红队基本操作1-----poc快速编写以及现有poc快速魔改批量扫描

0x01 简介

##这个教程是基础教程，高手就不要看了，不要浪费时间，两个脚本写出来总共花了十分钟，代码优化度不够不要骂我，你可以自己去重新写一个。

这篇文章是基于上一篇文章衍生出来的文章，也就是ueditor那篇文章。

为啥要写，因为搭环境太傻逼了，我至今ueditor的环境也没有在本地搭建成功，直接公网找国外的站复现漏洞才是最快的。

因此有了poc以及魔改poc变成批量扫描版的这个教程，我看公网上关于这个魔改的教程也比较少，因此在这里记录一下。

0x02 思路

写说简单的思路，再来讲解为啥这么搞。

思路如下：

- **先写一个能够单体检测的poc。**
- **然后再写一个脚本，用外部套壳的方法来实现批量功能。**

这里直接拿ueditor举例子

这个是我写的单体检测脚本

exp.py > work

```python
import requests
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("-u", help="Input the url that u wanna check")
args = parser.parse_args()
url = args.u


def work(url):
    url = url + "/ueditor/net/controller.ashx?action=catchimage"
    response = requests.get(url)
    if response.status_code == 200:
        print(url + "--get!")
    else:
        print(url + "--failed!")

def main():
    work(url)

if __name__ == "__main__":
    main()

```

这个是外部套壳脚本

```python
#!/usr/bin/python3
# coding: utf-8

import subprocess
import threadpool

def open_file(filename):
    with open(filename, 'r', encoding='UTF-8') as f:
        filecontent = f.read()
    return filecontent

def work(target):
    cmd = ["python", "exp.py", "-u", target]
    rsp = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    output, error = rsp.communicate()
    output = output.decode('gbk', 'ignore')
    print(output)


def main():
    filename = "ip.txt"
    filecontent = open_file(filename)
    filecontent = filecontent.split("\n")
    # print(filecontent)

    pool = threadpool.ThreadPool(10)
    requests = threadpool.makeRequests(work,filecontent)
    [pool.putRequest(req) for req in requests]
    pool.wait()

    # for i in filecontent:
    #     work(i)


if __name__ == '__main__':
    main()
```

先说下单体检测

主体就是这个部分，非常的简单

```python
def work(url):
    url = url + "/ueditor/net/controller.ashx?action=catchimage"
    response = requests.get(url)
    if response.status_code == 200:
        print(url + "--get!")
    else:
        print(url + "--failed!")
```

直接请求指纹，判断是否为200即可

然后外部传参调用的是这个argparse，非常的方便，比原生的input好用的多。

使用起来是下面这个样子

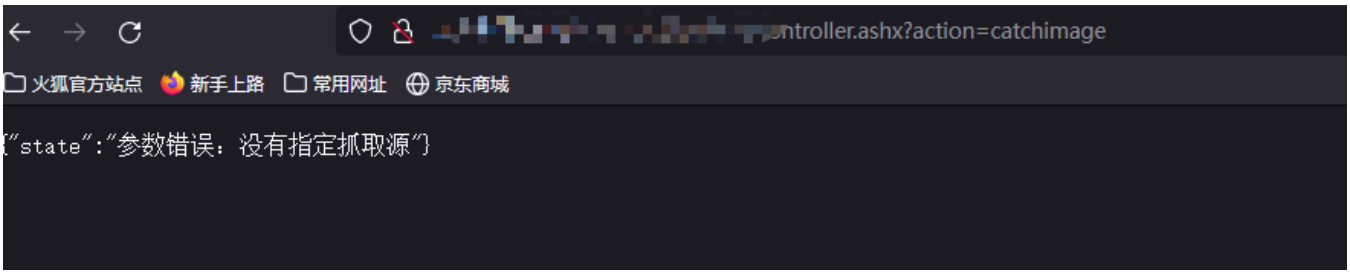
```
C:\mystuff\workspace\python>python exp.py -u "http://[redacted]:8090"
```

回显的结果是这个样子

```
C:\mystuff\workspace\python>python exp.py -u "[redacted]990"
[redacted]/ueditor/net/controller.ashx?action=catchimage--get!
```

这里就已经检出漏洞了

访问验证一下



验证成功

ok那么单体的功能就已经实现完毕了，但是现在需要在公网做批量扫描，还要加上批量功能。

但是这里存在一个问题，比如现在有很多个poc，可能是你自己写的，也可能是别人写的，每个poc如果都要使用嵌入新代码的方式来进行扫描，是不是贼恐怖?

因为你得写很多代码，大量重复的代码。

所以这里最优的方法就是用框架集成poc，然后poc统一输入输出。

但是如果是小脚本就没必要那么规范了，直接干就完事了，但是这里又不想改别人的代码，怕出bug，因此采用外部调用的方法。

下面讲一下这个批量脚本

```python
#!/usr/bin/python3
# coding: utf-8

import subprocess
import threadpool

def open_file(filename):
    with open(filename, 'r', encoding='UTF-8') as f:
        filecontent = f.read()
    return filecontent

def work(target):
    cmd = ["python", "exp.py", "-u", target]
    rsp = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    output, error = rsp.communicate()
    output = output.decode('gbk', 'ignore')
    print(output)
```

打开本地文件，这里放很多个ip

直接调用外部的脚本方法，然后把结果打印出来

```python
def main():
    filename = "ip.txt"
    filecontent = open_file(filename)
    filecontent = filecontent.split("\n")
    # print(filecontent)

    pool = threadpool.ThreadPool(10)
    requests = threadpool.makeRequests(work,filecontent)
    [pool.putRequest(req) for req in requests]
    pool.wait()

    # for i in filecontent:
    #     work(i)


if __name__ == '__main__':
    main()
```

main方法直接干 这里用了一个线程池实现多线程扫描

然后扫描的时候是这个样子的

速度非常的快，不仅是扫描速度快，ueditor搭环境的时间可能需要一天，这东西写脚本加找到漏洞总共用了15分钟左右，大幅度提高了工作效率，在更短的时间内为公司做了更大的贡献。

其他无了，溜了溜了。