[信息收集]

所谓万事开头难。
对于我个人来言，在决定认认真真挖掘一家厂商漏洞时，刚开始最难的就是信息收集，这是耗费精力最大，占用时间最长的一个环节，而且它是一个持续性的过程。

SRC漏洞挖掘中需要收集的信息大概包括

1. 厂商域名和IP段
2. 厂商业务信息

因为这是安全测试行为，并不是真正的攻防对抗行为，因此某些信息收集并不需要面面俱到。

[子域名收集]

首先来说厂商域名收集，因为这里其实厂商会在在规则中划定测试业务及域名，这里我们对兄弟域名挖掘不做赘述，我们只说说子域名。

针对子域名，互联网上各种方法其实很多了
1.基于SSL证书查询
2.第三方网站接口查询
3.Github
4.DNS解析记录
5.子域名枚举等等

我为什么把基于SSL证书查询子域名放在第一位，这里其实是有我一个心路历程的。
在去年12月份，我才开始使用这个方法去收集子域名，但其实这个方法在很久之前就披露了，因为我之前挖掘网易SRC漏洞已经两年多了，我认为我收集的网易域名信息已经尽可能的足够完整了。但是用了这个方法后，才知道还是有很多东西没有收集到。所以当一种需求出现不同方法时，还是应该多去尝试，不能满足于现状。

基于SSL证书查询的有
1.censys.io
2.crt.sh

第三方接口查询网站有
1.riskiq
2.shodan
3.findsubdomains

4.censys.io

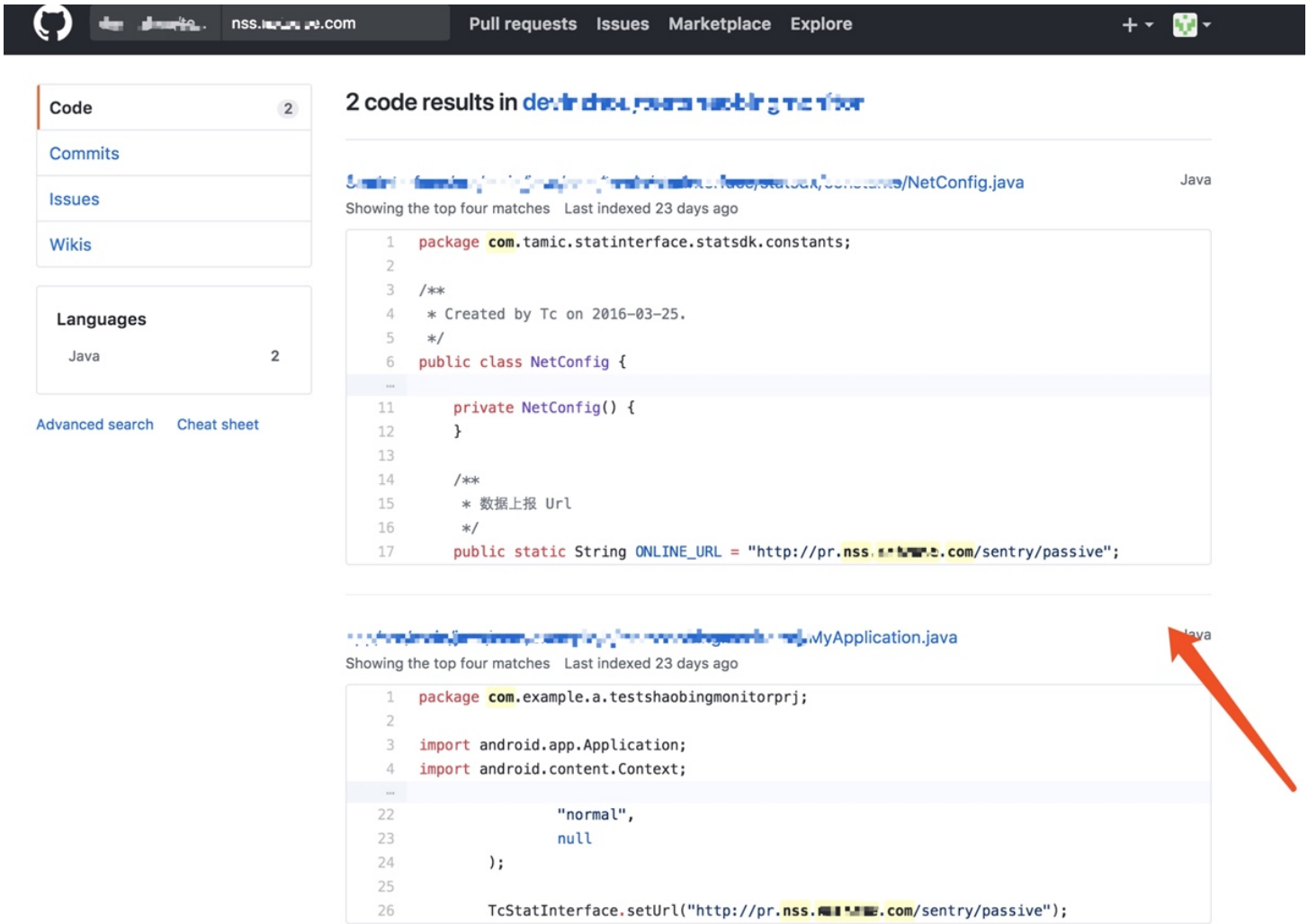
5.dnsdb.io

这里DNS解析记录和子域名枚举我会放到下面字典环节说到。

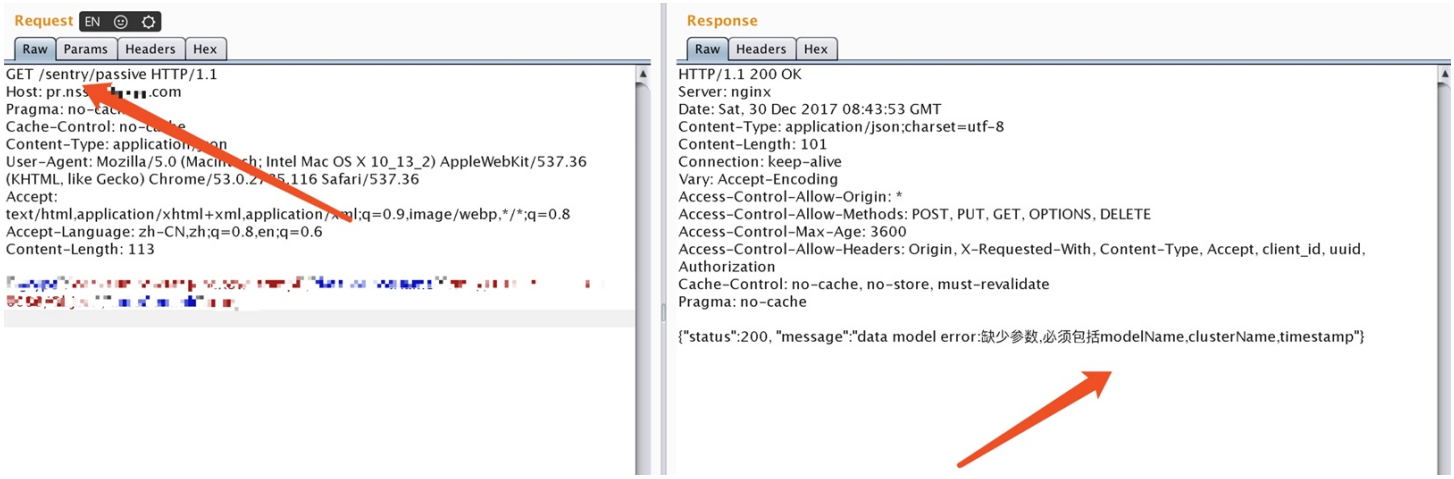
这里分享一个案例，我利用基于SSL证书查询到一个二级域名并在Github发现这个二级域名相关的三级域名测试代码，拿到生产网权限。

通过crt.sh查询发现了二级域名nss.a.com，这里其实我也使用了子域名枚举工具，但没发现什么可以利用的东西。
这时候我把二级域名放到Github下搜索，发现有个三级域名的接口

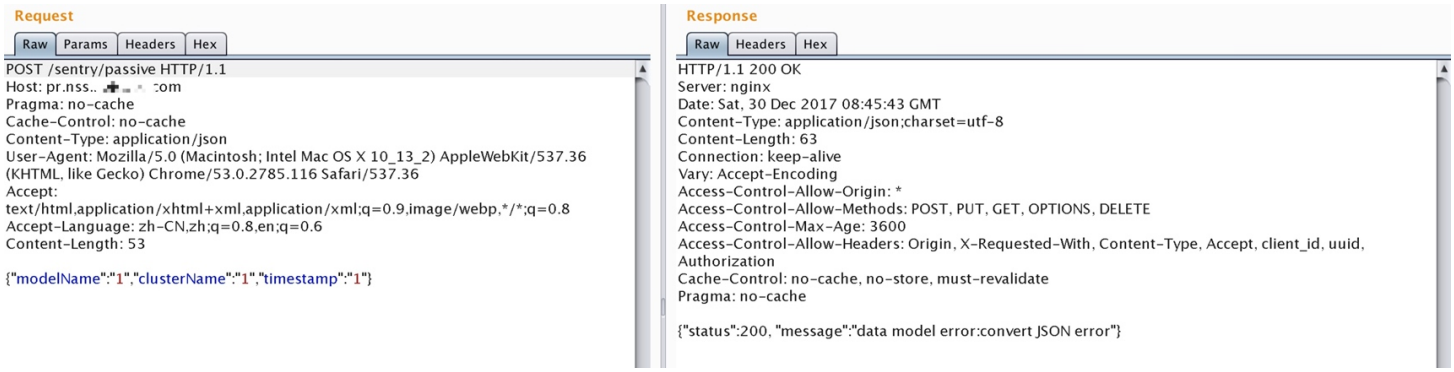
	2018-05-24	2018-05-24	2020-04-20	C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert SHA2 High Assurance Server CA
	2018-05-24	2018-01-02	2020-01-13	C=US, O=DigiCert Inc, OU=www.digicert.com, CN=GeoTrust RSA CA 2018
	2018-05-24	2017-09-07	2020-09-06	C=US, O=GeoTrust Inc., CN=GeoTrust SSL CA - G3
	2018-05-23	2017-01-13	2020-01-13	C=US, O=GeoTrust Inc., CN=GeoTrust SSL CA - G3
	2018-05-23	2018-05-23	2019-05-23	C=CN, O="TrustAsia Technologies, Inc.", OU=Domain Validated SSL, CN=TrustAsia TLS RSA CA
	2018-05-23	2018-05-23	2020-07-21	C=US, O=DigiCert Inc, OU=www.digicert.com, CN=GeoTrust RSA CA 2018
	2018-05-23	2018-05-23	2018-08-21	C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3



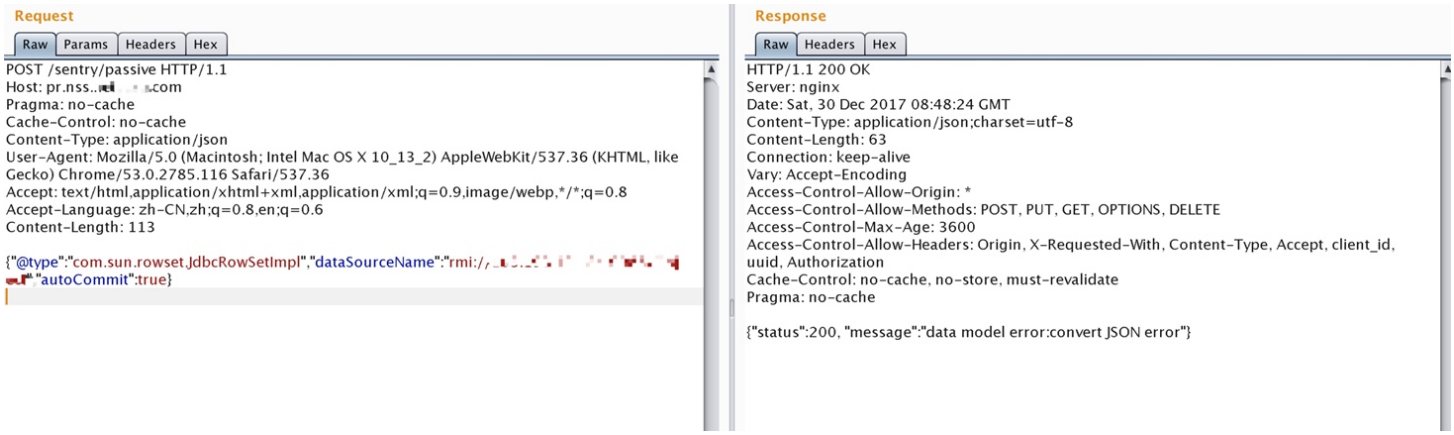
这里直接构造GET请求，发现提示缺少参数，参数名也提示了



我当时看了下代码，代码中要求请求头需要设置类型为json，所以正确的请求包如下



这时候我想到了fastjson反序列化漏洞，尝试下



成功进入生产内网

```
[redacted]-nss [redacted]:/home$ ifconfig
ifconfig
eth0      Link encap:Ethernet  HWaddr [redacted]
          inet addr:10.171.160.21  Bcast:10.171.175.255  Mask:255.255.240.0
          inet6 addr: [redacted] Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1400  Metric:1
          RX packets:21218218 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26872632 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1145452959 (1.0 GiB)  TX bytes:1966455741 (1.8 GiB)

eth1      Link encap:Ethernet  HWaddr [redacted]
          inet addr:10.172.128.83  Bcast:10.172.128.127  Mask:255.255.255.192
          inet6 addr: [redacted]:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10694731078 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10743969286 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5897444964425 (5.3 TiB)  TX bytes:6526799397258 (5.9 TiB)

eth2      Link encap:Ethernet  HWaddr fa:[redacted]
          inet addr:[redacted]  Bcast:[redacted]  Mask:255.255.252.0
          inet6 addr: [redacted] Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1101271796 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4172114 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

这里单独提出Github寻找子域名就是因为这个原因，有时候大范围的搜索子域名效果并不明显，我们需要更加针对性的搜索子域名再寻找目标，对了，github在登录状态下搜索返回的结果会更多。

[IP段收集]

说完子域名，接下来是说厂商IP相关的
在枚举子域名时，一般我们都能获取大概的厂商所处的IP段了，这时候我们可以通过AS号或者IP所属网络名称进行反查，获取厂商拥有的IP段。
如:我们在中国互联网信息中心的网站上 `http://IPwhois.cnnic.net.cn/` 查询123.58.191.1。


中国互联网络信息中心
China Internet Network Information Center
欢迎来自104.245.14.87的用户使用CNNIC IP地址注册信息查询系统
您可以查到CNNIC及中国大陆的IP地址分配信息
IPv4地址查询
请输入IPv4地址(如210.72.0.0 或 210.72.0.0/19 或 210.72.0.0 – 210.72.31.255)，查询IPv4地址的相关信息
123.58.191.1
联系人信息查询
请输入联系人账号(如IPAS1-CN)，查询联系人相关信息
IPv6地址查询
请输入IPv6地址(如2001:F88::或 2001:F88::/32)，查询IPv6地址相关信息
AS号码查询
请输入AS号码(如9811或AS9811)，查询AS号码相关信息
路由注册查询
请输入地址(如211.144.211.0/24或121.58.144.0/20)，查询路由注册相关信息
网络名称查询
请输入网络名称(如CSTNET)，查询网络名称相关信息
版权所有 中国互联网络信息中心 保留所有权利

这个是属于网易的IP。

IPv4地址段:	123.58.160.0 – 123.58.191.255
网络名称:	Netease-Network
单位描述:	Guangzhou NetEase Computer System Co., Ltd.
单位描述:	NetEase Building No.16 Ke Yun Road, Zhong Shan Avenue,
单位描述:	Guangzhou, P.R. China
国家代码:	CN
管理联系人:	AUTO1-FW
技术联系人:	AUTO1-FW
维护账号:	MAINT-AP-CNISP
事件响应账号:	IRT-CNISP-CN
地址状态:	ALLOCATED NON-PORTABLE
最后修改记录:	2015-09-08T01:35:27Z
数据来源:	APNIC
姓名:	Fengxian Wen
联系人代码:	AUTO1-FW
邮件地址:	fxwen@corp.netease.com
通讯地址:	Guangzhou NetEase Computer System Co.,Ltd
办公电话:	+86-20-85106115
投诉邮箱:	sa@corp.netease.com
国家代码:	CN
维护账号:	MAINT-AP-CNISP
最后修改记录:	2015-09-08T01:30:14Z
数据来源:	APNIC

我们查询发现他所属的网络名称为Netease-Network，我这里使用这个网络名称继续根据网络名称查询。

发现有这么多IP都是属于Netease-Network的。

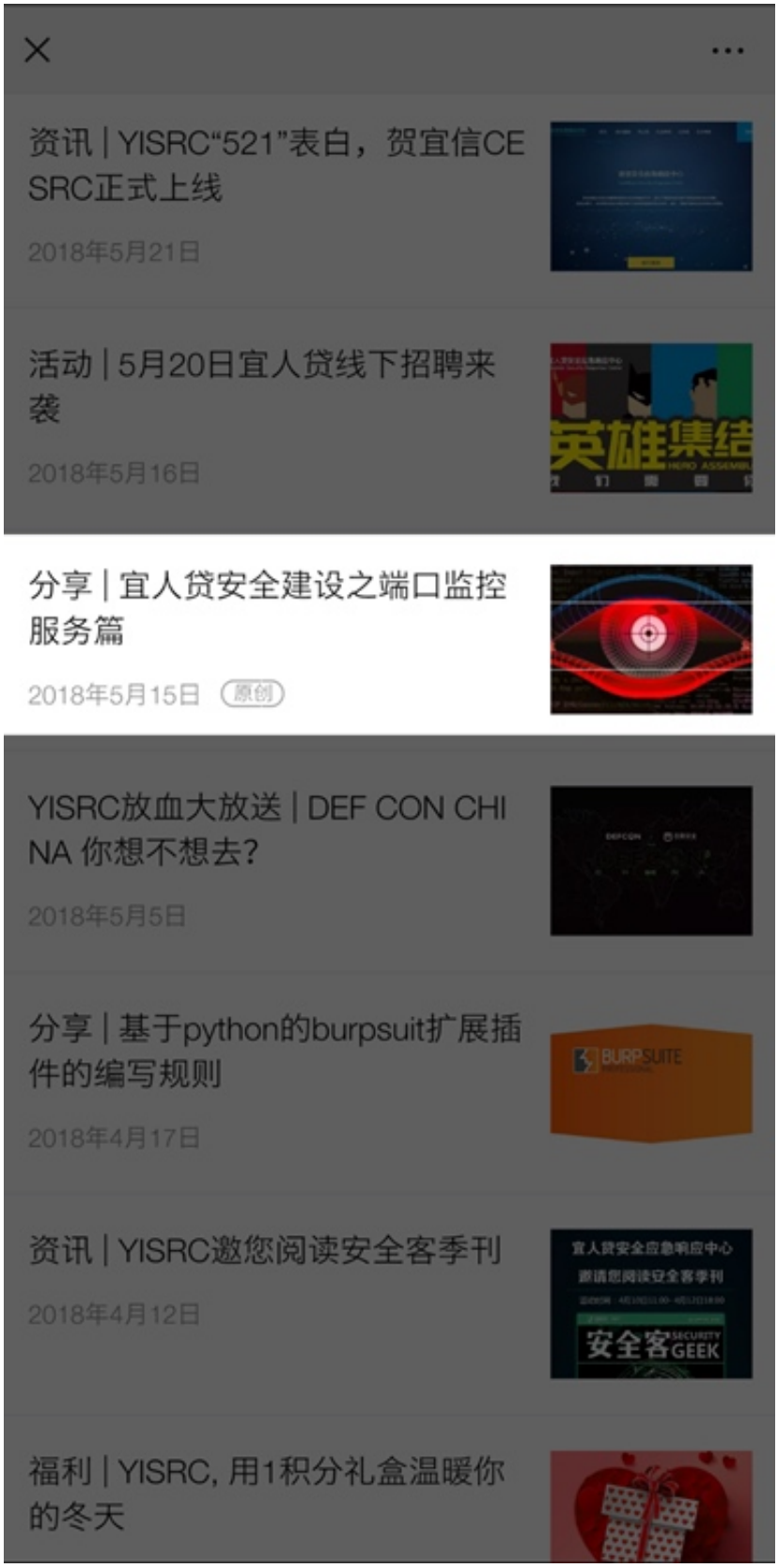
IPv4地址段:	42.186.0.0 – 42.186.255.255
网络名称:	Netease-Network
单位描述:	Guangzhou NetEase Computer System Co., Ltd
管理联系人:	ZX3316-AP
技术联系人:	ZX3316-AP
国家代码:	CN
地址状态:	ALLOCATED PORTABLE
维护账号:	MAINT-CNNIC-AP
次级维护帐号:	MAINT-CNNIC-AP
事件响应账号:	IRT-CNNIC-CN
路由维护帐号:	MAINT-CNNIC-AP
最后修改记录:	2016-06-20T05:52:01Z
数据来源:	APNIC
IPv4地址段:	123.58.160.0 – 123.58.191.255
网络名称:	Netease-Network
单位描述:	Guangzhou NetEase Computer System Co., Ltd.
单位描述:	NetEase Building No.16 Ke Yun Road, Zhong Shan Avenue,
单位描述:	Guangzhou, P.R. China
国家代码:	CN
管理联系人:	AUTO1-FW
技术联系人:	AUTO1-FW
维护账号:	MAINT-AP-CNISP
事件响应账号:	IRT-CNISP-CN
地址状态:	ALLOCATED NON-PORTABLE
最后修改记录:	2015-09-08T01:35:27Z
数据来源:	APNIC
IPv4地址段:	114.113.216.0 – 114.113.219.255

像网易这样体量较大的公司的网络名称也是比较多的，并不一定就只有一个网络名称，例如我们直接使用netease去查询IP，查询后所得的这些IP也是网易的。强调一点，这里网络名称的查询并不支持模糊查询。目前来说这是我丰富厂商IP段的一个比较好的方法。

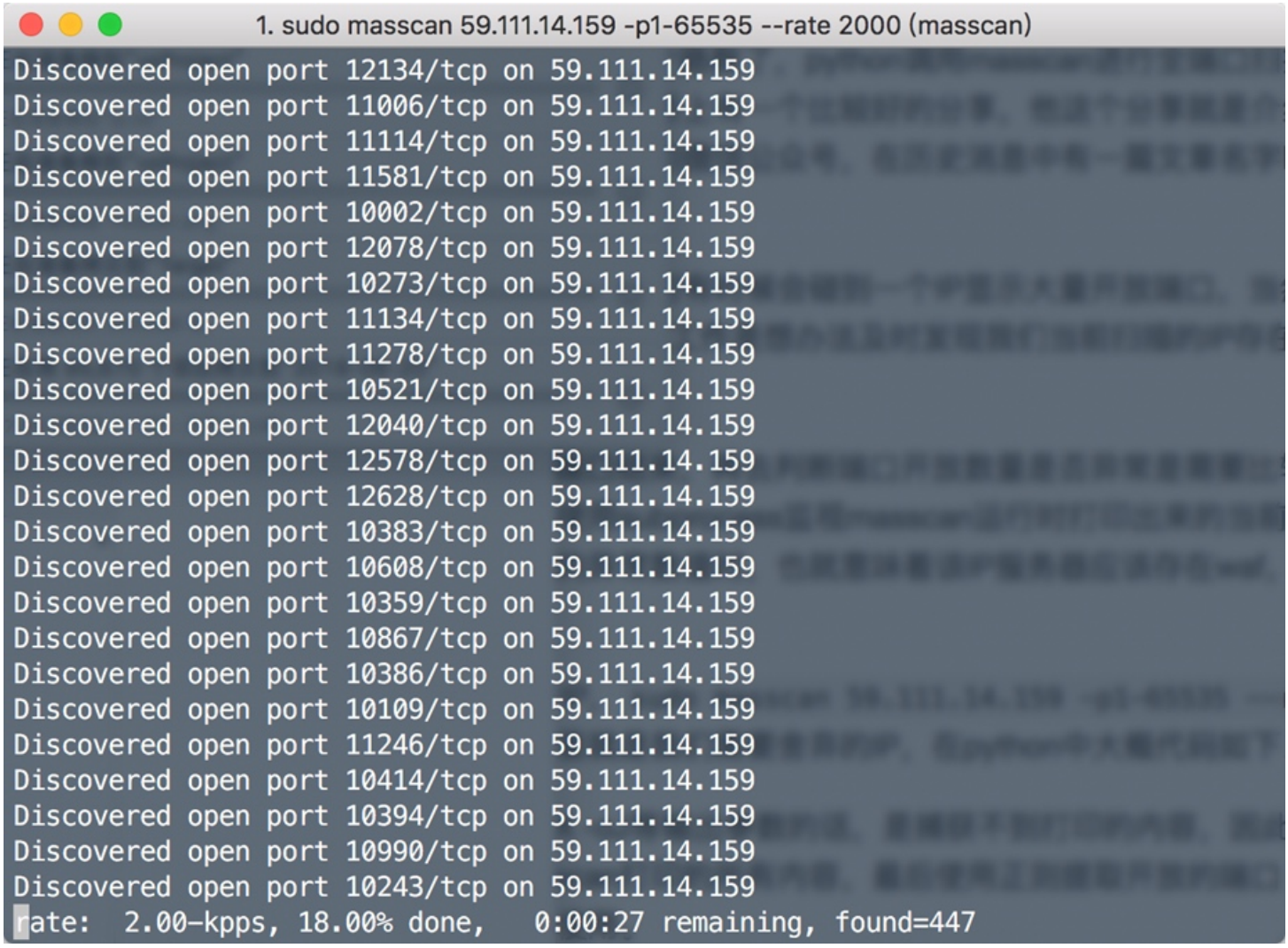
[端口扫描]

那么我们现在拿到了域名和IP段可以做些什么?
面对如此数量巨大的域名和IP段，扫描器肯定是少不了了。
所谓八仙过海各显神通，扫描器设计架构上大家各有见解，我这里分享下其他的细节上的东西。

在面对大批量IP时，相信大家都查不多是一个路数了，python调用masscan进行全端口扫描+nmap端口服务识别。在宜人安全应急响应中心的微信公众号上有一个比较好的分享，他这个分享就是介绍这一组合的架构，大家可以关注下宜人安全应急响应中心的微信公众号，在历史消息中有一篇文章名字叫做<<宜人贷安全建设之端口监控服务篇>>。



那么在这里我遇到的问题是，masscan扫描时有时候会碰到一个IP显示大量开放端口，当然这些开放的端口都是虚假的，是waf在起作用。



我们需要做的工作是想办法及时发现我们当前扫描的IP存在waf，记录并跳过此IP，继续下一个IP的扫描。

如果我们等待masscan扫描这个IP全端口结束，再去判断端口开放数量是否异常是需要比较久的时间，这里我们可以设定首先一个异常数值，并使用subprocess监视masscan运行时打印出来的当前开放端口数，当监视到的当前开放端口数超过我们设置的异常数值时，也就意味着该IP服务器应该存在waf，直接break进入下一个循环。

这里演示这个 `59.111.14.159` 这个IP， `sudo masscan 59.111.14.159 -p1-65535 --rate 2000` ，我们可以发现开放端口数量异常，那么这里就是我们需要舍弃的IP，在python中大概代码如下。

```python
import re
import os
import sys
import click
import subprocess
import threading

limitNumber = 80

lock = threading.Lock()

command = 'masscan 59.111.14.159 -p1-65535 --rate 2000'
child = subprocess.Popen(command,stdout=subprocess.PIPE,stderr=subprocess.STDOUT,shell=True)
while child.poll() is None:
    output = child.stdout.readline()
    line = str(output, encoding='utf-8').strip()
    if 'found=' in line:
        lock.acquire()
        print(line)
        lock.release()
        foundNumber = re.findall(r'found=(\d{1,5})', line)
        if int(foundNumber[-1]) > int(limitNumber):
            os.kill(child.pid, 9)
            print('疑似有WAF!存活端口'+ str(foundNumber[-1]) +'个')
```

这里还有个坑，如果masscan加上-oX -oJ等输出参数的话，是捕获不到打印的内容，因此我这里没有使用输出参数保存扫描结果，而是保存masscan打印的所有内容，最后使用正则提取开放的端口。当然这里也可以通过修改masscan的源码，重新编译使用。

masscan结束后就到了nmap，在nmap中，我会使用 `-sV -sT -Pn --version-all --open` 参数
-sV 是用来识别端口服务的。
-sT 其实在速度上并不如半开放扫描-sS，因为他需要完成整个握手包的，而且使用-sT目标主机会记录大量日志，-sT唯一的优势就是使用这个参数不需要root权限，普通用户权限就可以使用，而-sS是需要root权限。
-Pn就是跳过发现主机过程，nmap在扫描之前会首先探测主机是否开放，发现方式就是ping，因为我们之前已经使用了masscan识别到了开放端口，也就意味着主机是存活的，我们也就没有必要再次去nmap去探测主机是否开放。
--open 是限定只探测状态为open的端口
--version-all 会对每个端口尝试每个探测报文，在测试时候经常会碰到端口服务识别不出来，可以尝试使用这个参数说不定可以解决，因为他会对每个端口进行每一种类型，保证最大准确率。

端口识别出来后，当然就是针对每一种开放服务进行相应的安全检测，这些大同小异。

[字典的收集与使用优化]

接下来，我想分享的是对我来说比较重要的一个东西，字典。对于字典其实我相信很多师傅都有自己的一套方法，但是在互联网上分享的字典或者对应的字典工具的确有点藏着掖着的感觉。

使自己的字典更强大是我这几年一直在做的事情，作为一个没有太多资源的安全爱好者，我能做的就是去收集能收集到的一切信息并做整合优化。

在上面我只是提到了枚举子域名，没有多说，因为我想把它放到这一环节来提一下。
枚举子域名是收集信息中重要的一个环节，这个环节是否能有所收获，就在于字典是否强大。
接下来我将先说下域名类字典和站点类字典的收集，说完后，我才会再说关于字典的使用与优化。

[字典的获取]

用之于民，取之于民，我反过来说是因为，例如我们想拿字典来枚举子域名，那我们最好的字典数据源是什么，就是域名。

先说下子域名的字典从何而来
1.这里我一般是收集各个子域名枚举工具的已有的字典并去重,这个大家都知道
2.就是个体力活了，在rapid7的公开数据中有fdns和rnds的json数据

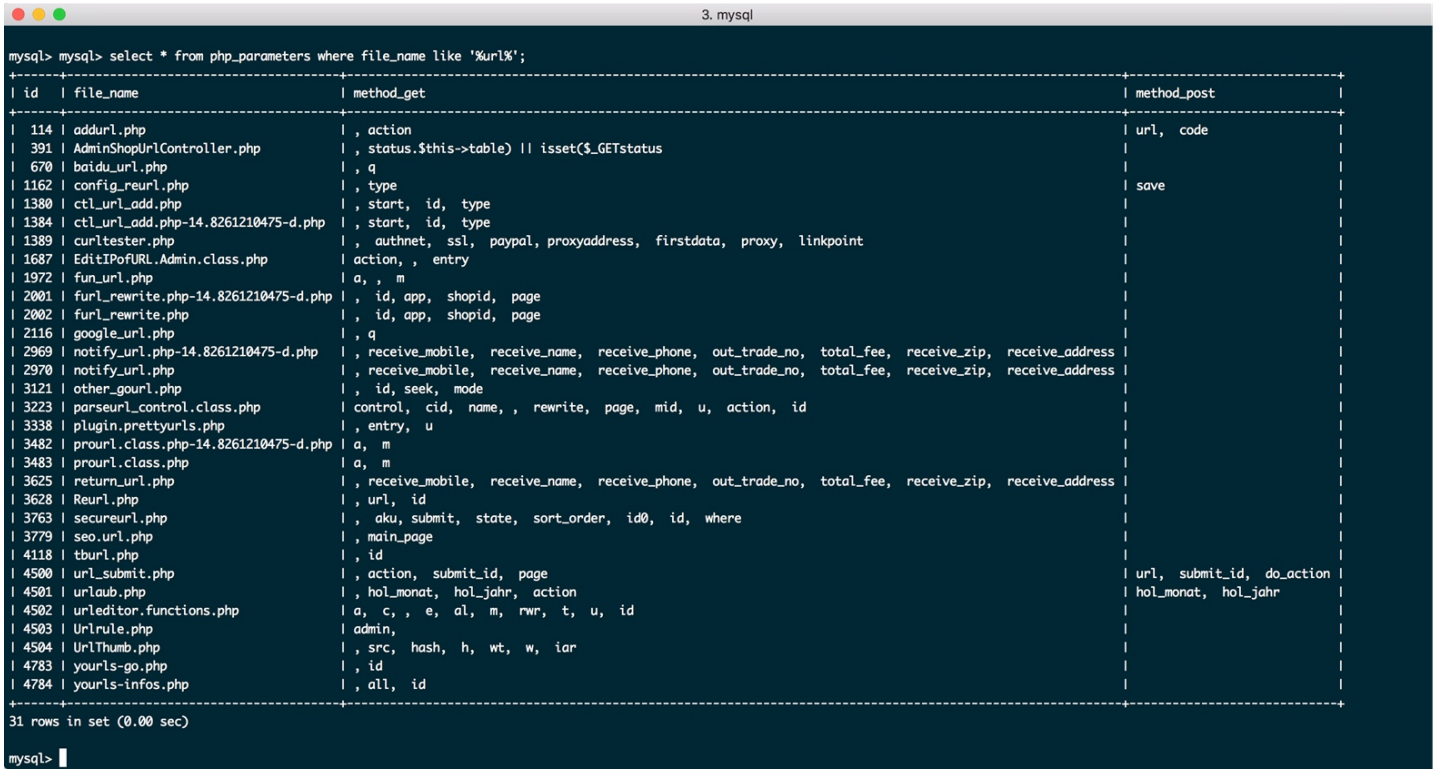
```
https://opendata.rapid7.com/sonar.rdns_v2/
https://opendata.rapid7.com/sonar.fdns_v2/
```

我们可以提取里面的子域名作为字典，当然这里面的数据是相当大的，其中脏数据的剔除也是需要做大量工作的，所以说是个体力活。

接着是站点类字典的收集，站点类的字典首先我是分了这样四个小类。

```
1.目录类
2.可执行脚本类
3.参数类
4.静态资源类(js脚本名)
```

在2016年我做了这样一个事情，我下载了1000多个网络上开源的web源码，通过正则提取这些源码的目录结构，可执行脚本文件名，参数名，请求方式，静态资源名(主要js脚本名)。



收集目录结构是为了爆破目录，收集可执行脚本文件名是为了爆破一些爬虫获取不到的可执行脚本文件，参数名当然也是为了爆破可执行脚本的参数，请求方式后来我反应过来没有太大必要收集，因为使用post请求，就可以囊括get请求了，这里收集js脚本名是因为其实我碰到很多站都会把一些重要的api存放在js文件中。

上面这种针对性的收集，肯定会减少爆破时遗漏的情况。

[通过字典突破的案例]

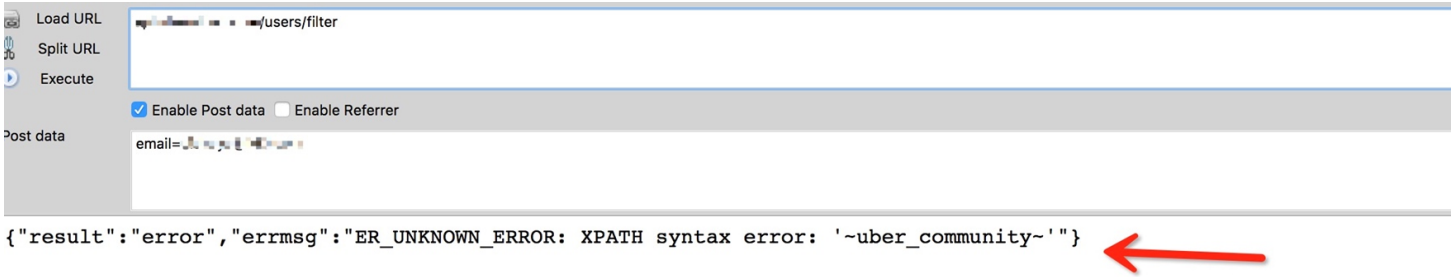
在2016年，我通过枚举Uber某站api和参数发现了一个二次注入。

当时，我的朋友@cy 在hackerone上提交了1个Uber的SQL注入漏洞，了解了下大概内容，由于该站点把一些api及其参数方法写入了js中，可以通过分析js构造数据包进行请求，其中1处api存在sql注入，提交给Uber后获得相应奖金。

当时存在漏洞的api早已修复，但是我凭一种直觉，当时我觉得应该还是有漏网之鱼的。通过爆破api路径，成功获得2个未写入前台js的的未授权后台api，然后根据js里已有的参数名，最终构造爆破出了8个参数。最终成功获取到一个未授权的数据写入api和一个未授权的数据查询api。

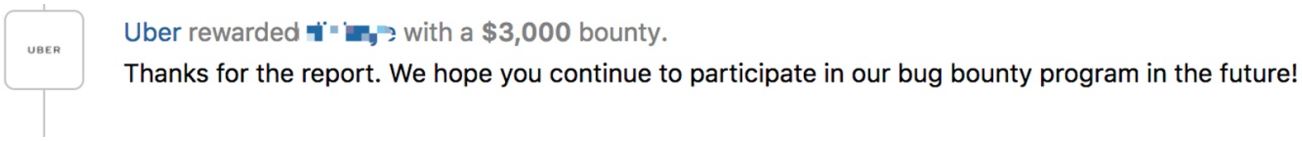
{"result":"success","data":{"openid":"test123456')and updatexml(1,concat(0x7e,(SELECT database()),0x7e),1)#","phone":"
","email":"
","firstName":"
","lastName":"
","picture":"http://
","promoCode":"
","createTime":"2016-06-29T15:00:44.783Z","createBy":"anonymous","id":464765}}

我在openid这个参数里面添加了注入语句，可以看到返回正常，接着我再通过查询api，查询刚刚添加的用户，形成二次注入




```
{"result":"error","errmsg":"ER_UNKNOWN_ERROR: XPATH syntax error: '~uber_community~'"}
```

最终获得3000$的奖励

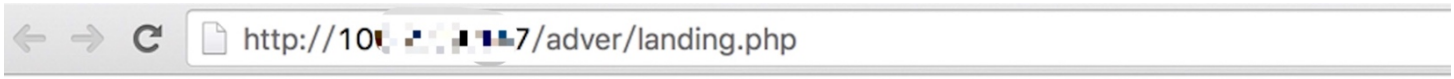


Uber这个案例虽然路由参数并不复杂，但的确让我开始注重字典的精准收集与合理利用，这也是我分类收集字典的初始原因。

这里还有另外一个案例，当我们碰到站点首页403或者404时，一般都会放弃看别的，但其实这些站点一定程度上往往更容易出现问题。

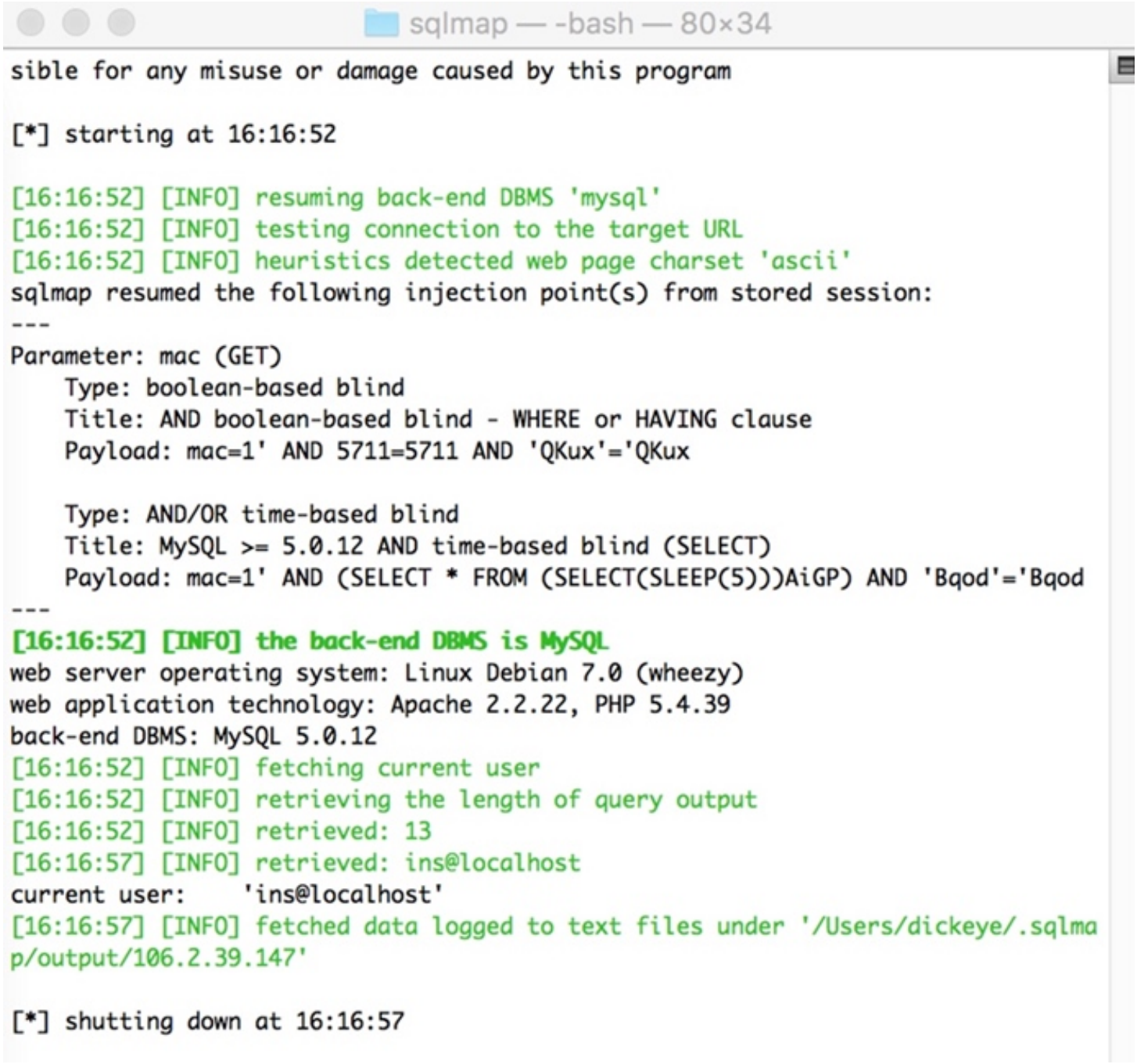
之前我遇到一个IP `106.**.**.147` 它的首页是403页面

这里我通过目录爆破和可执行脚本文件爆破，获取到
`http://106.**.**.147/adver/landing.php` 这个路径，访问提示参数错误



{"msg":"params error","status":0}

那么很明显我需要一个参数，通过爆破参数，成功枚举到参数，完整URL：`http://106.**.**.147/adver/landing.php?mac=1` 添加单引号，成功发现SQL注入，获得数据库权限，而且这是个游戏业务数据库。



通过这种暴力测试我在这几年获得了相当可观的奖金，因为这是SRC的安全测试，因此这种动静较大，留存日志较多的测试我也不需要担心什么问题，但是在真正的攻防对抗时还是不建议大动作。

[字典的使用与优化]

字典获取解决了，但问题也来了，字典当然是越全越好，越多越好，但是也要考虑效率问题，如此庞大的字典怎么才能更高效的使用?

我们可以将字典入库，增加dict_count字段用于计数，当我们的关键词命中时，对应的dict_count值加1，每次使用字典时，从数据库中order by dict_count desc提取关键词，生成一个根据关键词命中次数降序的字典，这样经常命中的关键词就会靠前，我们使用的字典的效率也会提高，循环往复我们的字典将会越加成熟。

[业务安全]

业务安全没有可以公开说的案例，因此只是说一些技巧。
当你决定认真挖掘一家厂商时，要记住业务安全是国内SRC最看重的。业务安全中最容易出现问题的有两个环节：
1.非普通用户拥有的权限，如：商家，合作方
2.新上线业务

针对非普通用户环节，我们需要想尽办法去获得商家，合作方，签约作者的权限，以便进行测试，这里甚至可以用上一些简单的社工方法。
新上线业务，可能有些师傅觉得不太会出现问题，但从我自己经历来看，出现问题还蛮多的，所以我们需要关注各个业务的最新动态，以便第一时间测试，例如关注微信公众号。

[APP测试]

业务离不开APP，不管是ios平台还是Android平台都有存在APP使用证书锁定的技术的情况，所谓证书锁定就是APP将目标服务器的SSL证书放到APP里面，这个证书被用于会话配置。常规方法将无法抓到这个APP的数据包。
在ios下我们可以使用越狱iPhone，安装SSL Pining Disable 插件实现禁止SSL Pining ，详情可见我的博客 `http://pwn.dog/index.php/ios/ios-disable-ssl-pinning.html`

在安卓下可以使用瘦蛟舞大神的安卓证书锁定解除工具
`https://github.com/WooyunDota/DroidSSLUnpinning`