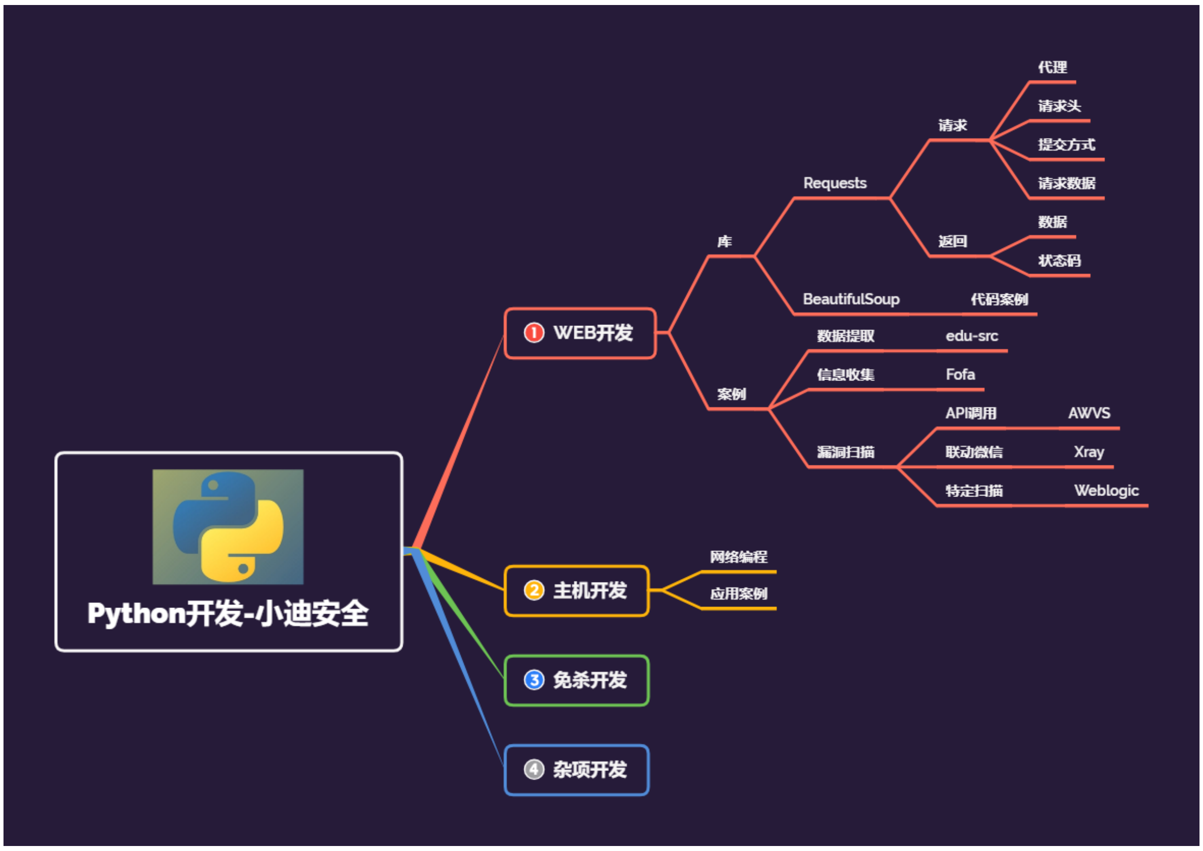
Day161　安全开发-Python-红队项目&漏扫调用&API推送微信&任务自动添加并启动



1.知识点

- 1、Python-应用方向-红队项目
- 2、Python-Xray&Awvs&SQLMAP
- 3、Python-推送微信&自动添加&启动

2.演示案例

2.1 Python-红队项目-Xray调用推送微信

```
漏扫API调用-Xray
参考：https://docs.xray.cool/#/webhook/webhook
应用案例：可通过自动化扫描后将实时结果进行微信推送,也可以应用在其他安全工具上。
xray webscan --url http://x.x.x.x --webhook-output http://127.0.0.1:5000/webhook
1、命令漏扫触发本地URL
2、Flask启动进行监听处理
3、借助Server酱API推送微信
```

```python
from flask import Flask, request
import requests

app = Flask(__name__)

@app.route('/webhook', methods=['POST'])
def xray_webhook():
    url = 'https://sctapi.ftqq.com/你的KEY.send?title=Xray find vuln!!!'
    try:
        #接受传递过来的数据转换json格式
        vuln=request.json
        content = """## xray 发现了新漏洞
        url: {url}
        插件: {plugin}
        漏洞类型: {vuln_class}
        请及时查看和处理
        """.format(url=vuln['data']['target']['url'], plugin=vuln['data']['plugin'],vuln_class=vuln['type'])
        print(content)
        data={
            'desp':content
        }
        print(data)
        requests.post(url,data=data)
        return 'ok'
    except Exception as e:
        pass


if __name__ == '__main__':
    app.run()
```

2.2 Python-红队项目-Awvs调用自动添加

```
漏扫API调用-AWVS
参考：https://blog.csdn.net/wy_97/article/details/106872773
应用案例：可通过脚本调用AWVS自动添加扫描,也可以应用在其他安全工具上。
1、启动工具&开启API-KEY
2、创建新任务并记录任务ID
3、启动新任务并记录返回ID
```

```
import requests

def new_id(key,url):
```

```python
    api_add_url = "https://localhost:3443/api/v1/targets"
    headers = {
        'X-Auth': key,
        'Content-type': 'application/json'
    }
    data = '{"address":"%s","description":"create_by_reaper","criticality":"10"}'%url
    r = requests.post(url=api_add_url, headers=headers, data=data, verify=False).json()
    id=r['target_id']
    if id is not None:
        print(url+'->任务添加成功，等待启动...')
        print(url + '->任务ID->'+id)
        return id

def start_id(key,id):
    # 核心代码
    url = 'https://localhost:3443/api/v1/scans'
    headers = {
        'X-Auth': key,
        'Content-type': 'application/json'
    }
    data = '{"profile_id":"11111111-1111-1111-1111-111111111111","schedule":{"disable":false,"start_date":null,"time_sensitive":false},"target_id":"%s"}' % id
    r = requests.post(url=url, headers=headers, data=data, verify=False).json()
    if r['scan_id'] is not None:
            print('任务ID->'+r['target_id']+"->任务启动成功，等待完毕...")

if __name__ == '__main__':
    key='你的key'
    for url in open('url.txt'):
        id=new_id(key,url.replace('\n',''))
        start_id(key,id)
```

2.3 Python-红队项目-SQLMAP调用自动添加

```
漏扫API调用-SQLMAP
参考：https://www.freebuf.com/articles/web/204875.html
应用案例：前期通过信息收集拿到大量的URL地址，这个时候可以配合SqlmapAPI接口进行批量的注入检测。
开发当前项目过程：（利用sqlmapapi接口实现批量URL注入安全检测）
0.启用sqlmap-API服务  python sqlmapapi.py -s
1.创建新任务记录任务ID     @get("/task/new")
2.设置任务ID扫描信息     @post("/option/<taskid>/set")
3.开始扫描对应ID任务     @post("/scan/<taskid>/start")
4.读取扫描状态判断结果     @get("/scan/<taskid>/status")
5.如果结束删除ID并获取结果     @get("/task/<taskid>/delete")
6.扫描结果查看 @get("/scan/<taskid>/data")
```

```
import requests,time,json

```

```python
#0.启用sqlmap-API服务  python sqlmapapi.py -s
#1.创建新任务记录任务ID    @get("/task/new")
#2.设置任务ID扫描信息 @post("/option/<taskid>/set    ")
#3.开始扫描对应ID任务 @post("/scan/<taskid>/start")
#4.读取扫描状态判断结果 @get("/scan/<taskid>/status")
#5.如果结束删除ID并获取结果 @get("/task/<taskid>/delete")
#6.扫描结果查看 @get("/scan/<taskid>/data")

def new_id():
    headers = {
        'Content-Type': 'application/json'
    }
    url='http://127.0.0.1:8775'+'/task/new'
    resp=requests.get(url,headers=headers).json()
    taskid=resp['taskid']
    if resp['success'] is True:
        print('->1、创建任务ID成功，ID:' + taskid)
        return taskid

def set_id(id,scanurl):
    headers = {
        'Content-Type': 'application/json'
    }
    data={
        'url':scanurl
    }
    url = 'http://127.0.0.1:8775/option/%s/set' % id
    resp = requests.post(url, data=json.dumps(data),headers=headers).json()
    if resp['success'] is True:
        print('->2、设置任务ID成功，ID:' + taskid)
        print('->2、设置任务URL成功，URL:' + scanurl)
        return taskid

def scan_id(id,scanurl):
    headers = {
        'Content-Type': 'application/json'
    }
    data = {
        'url': scanurl
    }
    url = 'http://127.0.0.1:8775/scan/%s/start' % id
    resp = requests.post(url, data=json.dumps(data), headers=headers).json()
    if resp['success'] is True:
        print('->3、启动扫描任务ID成功，ID:%s'%id)
        print('->3、启动任务URL成功，URL:%s'%scanurl)

def status_id(id):
    url = 'http://127.0.0.1:8775/scan/%s/status'%id
    print('->4、扫描任务进行中，请等待结束，ID:%s' %id)
    while 1:
        resp = requests.get(url).text
        if 'running' in resp:
            #print(resp)
            continue
        else:
            print('->4、扫描任务ID已完成，ID:%s' %id)
            break

def data_id(id,scanurl):
    url = 'http://127.0.0.1:8775/scan/%s/data' % id
    resp = requests.get(url)
    #print(resp.json()['data'][0]['status'])
    if resp.json()['data'][0]['status'] == 1:
        print('--->存在注入---<:'+'\n'+scanurl)
    with open('result.txt', 'a+') as f:
        f.write(resp.text)
        f.write('\n' + '===========python sqlmapapi by xiaodisec=============' + '\n')
        f.write('-----------------------------------------------------' + '\n')
        f.close()
    print('->5、注入任务ID已完成，请查看结果：result.txt')

def delete_id(id):
    url = 'http://127.0.0.1:8775/task/%s/delete' % id
    resp = requests.get(url).json()
    if resp['success'] is True:
        print('->6、删除任务ID:%s成功' % id)
    time.sleep(3)

if __name__ == '__main__':
    for url in open('url.txt'):
        taskid = new_id()
        set_id(taskid,url.replace('\n',''))
        scan_id(taskid,url.replace('\n',''))
        status_id(taskid)
        data_id(taskid,url.replace('\n',''))
        delete_id(taskid)
        print('---------------------------------')
```