XXE, SSRF and information leak in Jenkins Job Import Plugin <= 2.1

Security advisory

2019-01-30

Thomas Chauchefoin
Julien Szlamowicz

Vulnerability description

Presentation of Job Import

This *Jenkins* plugin allows importing jobs from other *Jenkins* instances. At the time of writing of this advisory, this software sums up 10804 installs.[1]

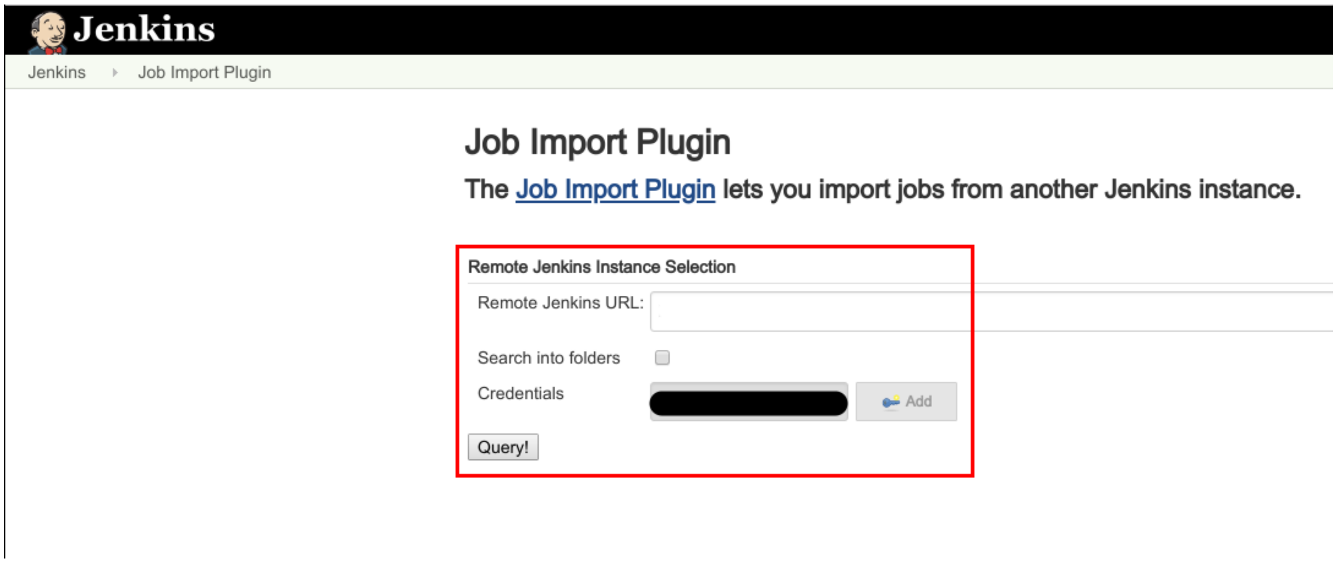

Figure 1: Interface of the plugin.

Issues

Synacktiv discovered that this plugin **does not correctly configure the XML engine** used to process the job file coming from the remote instance. It allows attackers to use arbitrary XML external entities that are then processed by the XML engine. This could lead to varieties of actions, such as:

- Denial of service;
- Reading arbitrary files on the server;
- Scanning the internal network from this host.

Indeed, user-controlled XML values are parsed using the `javax.xml.parsers.DocumentBuilderFactory` without security precautions. No configuration parameter is specified when initializing the document builder, thus *doctypes* and entities (including external ones) are automatically processed and resolved.

In addition, it has been found that this plugin can be used **to perform arbitrary HTTP GET requests to any host** (*Server Side Request Forgery*), including the local host and the ones within the same network segment. This behavior can be abused by querying internal services that do not require any kind of authentication (several *NoSQL* databases, caches, etc).

As a side effect, by performing such requests to a host controlled by the attacker, **it is also possible to obtain credentials from *Jenkin*'s vault**, even if they are not owned by the user.

Affected versions

The following version is known to be affected:

1 https://plugins.jenkins.io/job-import-plugin

- *job-import-plugin* 2.1

Mitigation

Upgrade to *Job Import* 3.1.

Timeline

Date	Action
2018-05-25	Advisory sent to *Jenkins* security team (through *https://issues.jenkins-ci.org*)
2018-05-30	Release of the version 3.0, silently fixing some issues described in this document.
2019-01-28	Release of the version 3.1, fixing all the issues.
2019-01-30	Publication of this advisory.

Technical description and proof-of-concept

In `client/restApiClient.java`, the class `RestApiClient` implements a method named `getRemoteItems`. It is in charge of fetching and parsing a XML document originating from the remote *Jenkins* instance:

```
public final class RestApiClient {

    private static final Logger LOG = Logger.getLogger(RestApiClient.class.getName());

    public static List<RemoteItem> getRemoteItems(RemoteFolder parent, String url,
CredentialsUtils.NullSafeCredentials credentials, boolean recursiveSearch) {
        List<RemoteItem> items = new ArrayList<>();
        try {
            if (StringUtils.isNotEmpty(url)) {
                Document doc =
DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(
                        URLUtils.fetchUrl(url + Constants.XML_API_QUERY,
credentials.username, credentials.password));
                NodeList nl = doc.getElementsByTagName("job");
```

The parameter *url* is fully controlled by the user and the only constraint is that it should not be empty. The resource at this URL is directly given to `DocumentBuilder.parse()`. However, by default, `javax.xml.parsers.DocumentBuilderFactory` does not prevent the declaration of additional *doctypes* and usage of XML external entities, leading to a XXE vulnerability.

As already stated, `getRemoteItems` does not try to apply restrictions on the server pointed by the *url* parameter, even in the implementation of `URLUtils.fetchUrl` (`utils/URLUtils.java`):

```
public final class URLUtils {
    public static InputStream fetchUrl(String url, String username, String password) throws
MalformedURLException, IOException {
        notNull(url);
        notNull(username);
        notNull(password);
        URLConnection conn = new URL(url).openConnection();
        if (!username.isEmpty()) {
            conn.setRequestProperty("Authorization", "Basic " +
DatatypeConverter.printBase64Binary((username + ":" + password).getBytes()));
        }
        return conn.getInputStream();
    }
```

It is then possible to force the plugin to perform GET requests to any host of the same network segment, Internet or the local host itself. If the attacker chooses to attach credentials from *Jenkin's vault* to its request, they will be sent in the `Authorization` header.