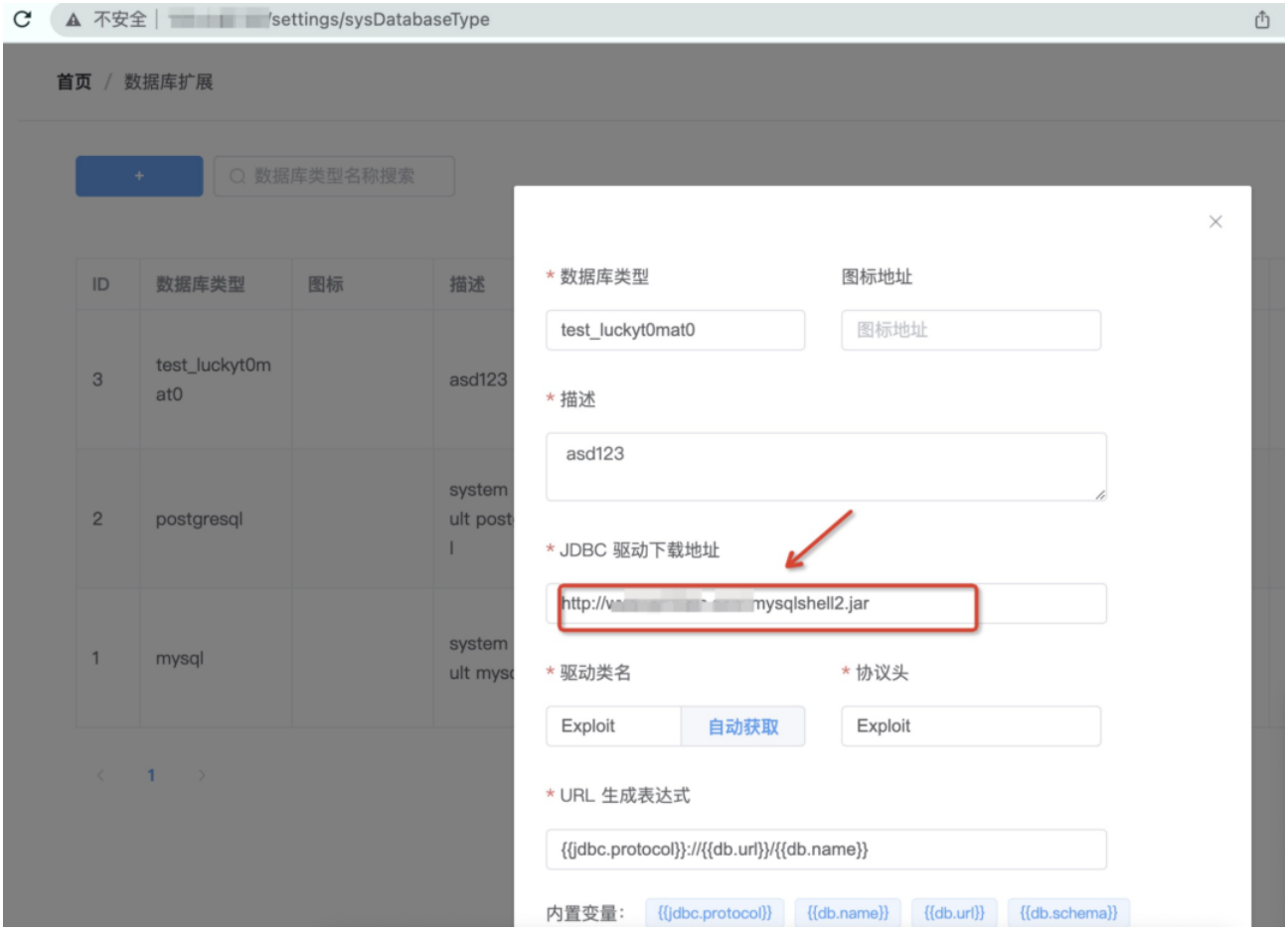
Affected source code: https://github.com/vran-dev/databasir/blob/master/core/src/main/java/com/databasir/core/infrastructure/connection/CustomDatabaseConnectionFactory.java, the vulnerability is located at the function point of [database extension - JDBC driver download address], which can cause arbitrary code execution through remote loading of JAR package (the part of jar package loading code needs to be verified) to be read with the code on lines 62 through 76:

```
        Class<?> clazz = null;
        Driver driver = null;
        try {
            clazz = Class.forName(type.getJdbcDriverClassName(), true, loader);
            driver = (Driver) clazz.getConstructor().newInstance();
        } catch (ClassNotFoundException e) {
            log.error("init driver error", e);
            throw DomainErrors.CONNECT_DATABASE_FAILED.exception("驱动初始化异常，请检查 Driver
name: " + e.getMessage());
        } catch (InvocationTargetException
                | InstantiationException
                | IllegalAccessException
                | NoSuchMethodException e) {
            log.error("init driver error", e);
            throw DomainErrors.CONNECT_DATABASE_FAILED.exception("驱动初始化异常: " +
e.getMessage());
        }
```

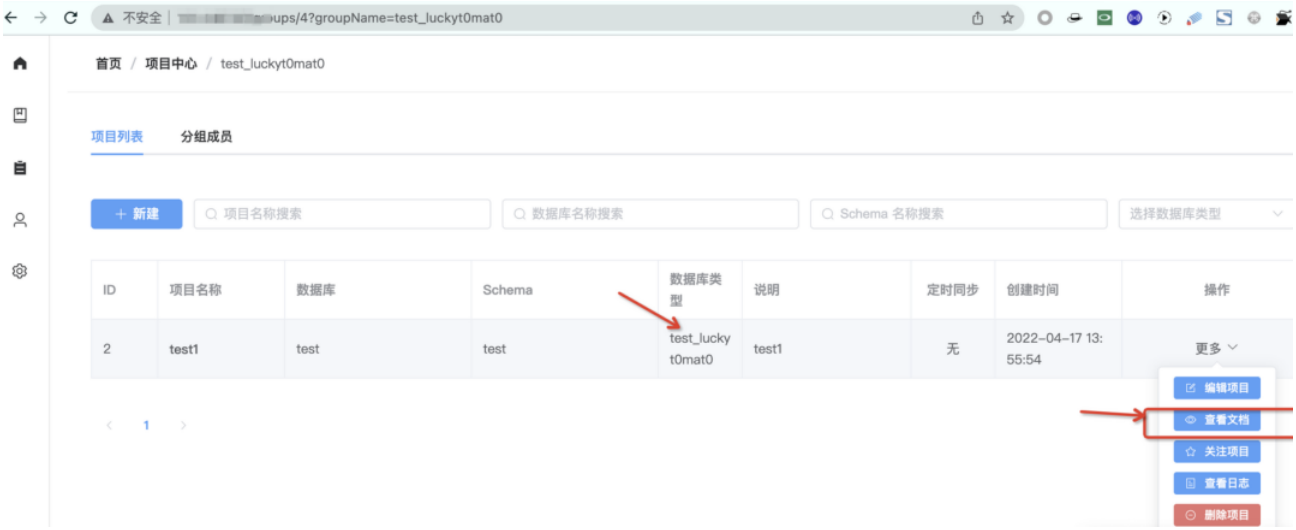
Start by building a bounce shell JAR that contains pyload for our bounce shell command.



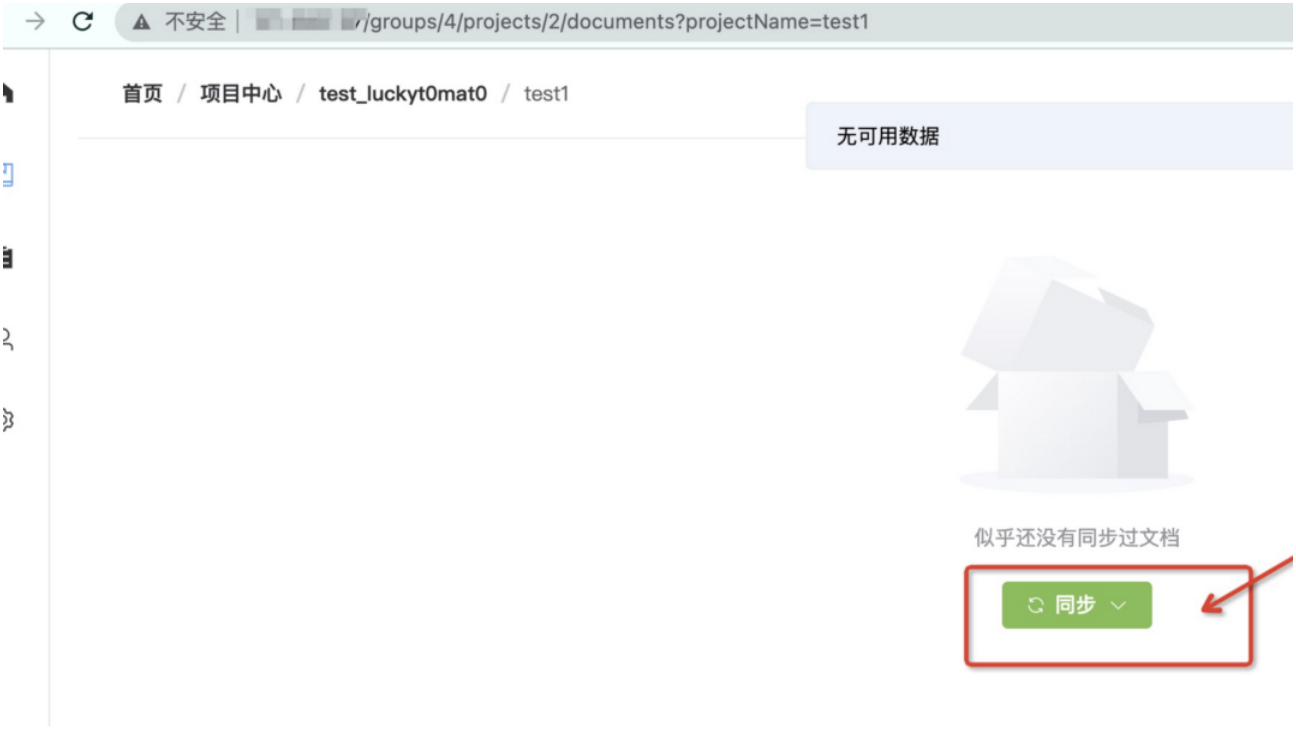
When the build is complete, create a database extension called test_luckyt0mat0 in the [Database extension-JDBC Driver Download Address] function and fill in the JAR package address.



Then create a new project in the project center and introduce our database extension called test_luckyt0mat0.



Then enter the project and click Sync.



Get Shell!

```
root@iZbp1hraxr5f0bggznt8h7Z:~# nc -lvvp 8888
Listening on [0.0.0.0] (family 0, port 8888)
Connection from [redacted] 64291 received!
whoami
root
```

Severity

Moderate

CVE ID

CVE-2022-24861

Weaknesses

CWE-78

Credits

LuckyT0mat0