

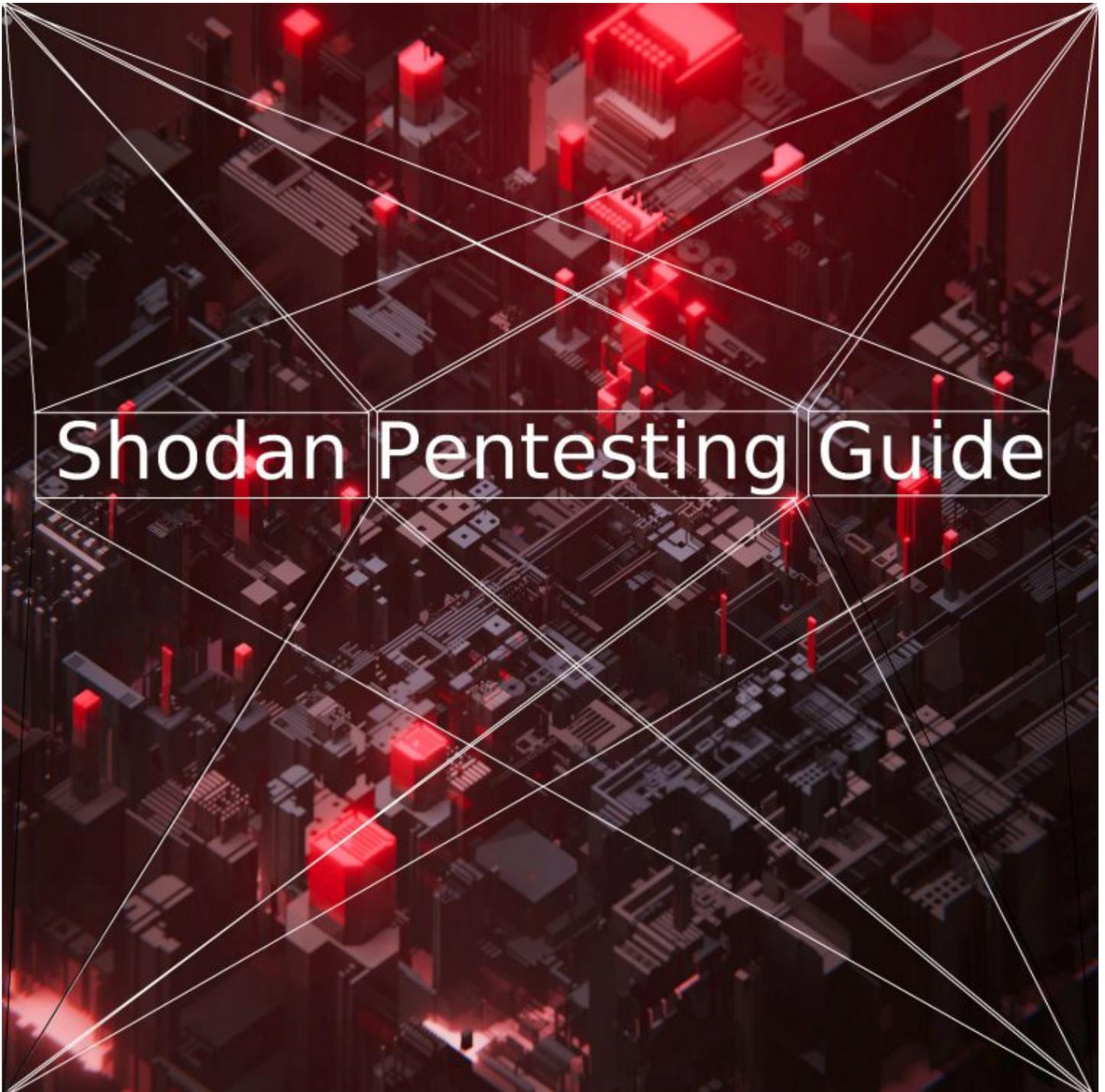


Community Homepage

PENTESTING

Shodan Pentesting Guide

Delving deep into Shodan's mine



Shodan is a tool for searching devices connected to the internet. Unlike search engines which help you find websites, Shodan helps you find information about desktops, servers, IoT devices, and more. This information includes metadata such as the software running on each device.

Common uses of Shodan include Network Security, Market Research, Cyber Risk, scanning IoT devices, and Tracking Ransomware. This guide will focus on comprehensively covering these applications in a pentesting context.

Table of Contents [[show](#)]

What is Shodan?

Shodan is a search engine for Internet-connected devices. It was created by John C. Matherly (@achilleian) in 2009.

Shodan is a tool that lets you explore the internet; discovering connected devices or network services, monitoring network security, making global statistics and so on.

The Shodan's website/database references results from extensive port scanning of the Internet.

Shodan interfaces

This section will show you the various ways you can connect to Shodan.

It's possible to interact with Shodan via the well known [website](#), the official python command-line interface tool and library, a variety of community driven libraries for many languages and also the official REST API.

CLI tool

The official shodan command-line interface ([CLI](#)) is written in python, for quick usage in your terminal.

Install

In a virtual python environment like [pyenv](#):

```
$ easy_install shodan
```

On [BlackArch](#) you can also install the following package:

```
# pacman -S python-shodan
```

Once you have installed shodan CLI tool, to setup your API token just do:

```
$ shodan init <YOUR_API_KEY>
```

Command overview

A dozen of straightforward commands are available:

```
-help
```

```
$ shodan -help
```

```
Usage: shodan [OPTIONS] COMMAND [ARGS]...
```

Options:

```
-h, -help  show this message and exit.
```

Commands:

```
alert      Manage the network alerts for your account.
convert    Convert the given input data file into a different
format.
count      Returns the number of results for a search.
data       Bulk data access to Shodan.
domain     View all available information for a domain.
download   Download search results and save them in a compressed
JSON file.
honeyscore Check whether the IP is a honeypot or not.
host       View all available information for an IP address.
info       Shows general information about your account.
init       Initialize the Shodan command-line.
myip       Print your external IP address.
org        Manage your organization's access to Shodan.
parse      Extract information out of compressed JSON files.
radar      Real-Time Map of some results as Shodan finds them.
scan       Scan an IP/ netblock using Shodan.
search     Search the Shodan database.
stats      Provide summary information about a search query.
stream     Stream data in real-time.
version    Print version of this tool.
```

info

If you have setup your API token, you can check the number of credits you have left:

```
$ shodan info
```

```
Query credits available: 100
```

```
Scan credits available: 100
```

Query credits are used to search Shodan and scan credits are used to scan IPs.

A search request consumes 1 query credit and scanning 1 IP consumes 1 scan credit.

```
version
```

When writing this article I was using shdoan 1.21.2:

```
$ shodan version
```

```
1.21.2
```

```
count
```

Returns the number of results for a search query.

```
$ shodan count openssh
```

```
23128
```

```
$ shodan count openssh 7
```

```
219
```

```
download
```

Search Shodan and download the results into a file where each line is a JSON [banner](#).

By default it will only download 1,000 results, if you want to download more look at the `-limit` flag.

The download command lets you save the results and process them afterwards using the parse command.

So if you often search for the same queries it will help you save credits.

The export credits are used to download data from the website at the rate of: 1 export credit lets you download up to 10,000 results. They are single-use which means that once you use them they don't automatically renew at the start of the month.

But if you don't have export credits, you can use 1 query credit to save 100 results.

```
$ shodan download -h
```

```
Usage: shodan download [OPTIONS] <filename> <search query>
```

Download search results and save them in a compressed JSON file.

Options:

```
--limit INTEGER The number of results you want to download. -1 to  
download
```

```
all the data possible.
```

```
--skip INTEGER The number of results to skip when starting the  
download.
```

```
-h, --help Show this message and exit.
```

For example here I will download 1000 results of the query openssh:

```
$ shodan download openssh-data openssh
```

```
Search query: openssh
```

```
Total number of results: 23128
```

```
Query credits left: 100
```

```
Output file: openssh-data.json.gz
```

```
[#####-] 99% 00:00:00
```

```
Saved 1000 results into file openssh-data.json.gz
```

After the download you can check how many credits you have left:

```
$ shodan info
Query credits available: 95
Scan credits available: 100
```

```
host
```

See information about the host such as where it's located, what ports are open and which organization owns the IP.

```
$ shodan host 1.1.1.1
1.1.1.1
Hostnames: one.one.one.one
Country: Australia
Organization: Mountain View Communications
Updated: 2020-01-21T22:26:00.168041
Number of open ports: 3

Ports:
53/udp
80/tcp

443/tcp
|-- SSL Versions: -SSLv2, -SSLv3, TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

$ shodan host 138.201.81.199
138.201.81.199
Hostnames: apollo.archlinux.org
Country: Germany
Organization: Hetzner Online GmbH
Updated: 2020-01-21T03:02:11.476262
Number of open ports: 4
```

Ports:

```
22/tcp OpenSSH (8.1)
```

```
25/tcp Postfix smtpd
```

```
80/tcp nginx (1.16.1)
```

```
443/tcp nginx (1.16.1)
```

```
|-- SSL Versions: -SSLv2, -SSLv3, -TLSv1, -TLSv1.1, TLSv1.2,  
TLSv1.3
```



```
myip
```

Returns your Internet-facing IP address.

```
$ shodan myip
```

```
199.30.49.210
```

```
parse
```

Use parse to analyze a file that was generated using the download command.

It lets you filter out the fields that you're interested in, convert the JSON to a CSV and is friendly for pipe-ing to other scripts.

```
$ shodan parse -h
```

```
Usage: shodan parse [OPTIONS] <filenames>
```

Extract information out of compressed JSON files.

Options:

```
--color / --no-color
```

```
--fields TEXT List of properties to output.
```

```
-f, --filters TEXT Filter the results for specific values using
```

key:value

pairs.

-o, --filename TEXT Save the filtered results in the given file (append if file exists).

--separator TEXT The separator between the properties of the search results.

-h, --help Show this message and exit.

The following command outputs filtered data for the previously downloaded openssh data:

```
$ shodan parse --fields location.country_code3,ip_str,hostnames -f port:2222 openssh-data.json.gz
```

```
ITA 89.107.109.247
```

```
HUN 193.6.173.187
```

```
FRA 77.87.111.110 pro-sip1.srv.proceau.net
```

```
USA 50.210.94.33
```

```
USA 35.130.36.118 035-130-036-118.biz.spectrum.com
```

```
AUT 80.120.19.180
```

```
JPN 124.155.95.212 v095212.ppp.asahi-net.or.jp
```

```
POL 83.144.70.114 83-144-70-114.static.chello.pl
```

```
BGR 84.238.200.8
```

```
AUT 80.120.19.168
```

```
USA 162.211.126.140
```

```
CAN 76.10.173.222 mail.nanoman.ca
```

```
USA 24.172.82.71 rrcs-24-172-82-71.midsouth.biz.rr.com
```

```
AUT 80.120.19.182
```

```
ITA 188.14.96.151 host151-96-static.14-188-
```

```
b.business.telecomitalia.it
```

```
USA 216.67.111.198 216-67-111-198.static.acsalaska.net
```

```
USA 73.179.238.221 c-73-179-238-221.hsd1.fl.comcast.net
```

```
HKG 113.28.18.59 113-28-18-59.static.imsbiz.com
```

```
$ shodan parse --fields
port,ip_str,location.city,location.postal_code -f
location.country_code:FR --separator , openssh-data.json.gz
22,188.92.65.5,Hésingue,68220
2222,77.87.111.110,,
22,51.89.105.163,,
22,5.135.218.249,,
22,93.177.70.142,,
2222,81.250.129.207,Paris,75116
22,51.255.85.97,,
22,193.52.218.40,Aix-en-provence,13090
22,51.77.112.86,,
22,149.202.19.41,,
22,5.39.117.104,,
22,195.154.53.223,Beaumont,95260
22,37.71.132.198,,
22,178.33.71.35,,
22,212.83.188.179,Jouy-le-moutier,95280
2222,195.200.166.216,Berre-l'etang,13130
22,82.251.157.165,Paris,75004
```

search

This command lets you search Shodan and view the results in a terminal-friendly way.

By default it will display the IP, port, hostnames and data. You can use the `--fields` parameter to print whichever banner fields you're interested in.

A simple query won't consume any credits but if you use a search filter or request page 2 and beyond, credits will be consumed.

```
$ shodan search -h
```

```
Usage: shodan search [OPTIONS] <search query>
```

Search the Shodan database

Options:

```
--color / --no-color
```

```
--fields TEXT List of properties to show in the search results.
```

```
--limit INTEGER The number of search results that should be returned.
```

Maximum: 1000

```
--separator TEXT The separator between the properties of the search results.
```

```
-h, --help Show this message and exit.
```

Example of query that won't cost credits:

```
$ shodan search --fields ip_str,port,os smb
```

```
156.226.167.81 445 windows Server 2008 R2 Datacenter 7601 Service Pack 1
```

```
156.243.104.194 445 windows Server 2008 R2 Enterprise 7601 Service Pack 1
```

```
91.230.243.89 445 windows 10 Pro 16299
```

```
85.3.170.18 445 windows 6.1
```

```
213.238.170.132 445 windows Server 2012 R2 Standard 9600
```

```
154.208.176.81 445 windows Server 2008 R2 Enterprise 7601 Service Pack 1
```

```
103.235.171.78 445 windows Server 2016 Datacenter 14393
```

```
102.130.40.85 445 windows Server 2016 Standard 14393
```

```
50.3.151.113 445 windows Server 2012 R2 Standard 9600
```

```
220.241.112.233 445 windows Server 2019 Standard 17763
```

```
100.27.15.229 445 wwindows Server 2012 R2 Standard 9600
```

```
212.71.136.11 445 Unix
156.255.174.225 445 windows Server 2008 R2 Datacenter 7601 Service
Pack 1
156.232.162.239 445 windows Server 2008 R2 Enterprise 7601
Service Pack 1
186.210.102.132 445 Unix
154.94.153.34 445 windows Server 2012 R2 Datacenter 9600
213.130.28.31 445 windows 6.1
```

Example of query that will cost 1 credit (because using a filter):

```
$ shodan search --fields ip_str,port,org,info product:mongodb
165.22.3.203 27017 Digital Ocean
213.159.208.76 27017 JSC The First
209.6.48.11 27017 RCN
23.239.0.110 27017 Linode
52.220.230.134 27017 Amazon.com
47.91.139.188 27017 Alibaba
159.203.169.196 27017 Digital Ocean
49.233.135.180 27017 Tencent cloud computing
122.228.113.75 27017 WENZHOU, ZHEJIANG Province, P.R.China.
106.14.42.66 27017 Hangzhou Alibaba Advertising Co.,Ltd.
59.108.91.3 27017 Beijing Founder Broadband Network Technology
Co.,L
115.29.176.18 27017 Hangzhou Alibaba Advertising Co.,Ltd.
148.251.46.75 27017 Hetzner Online GmbH
3.121.222.150 27017 Amazon.com
47.75.211.162 27017 Alibaba
200.219.217.122 27017 Equinix Brazil
```

scan

Scan an IP/ netblock using Shodan.

```
$ shodan scan -h
```

```
Usage: shodan scan [OPTIONS] COMMAND [ARGS]...
```

Scan an IP/ netblock using Shodan.

Options:

-h, --help Show this message and exit.

Commands:

internet Scan the Internet for a specific port and protocol using the...

list Show recently launched scans

protocols List the protocols that you can scan with using Shodan.

status Check the status of an on-demand scan.

submit Scan an IP/ netblock using Shodan.

Launching a scan will cost credits:

1 scan credit lets you scan 1 IP

By default a scan result will be displayed to *stdout* but you can save it to a file to be able to parse it later.

```
$ shodan scan submit --filename 104.27.154.244_scan.json.gz  
104.27.154.244
```

If the host has already been scanned in the last 24 hours, you won't be able to scan it again without an Enterprise grade plan.

```
$ shodan scan submit --filename 104.27.154.244_scan.json.gz  
104.27.154.244
```

Starting Shodan scan at 2020-01-22 23:46 - 100 scan credits left
No open ports found or the host has been recently crawled and cant get scanned again so soon.

You are also able to see the scans you previously launched with their ID and status:

```
$ shodan scan list
# 2 Scans Total - Showing 10 most recent scans:
# Scan ID Status Size Timestamp
zmWj3RNgIPbiQjx9 PROCESSING 1 2020-01-22T22:49:39.037000
8J9yu7jqTQ07AIiP PROCESSING 1 2020-01-22T22:46:34.790000
```

To save your scan results you are not forced to use `-filename`. You can simply launch a scan without saving it, and download the results later thanks to the scan ID:

```
$ shodan download --limit -1 scan-results.json.gz
scan:zmWj3RNgIPbiQjx9
```

As scan are done asynchronously, you can check the status of a scan at any moment.

```
$ shodan scan status zmWj3RNgIPbiQjx9
DONE
```

To see the scan ID when launching a scan you can use the verbose mode:

```
$ shodan scan submit --verbose 13.226.145.4
```

```
Starting Shodan scan at 2020-01-23 00:00 - 97 scan credits left
# Scan ID: 3z6Cqf1CCyVLtc6P
# Scan status: DONE
```

Customers with an Enterprise Data License will be allowed to request a scan of the entire Internet by simply specifying the port and protocol/module.

```
$ shodan scan internet 8080 wemo-http
```

Available protocols and modules can be listed with shodan scan protocols.

```
stats
```

Provide summary information about a search query

```
$ shodan stats -h
```

```
Usage: shodan stats [OPTIONS] <search query>
```

Provide summary information about a search query

Options:

--limit INTEGER The number of results to return.

--facets TEXT List of facets to get statistics for.

-O, --filename TEXT Save the results in a CSV file of the provided name.

-h, --help Show this message and exit.

It seems that by default you will get only top 10 and not for all facets:

```
$ shodan stats nginx
```

```
Top 10 Results for Facet: country
```

```
US 13,598,596
```

```
CN 6,013,993
```

```
ZA 3,067,296
```

```
DE 1,560,114
```

```
HK 1,065,990
```

```
RU 869,931
```

```
FR 859,715
```

GB 555,946

NL 550,591

JP 526,386

Top 10 Results for Facet: org

Amazon.com 1,897,943

CloudInnovation infrastructure 1,288,280

Leaseweb USA 1,200,146

EGIHosting 1,131,973

DXTL Tseung Kwan O Service 1,052,688

Hangzhou Alibaba Advertising Co.,Ltd. 770,553

Digital Ocean 749,221

Asline Limited 680,364

Power Line Datacenter 678,264

Quantil Networks 585,935

But we can customize this behavior:

```
$ shodan stats --facets domain,port,asn --limit 5 nginx
```

Top 5 Results for Facet: domain

amazonaws.com 2,208,958

scalabledns.com 435,980

googleusercontent.com 308,114

t-ipconnect.de 225,276

your-server.de 180,711

Top 5 Results for Facet: port

80 10,019,366

443 5,300,058

5000 588,809

5001 563,208

8080 453,604

Top 5 Results for Facet: asn

as37353 2,447,679

as35916 1,878,181

as15003 1,508,786

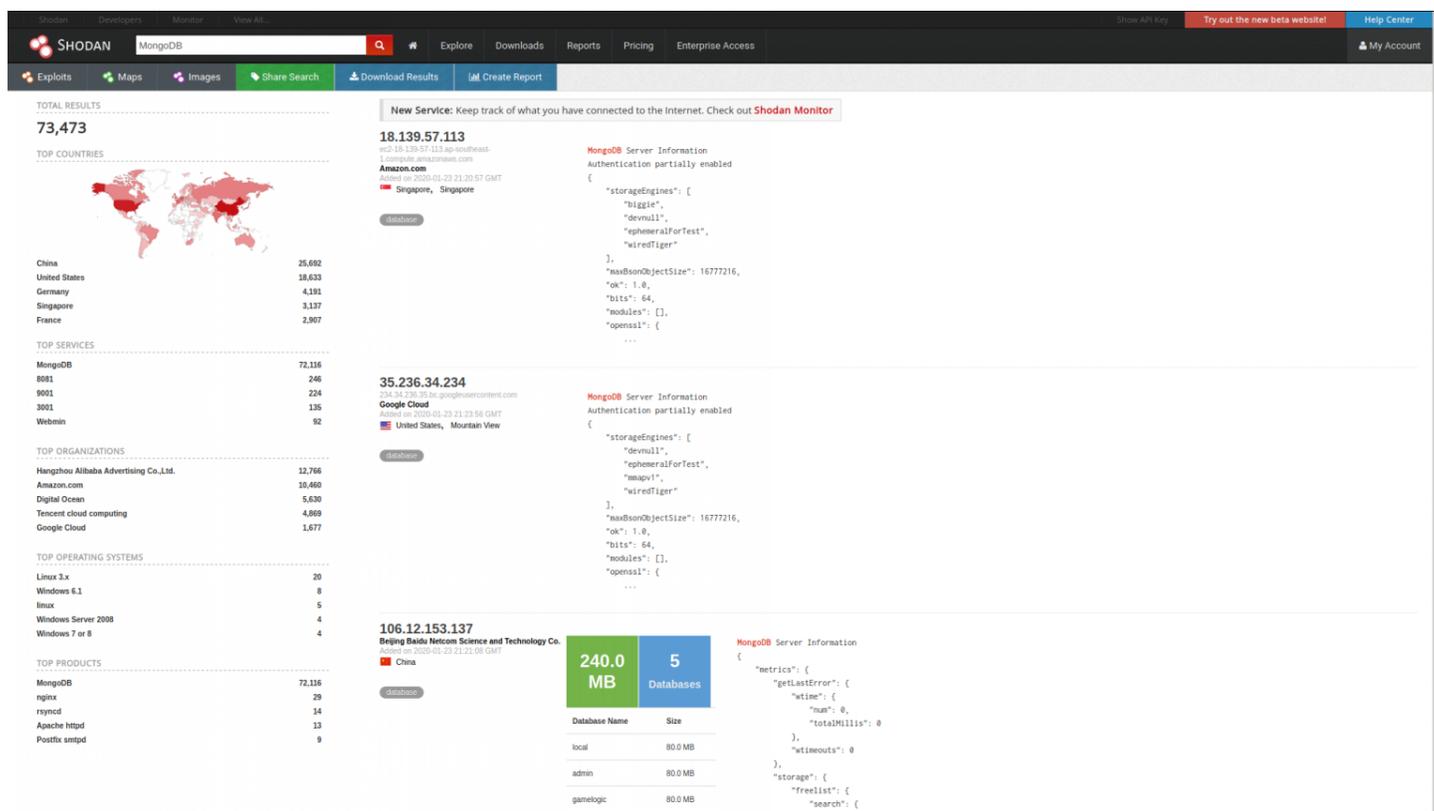
as16509 1,236,249

as18779 1,132,180

Website

Main interface

The main interface of Shodan is the [search engine](#).



It works like the search command of the CLI tool but with a fancy WebUI to display the results. It shows a summary for each host, the total count of hosts that matched the query like the count command of the CLI and some stats like the stats command.

18.139.57.113 ec2-18-139-57-113.ap-southeast-1.compute.amazonaws.com [View Raw Data](#)

Database

City	Singapore
Country	Singapore
Organization	Amazon.com
ISP	Amazon.com
Last Update	2020-01-23T21:20:57.090403
Hostnames	ec2-18-139-57-113.ap-southeast-1.compute.amazonaws.com
ASN	AS3

Web Technologies

- Bootstrap
- Google Font API
- jQuery
- Kibana
- Node.js

Vulnerabilities

Note: the device may not be impacted by all of these issues. The vulnerabilities are implied based on the software and version.

CVE-2019-0196 A vulnerability was found in Apache HTTP Server 2.4.17 to 2.4.38. Using fuzzed network input, the http/2 request handling could be made to access freed memory in string comparison when determining the method of a request and thus process the request incorrectly.

CVE-2019-0197 A vulnerability was found in Apache HTTP Server 2.4.34 to 2.4.38. When HTTP/2 was enabled for a http: host or H2Upgrade was enabled for h2 on a https: host, an Upgrade request from http/1.1 to http/2 that was not the first request on a connection could lead to a misconfiguration and crash. Errors that occur enable the h2 protocol or that subscribed to further connections.

Ports

80 443 3000 3001 3002 3310 5601 8000 9000 9090 9091 27017

Services

80
http
httpd

Apache httpd Version: 2.4.29
 HTTP/1.1 200 OK
 Date: Mon, 20 Jan 2020 05:17:28 GMT
 Server: Apache/2.4.29 (Ubuntu)
 Last-Modified: Tue, 11 Jun 2019 11:42:43 GMT
 ETag: "2aa6-58b0acb80150e"
 Accept-Ranges: bytes
 Content-Length: 10918
 Vary: Accept-Encoding
 Content-Type: text/html

443
https
httpsd

Apache httpd Version: 2.4.29
 HTTP/1.1 200 OK
 Date: Sun, 19 Jan 2020 20:05:08 GMT
 Server: Apache/2.4.29 (Ubuntu)
 Last-Modified: Fri, 17 Jan 2020 11:43:11 GMT
 ETag: "c93-59c5472301f70"
 Accept-Ranges: bytes
 Content-Length: 3219
 Vary: Accept-Encoding
 Content-Type: text/html

Once you have selected a host, you will be able to see a host specification table, vulnerabilities impacting the host, open ports and banners for open ports.

Downloading data

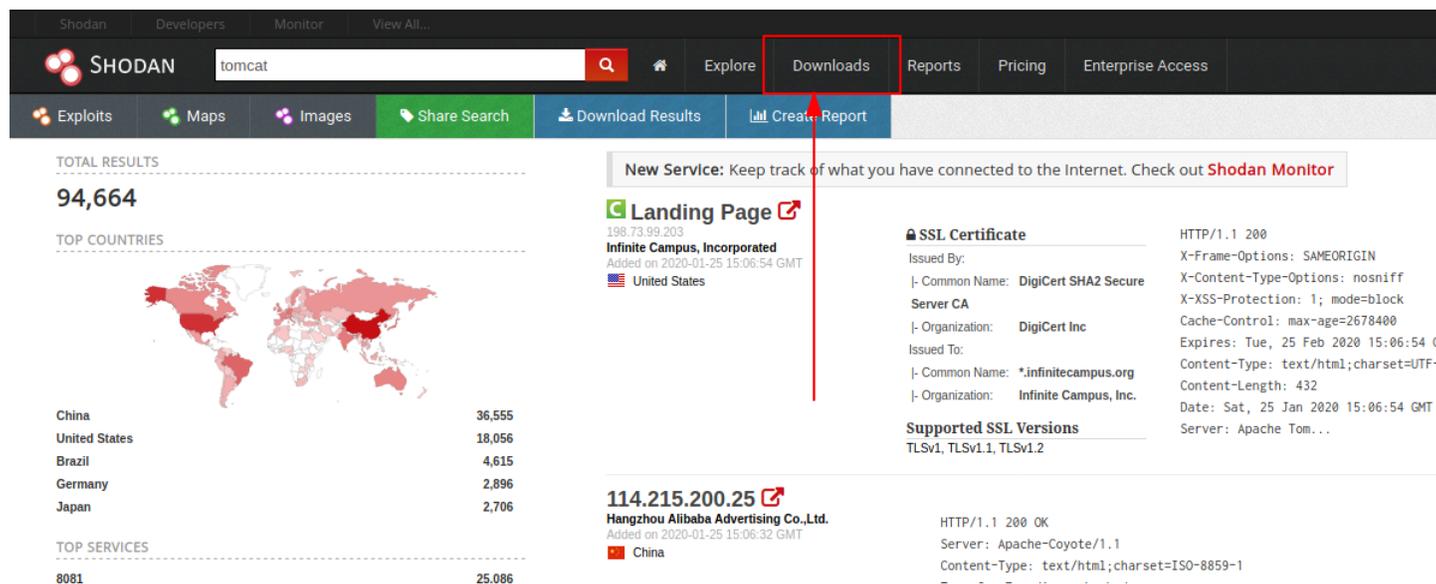
After you made a search, a *Download Results* button will be available:

Then you will be able to download the search results in JSON, CSV or XML.

Only the JSON format will contain the full data and be compatible with the Shodan CLI tool. CSV format will only contain IP, port, banner, organization and hostnames.

The XML format is deprecated by Shodan and consumes more space than the JSON one.

You can then view your download history in the [Downloads](#) section.

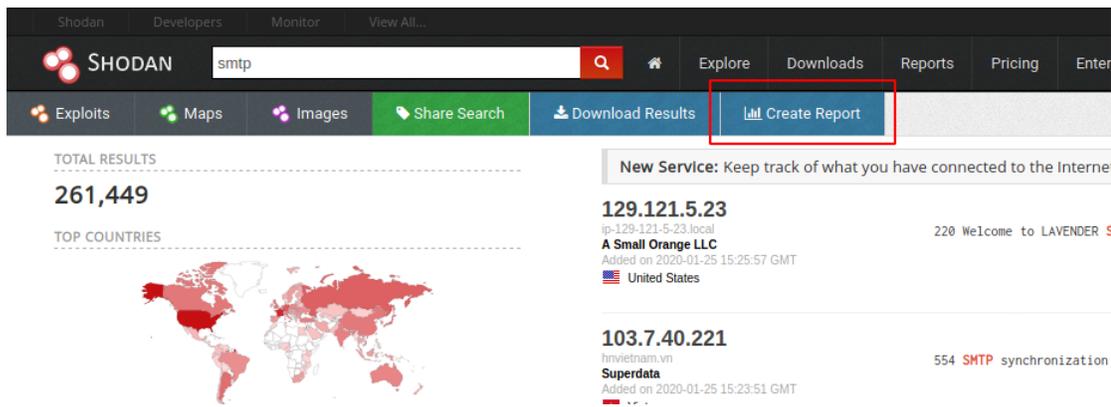


Generating reports

The website lets you generate a report based off of a search query.

The report contains graphs/ charts providing you a big picture view of how the results are distributed across the Internet. This feature is free and available to anyone.

To generate a report, click on the Create Report button from the search results page:



Shodan Developers Monitor View All...

SHODAN smtp

Explore Downloads Reports Pricing Enter

Exploits Maps Images Share Search Download Results **Create Report**

TOTAL RESULTS
261,449

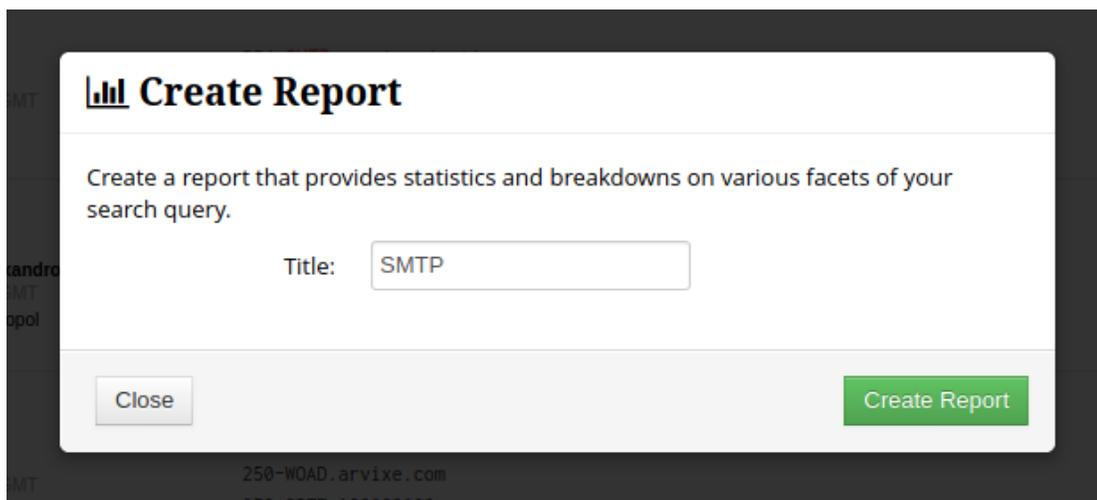
TOP COUNTRIES

New Service: Keep track of what you have connected to the Internet

129.121.5.23
ip-129-121-5-23.local
A Small Orange LLC
Added on 2020-01-25 15:25:57 GMT
United States

103.7.40.221
hsvietnam.vn
Superdata
Added on 2020-01-25 15:23:51 GMT

Name your report:



Create Report

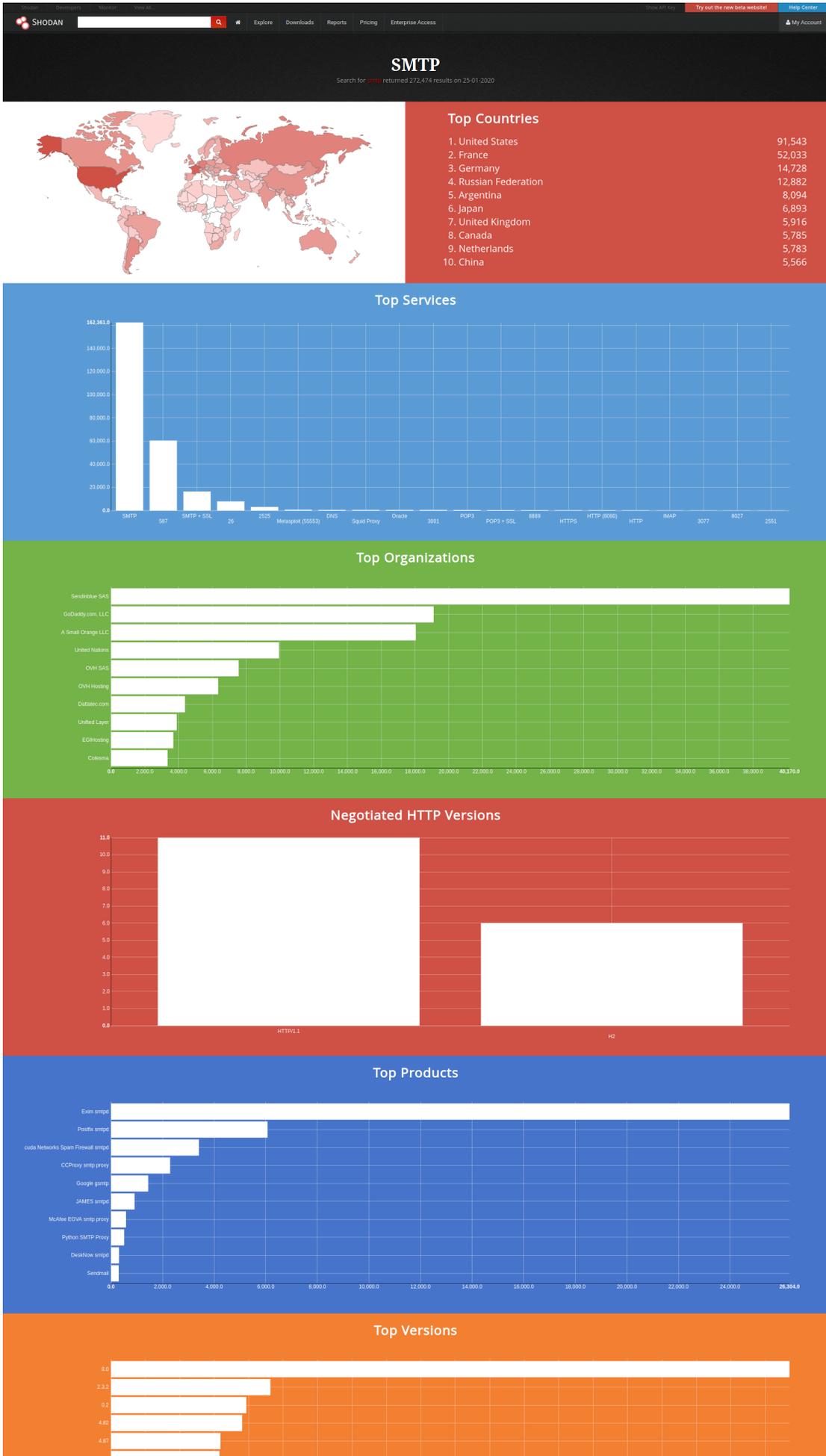
Create a report that provides statistics and breakdowns on various facets of your search query.

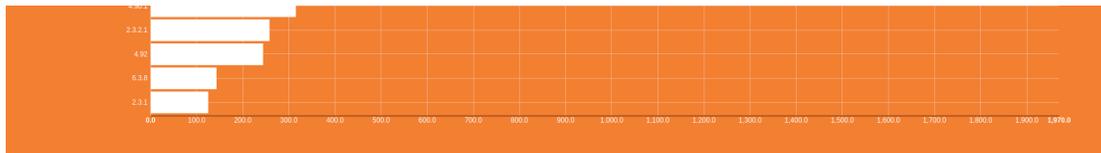
Title:

Close **Create Report**

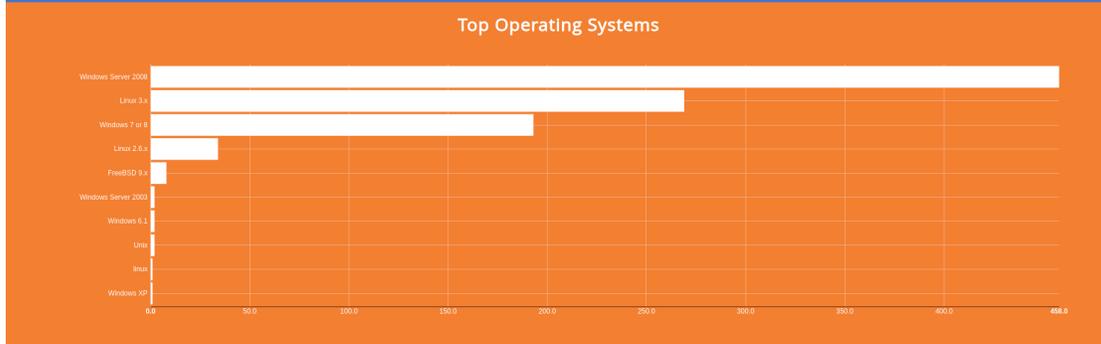
Creating a report will usually take a few minutes, you will receive an email when the report is ready with the link.

Else you can find all your previous reports on the [report page](#).





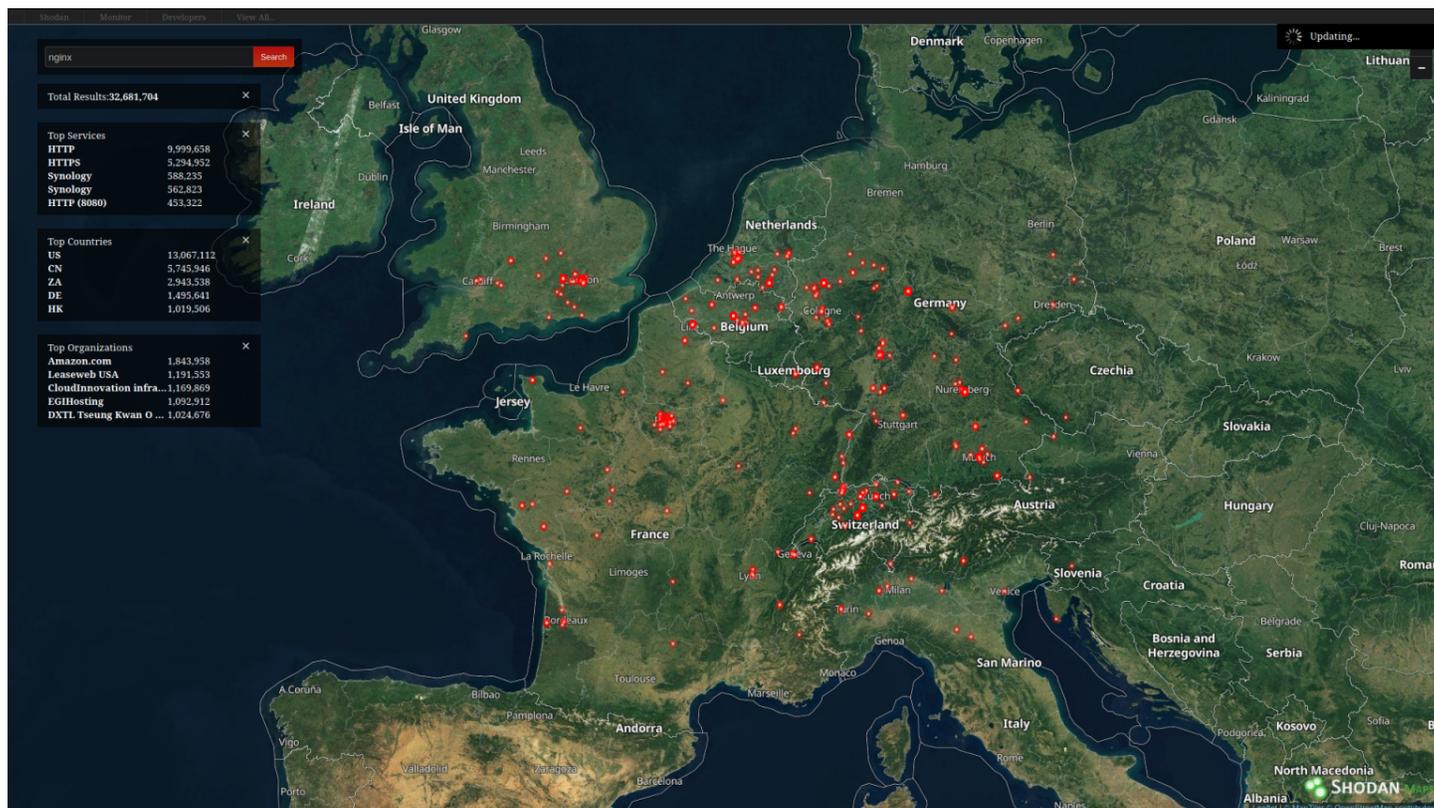
Top Domains



A report is static and won't update automatically.

Maps

The [map interface](#) to search the Shodan database works like the stats command of the CLI but displays the results in an interactive map depending on the physical location of the host.



As it won't show more than 1000 results, you will have to zoom in and out or move around to display other results

Images

[Images](#) is a searchable gallery of screenshots from crawled devices.

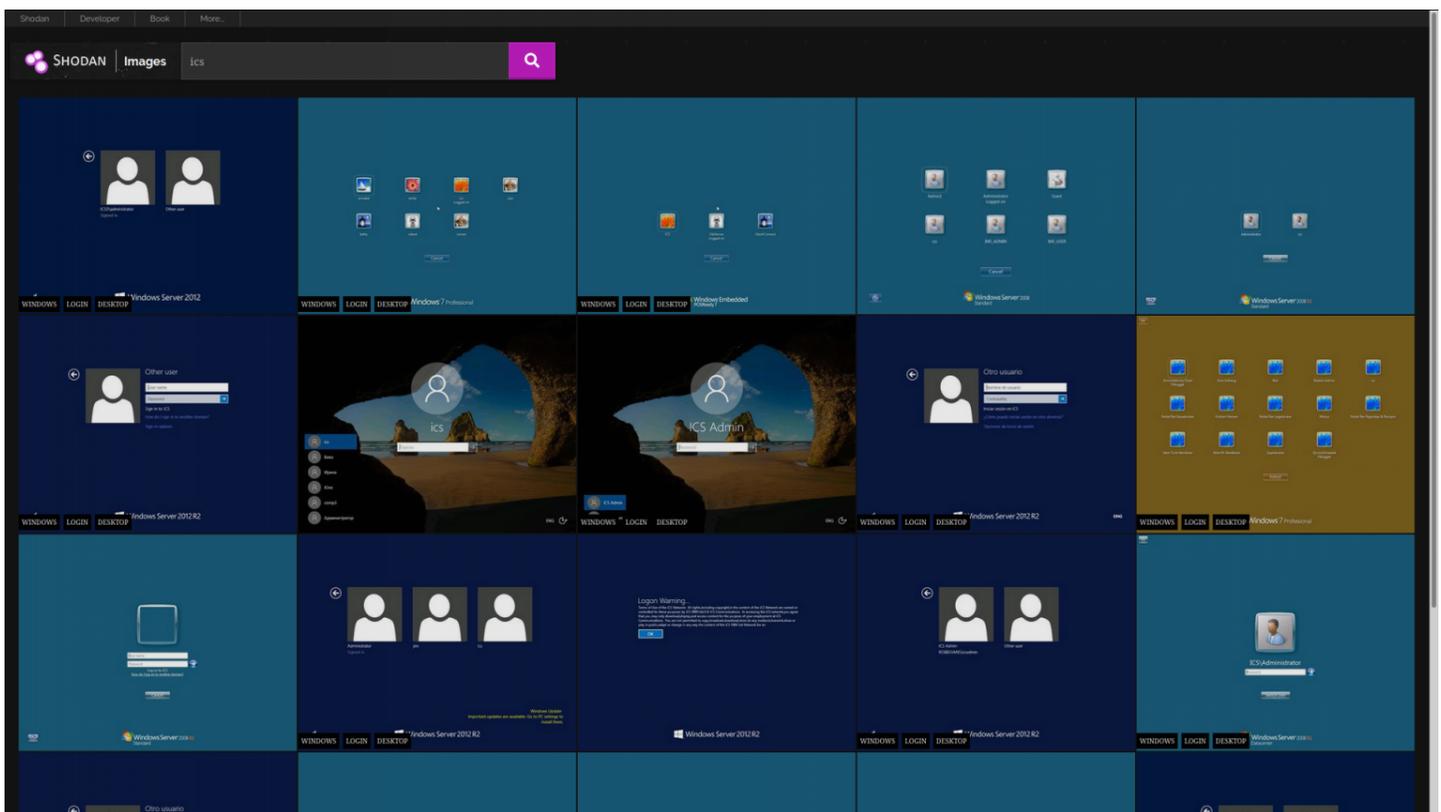


Image data is gathered from 5 different sources: VNC, Remote Desktop (RDP), RTSP, Webcams and X Windows.

A `has_screenshot:true` filter can be used in the global search engine to keep only hosts that have a screenshot.

Exploits

Exploits is a search engine that looks for exploits across a variety of vulnerability databases at once.

The screenshot shows the Shodan search interface with 'mongodb' as the search term. The results are categorized by source, platform, type, and author. The top result is 'MongoDB nativeHelper.apply Remote Code Execution' by agix, with a link to the exploit. Other results include 'MongoDB - 'conn' Mongo Object Remote Code Execution' by SCRT Security and 'MongoDB 2.2.3 - nativeHelper.apply Remote Code Execution' by agix. The interface also shows a sidebar with filters and a search bar.

Developer dashboard

Your **developer dashboard** shows you your credits consumption and API plan.

The screenshot displays the Shodan Developer dashboard. It features a '30 DAY USAGE' line chart showing a spike in usage around January 23, 2020. Below the chart is a 'MONTHLY USAGE' table:

Month	Query Credits Used	Scan Credits Used
Jan, 2020	6	4

The dashboard also includes a 'CURRENT API PLAN' section titled 'Developer API plan' with three credit usage metrics:

- Query Credits Used:** 0% (94 query credits available)
- Scan Credits Used:** 4% (96 scan credits available)
- IPs Monitored:** 0% (16 IPs available to monitor)

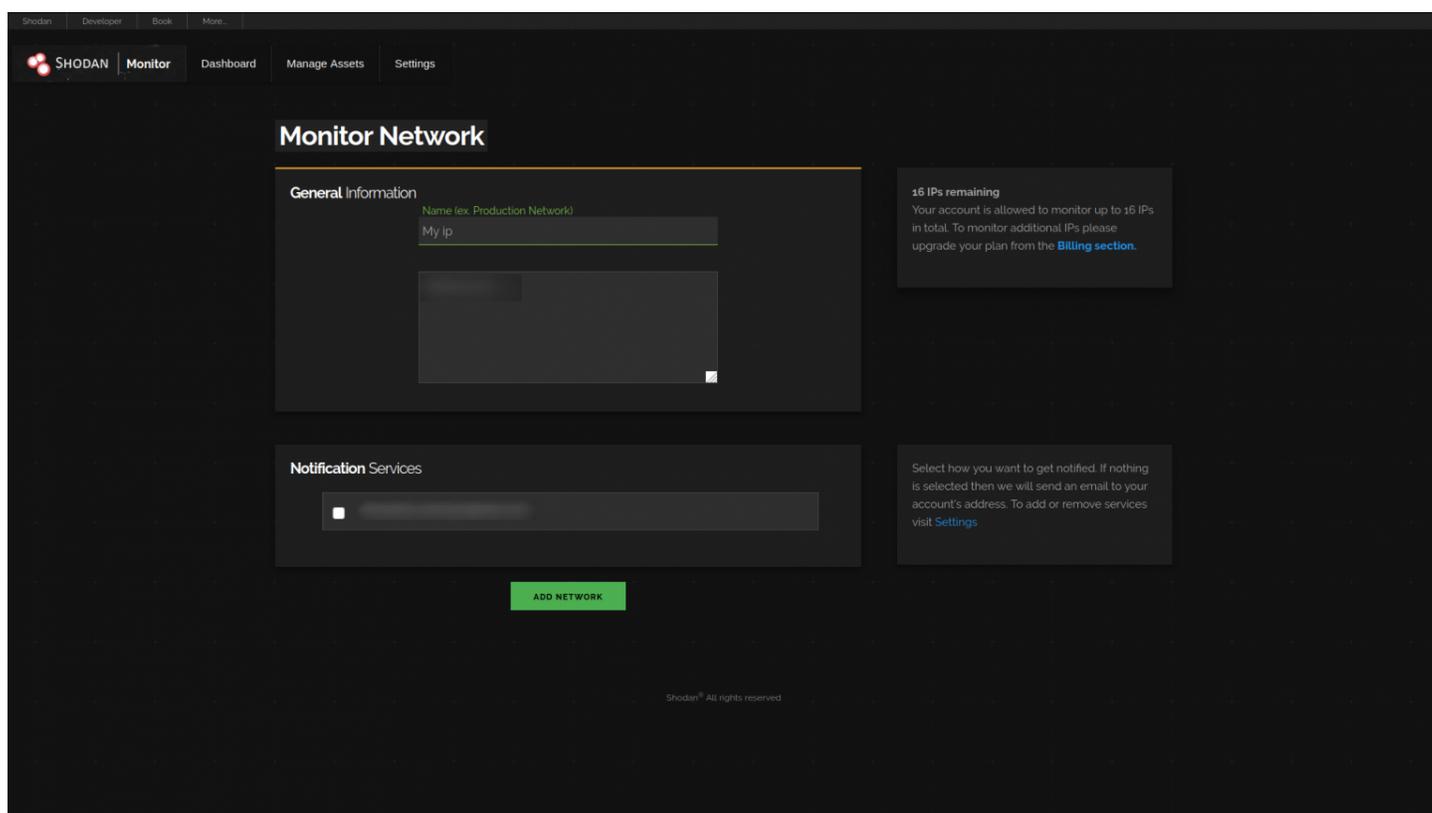
At the bottom, there is a 'GET FEATURED!' section with contact information for support@shodan.io. The footer indicates 'Shodan © 2013-2019, All Rights Reserved'.

Network monitor

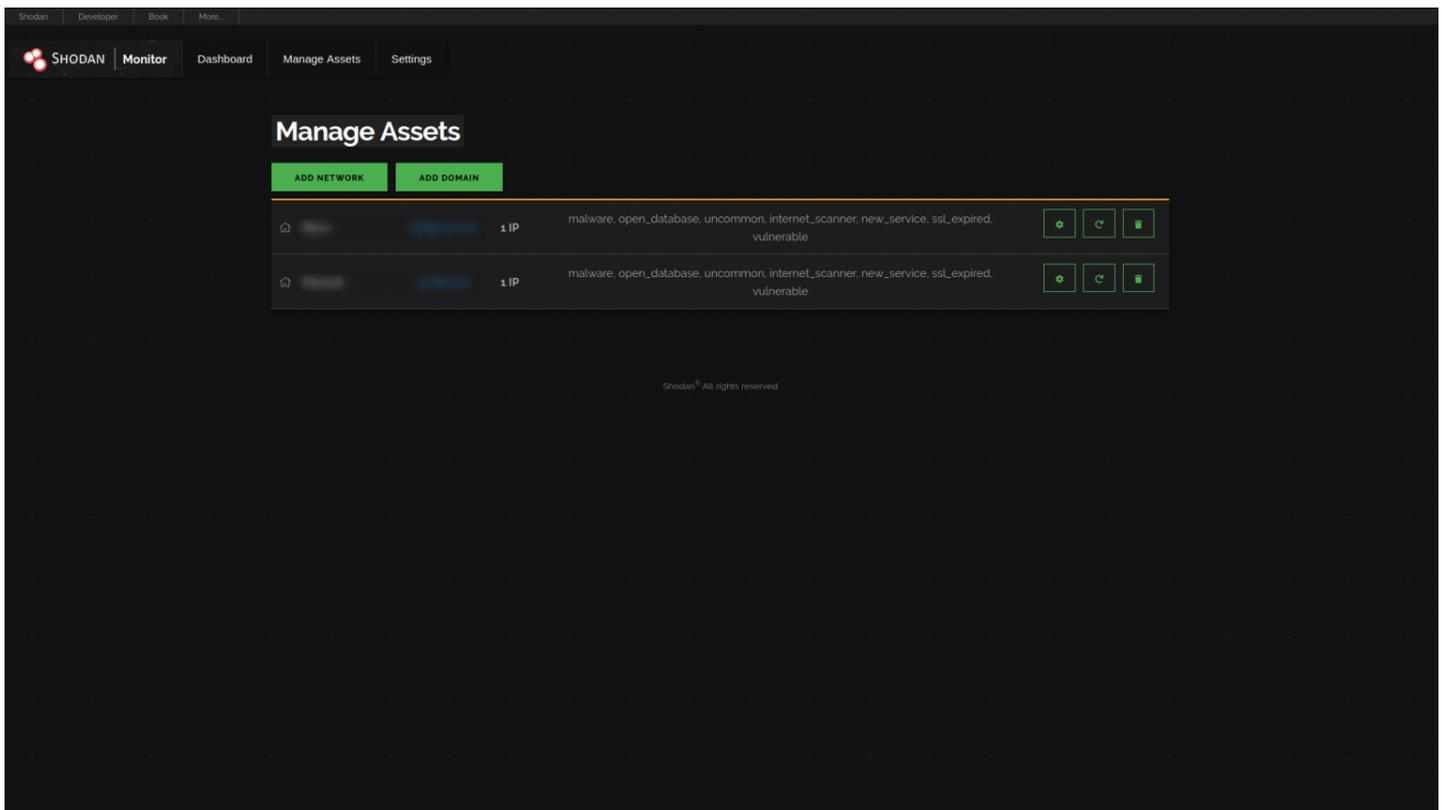
Keep track of the devices that you have exposed to the Internet. Setup notifications, launch scans and gain complete visibility into what you have connected.

The [monitor](#) dashboard let you tracks your devices, alert you if something suspicious was detected, launch scan and display what's found on synthetic dashboard.

To begin with, add an IP, a range or a domain to monitor and choose a notification service.



Then you can manage your assets, from here you can launch scans or modify trigger rules.



You can select which kind of event will trigger an alert.

Trigger Rules

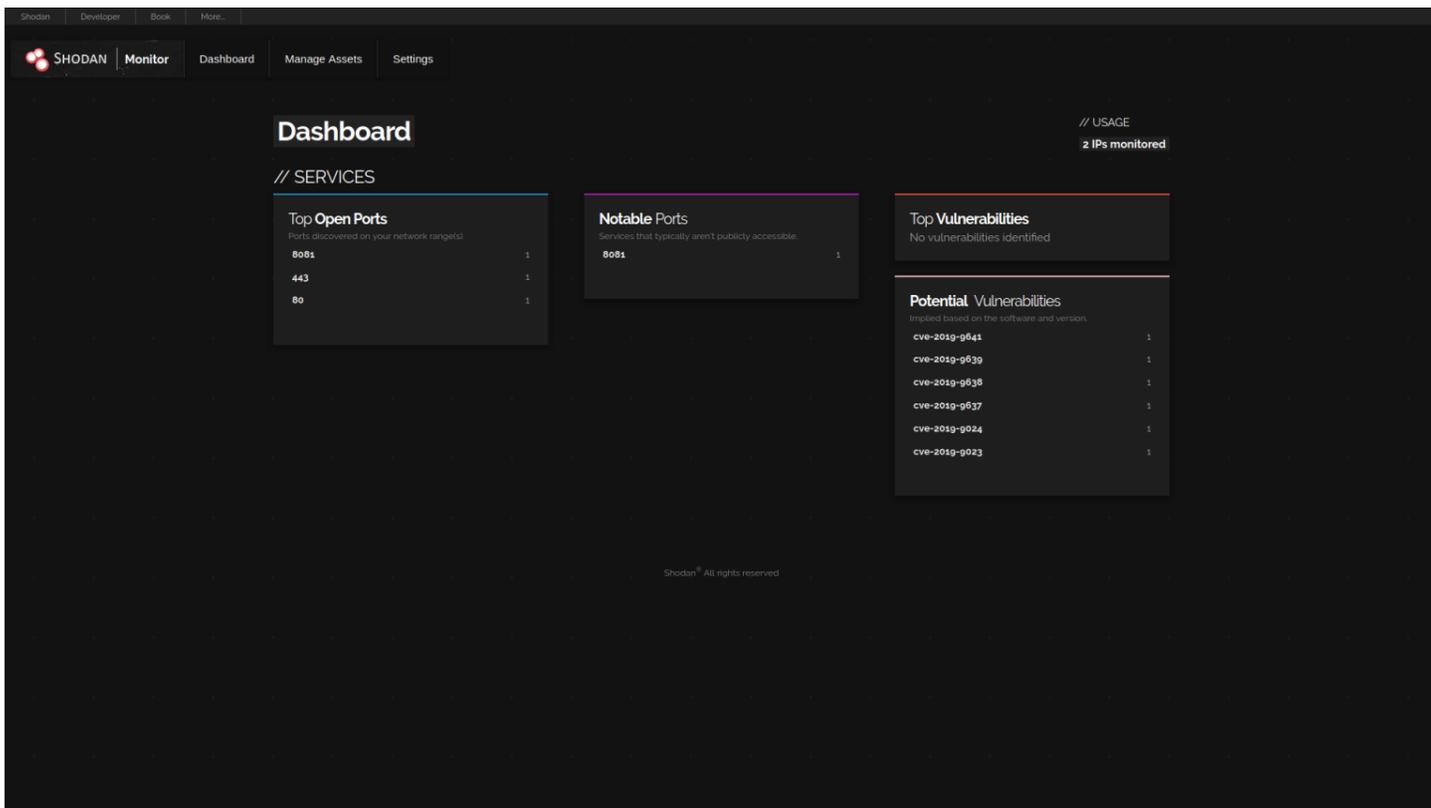
Select the types of notifications that you would like to receive. If none are selected we will let you know whenever Shodan discovers any service.

<input type="checkbox"/>	industrial_control_system	i
<input checked="" type="checkbox"/>	internet_scanner	i
<input type="checkbox"/>	iot	i
<input checked="" type="checkbox"/>	malware	i
<input checked="" type="checkbox"/>	new_service	i
<input checked="" type="checkbox"/>	open_database	i
<input checked="" type="checkbox"/>	ssl_expired	i
<input checked="" type="checkbox"/>	uncommon	i
<input checked="" type="checkbox"/>	vulnerable	i

[SAVE CHANGES](#)

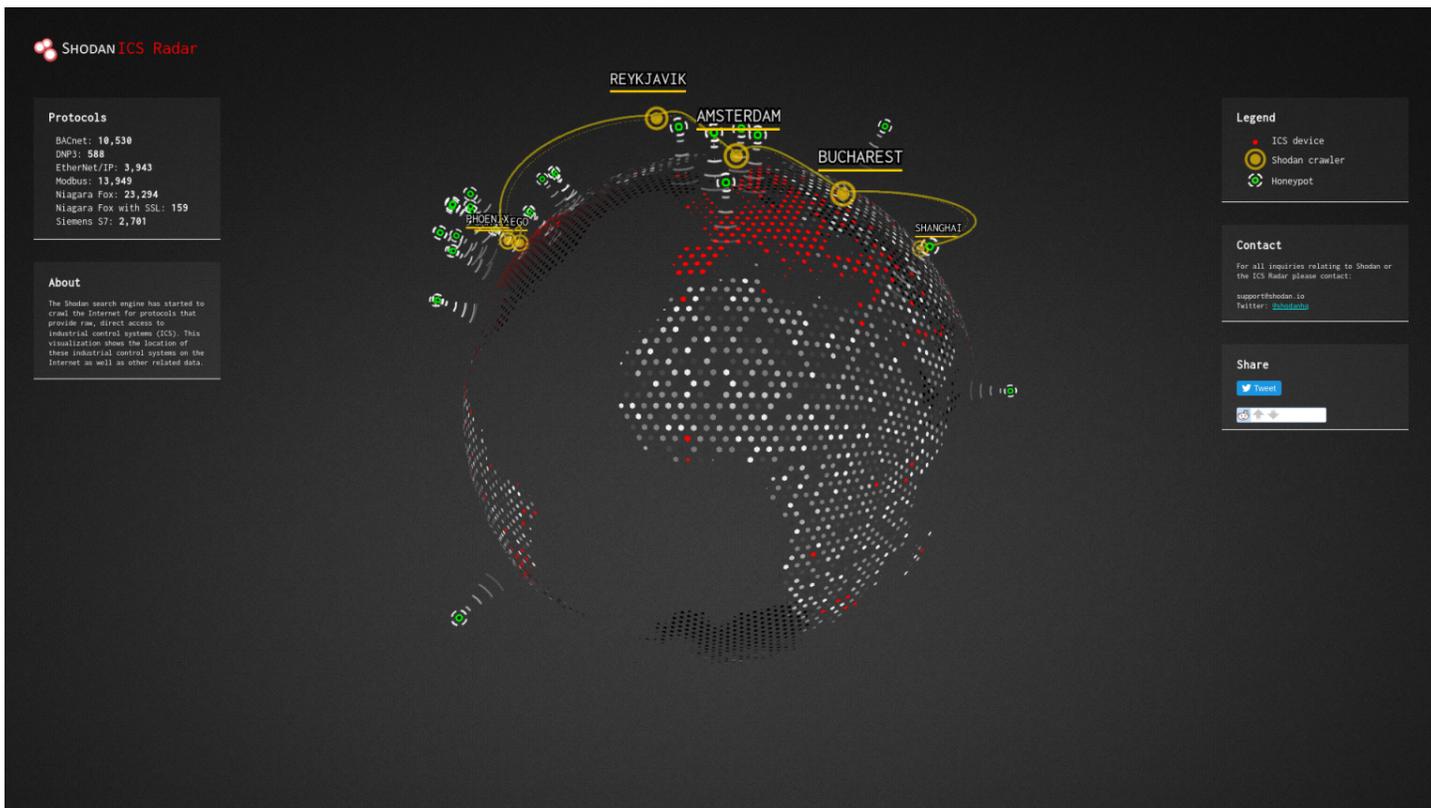
[Remove Network](#)

Then the dashboard shows the exposed services.



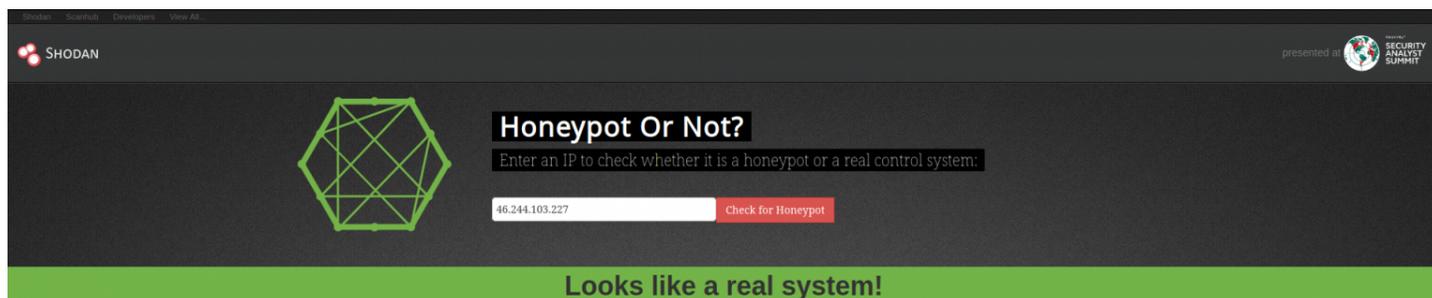
ICS radar

ICS Radar is a 3D map of Industrial Control Systems (ICS) devices found by Shodan crawlers.



Honeypot score

The service called [Honeypot or not?](#) will attribute a *Honeyscore* to an IP address, a probability of being a honeypot.



Frequently Asked Questions

1. How does it work?

The defining characteristics of known honeypots were extracted and used to create a tool to let you identify honeypots! The probability that an IP is a honeypot is captured in a "Honeyscore" value that can range from 0.0 to 1.0. This is still a prototype! work-in-progress so if you find some problems please email me at jmath@shodan.io

2. What's the purpose?

Honeypots are a great tool for learning more about the Internet, the latest malware being used and keep track of infections. When trying to catch an intelligent attacker though, many honeypots fall short in creating a realistic environment. Honeyscore was created to raise awareness of the short-comings of honeypots.

3. What technology did you use?

The Honeyscore website and algorithm uses the following APIs/ frameworks:

- Shodan Developer API
- Python
- Jade Node Template Engine

4. Contact information?

You can reach me at the following locations:

Email: jmath@shodan.io
Twitter: [@achilleian](#)

It's just an abstraction of the API like the honeyscore command of the CLI:

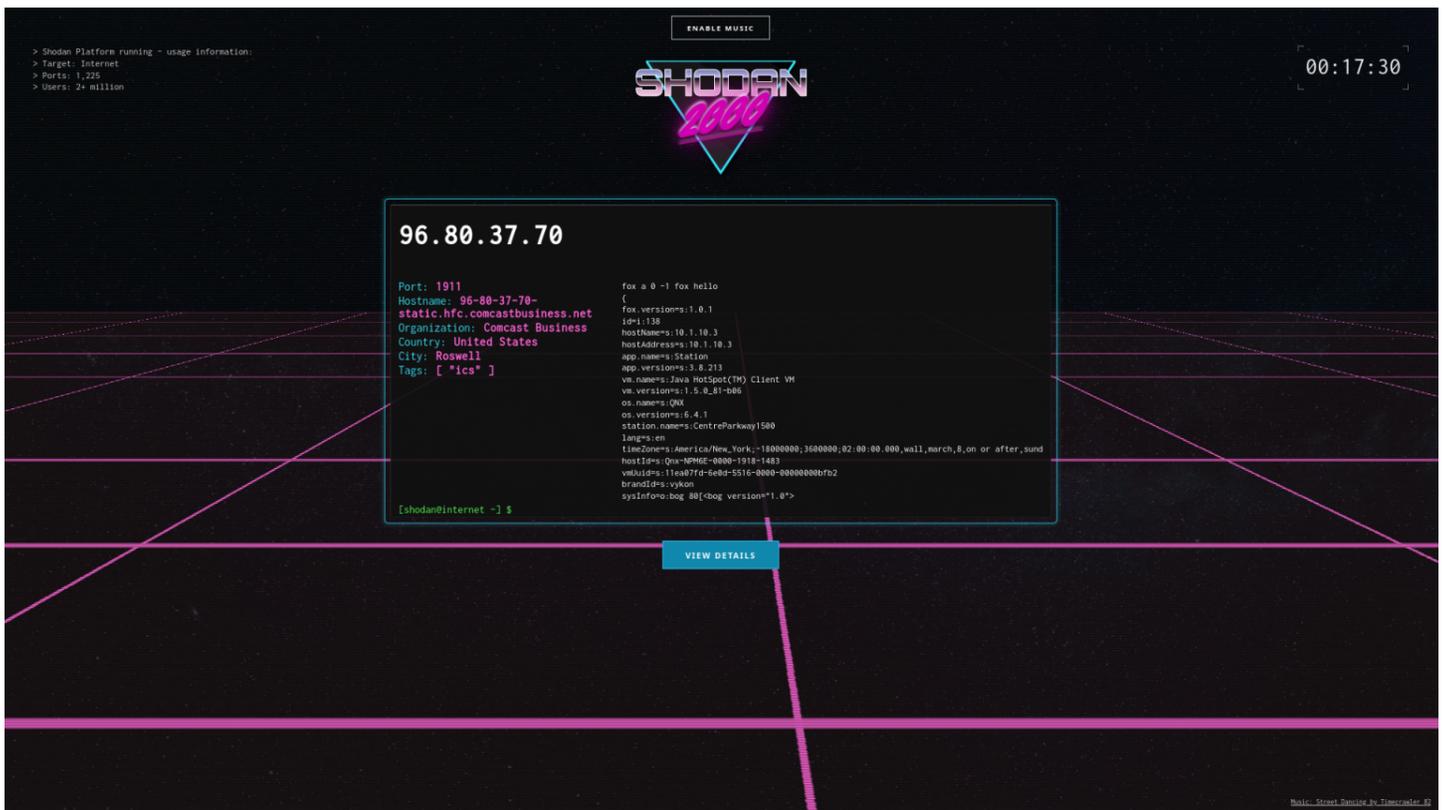
```
$ shodan honeyscore 46.244.103.227
```

```
Not a honeypot
```

```
Score: 0.3
```

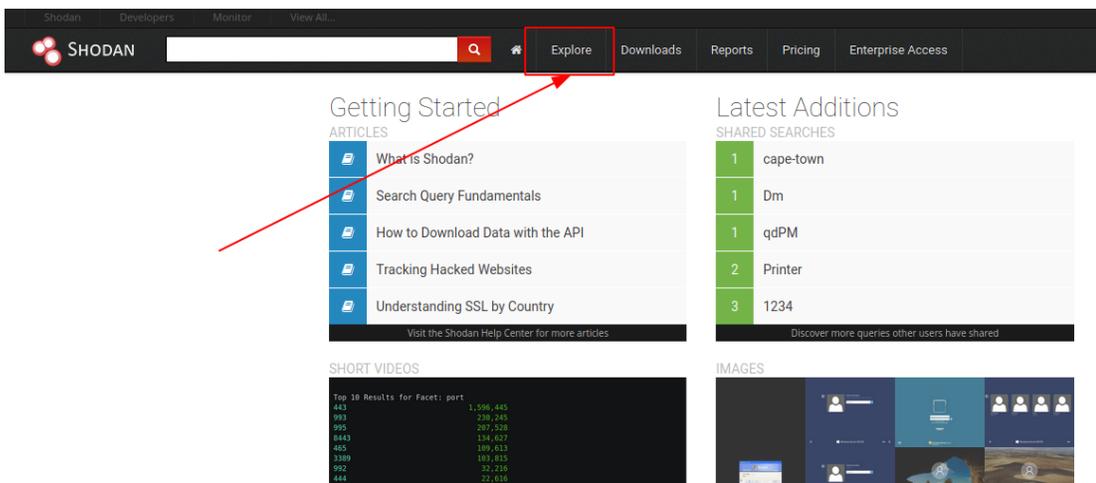
Shodan 2000

[Sodan 2000](#) is a Tron-like interface that randomly displays an host.



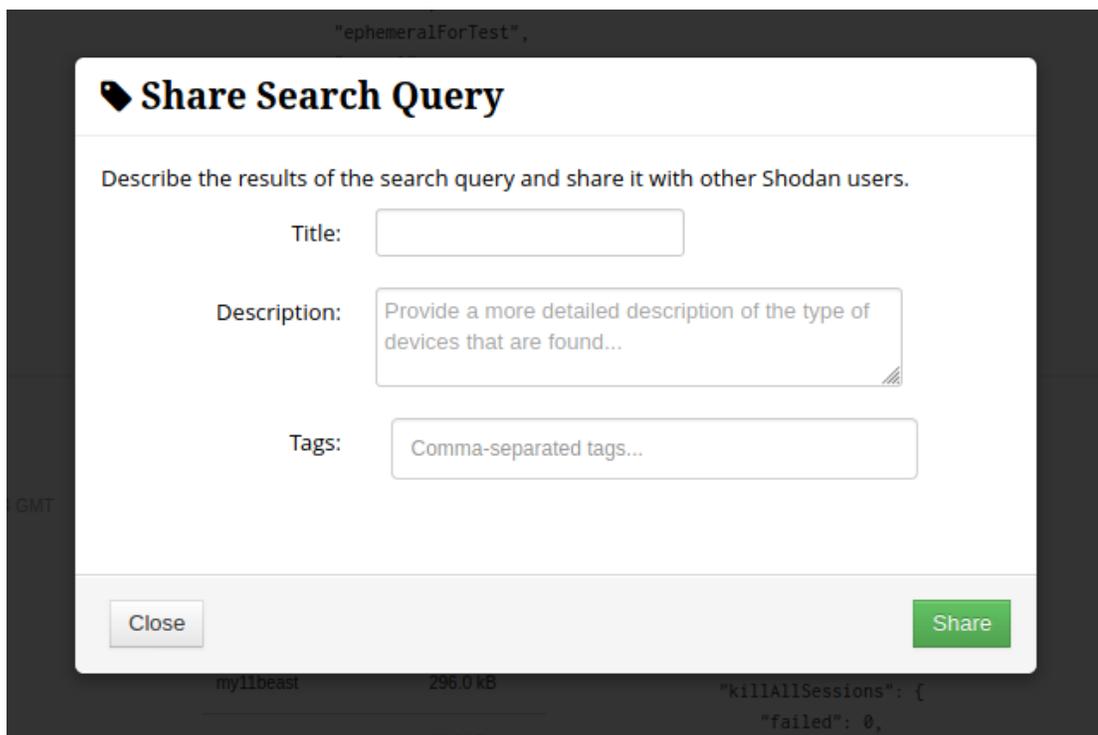
Community queries

You can [explore](#) queries exported and shared by other users of the community.



The shared queries have a title, a description and tags so you can browse them easily.

To share a query, click the *Share Search* button on a search result page.



REST API

Shodan comes with a REST API, it can be used to build a web application service based on Shodan or create a wrapper library if none already exists in your favorite language.

The base URL of the API is: <https://api.shodan.io> and all API methods are rate-limited to 1 req/sec.

The API is authenticated so if you forget to provide your API key, you will get a HTTP 401 error.

Here is an example of how you can query your API Plan Information with curl:

```
curl -s https://api.shodan.io/api-info?key={YOUR_API_KEY} | jq
```

```
{
  "scan_credits": 95,
  "usage_limits": {
    "scan_credits": 100,
    "query_credits": 100,
```

```
"monitored_ips": 16
},
"plan": "dev",
"https": false,
"unlocked": true,
"query_credits": 94,
"monitored_ips": 2,
"unlocked_left": 94,
"telnet": false
}
```

Another query to get a host information:

```
curl -s https://api.shodan.io/shodan/host/1.1.1.1?key=
{YOUR_API_KEY} | jq
```

```
{
"region_code": null,
"ip": 16843009,
"postal_code": null,
"country_code": "AU",
"city": null,
"dma_code": null,
"last_update": "2020-01-25T15:55:54.880090",
"latitude": -33.494,
"tags": [],
"area_code": null,
"country_name": "Australia",
"hostnames": [
"one.one.one.one"
],
"org": "Mountain View Communications",
"data": [
```

```
{
  "_shodan": {
    "id": "f4218ca0-2728-4d7b-97f8-875f4f04149d",
    "options": {
      "referrer": "601b650e-3cc7-4189-babe-921fdf53a9e2",
      "hostname": "www.1yhaoduo.com"
    },
    "ptr": true,
    "module": "http",
    "crawler": "d264629436af1b777b3b513ca6ed1404d7395d80",
    "hash": -237371161,
    "os": null,
    "opts": {},
    "ip": 16843009,
    "isp": "APNIC and Cloudflare DNS Resolver project",
    "http": {
      "html_hash": 1145258596,
      "robots_hash": null,
      "redirects": [],
      "securitytxt": null,
      "title": "DNS resolution error | www.1yhaoduo.com | Cloudflare",
      "sitemap_hash": null,
      "waf": "CloudFlare",
      "robots": null,
      "favicon": null,
      "host": "www.1yhaoduo.com",
      ...
    }
  }
}
```

Check the [REST API Documentation](#) for a complete description of all methods.

Language wrappers (libraries)

To interface your tool with the Shodan API you can use one of the wrapper libraries.

The official one is made in Python, but there are also [community libraries](#) in Ruby, PHP, Haskell, Rust, Perl, Node.js, Go, PowerShell, Java and C#.

I will give examples for those three:

- [Python – shodan-python](#)
- [Ruby – shodanz](#)
- [Node.js – shodan-client](#)

Python – shodan-python

Installation

The installation is the same as for the CLI tool as the CLI tool is made upon the python library, they are packaged together.

In a virtual python environment like [pyenv](#):

```
$ easy_install shodan
```

On [BlackArch](#) you can also install the following package:

```
# pacman -S python-shodan
```

Then the API key will always be initialized like that in our code:

```
import shodan
```

```
SHODAN_API_KEY = 'API key here'
```

```
api = shodan.Shodan(SHODAN_API_KEY)
```

Note: the library is working for both python 2 and 3 but we'll use only python 3 as python 2 is deprecated.

Examples

Basic search:

```
try:
```

```
    # Search Shodan
    results = api.search('apache')

    ## Show results
    print('Results found: {}'.format(results['total']))
    for result in results['matches']:
        print('IP: {}'.format(result['ip_str']))
    print(result['data'])
    print('')
except shodan.APIError as e:
    print('Error: {}'.format(e))
```

Example of output:

```
IP: 65.99.237.196
HTTP/1.1 200 OK
Date: Sat, 25 Jan 2020 16:07:19 GMT
Server: Apache
Transfer-Encoding: chunked
Content-Type: text/html
```

```
IP: 212.72.184.58
HTTP/1.1 200 OK
Date: Sat, 25 Jan 2020 16:07:29 GMT
Server: Apache/2.2.22 (Debian) mod_python/3.3.1 Python/2.7.3
mod_ssl/2.2.22 OpenSSL/1.0.1t
```

```
X-Powered-By: PHP/5.4.45-0+deb7u14
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Last-Modified: Sat, 25 Jan 2020 16:07:29 GMT
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html
```

```
IP: 208.109.44.217
HTTP/1.1 404 Not Found
Date: Sat, 25 Jan 2020 16:07:20 GMT
Server: Apache
Content-Length: 381
Content-Type: text/html; charset=iso-8859-1
```

Available ports of a host:

```
try:
    # Lookup the host
    host = api.host('1.1.1.1')

    # Print general info
    print("""
        IP: {}
        Organization: {}
        Operating System: {}
        """.format(host['ip_str'], host.get('org', 'n/a'),
        host.get('os', 'n/a')))

    # Print all banners
```

```
for item in host['data']:  
    print("""  
        Port: {}  
        Banner: {}  
        """.format(item['port'], item['data']))  
except shodan.APIError as e:  
    print('Error: {}'.format(e))
```

Example of output:

```
IP: 1.1.1.1  
Organization: Mountain View Communications  
Operating System: None
```

```
Port: 80  
Banner: HTTP/1.1 409 Conflict  
Date: Sat, 25 Jan 2020 15:55:54 GMT  
Content-Type: text/html; charset=UTF-8  
Transfer-Encoding: chunked  
Connection: close  
Set-Cookie: __cfduid=d6241813d879cf2a39d03f5d6ce5a1abc1579967754;  
expires=Mon, 24-Feb-20 15:55:54 GMT; path=/;  
domain=.www.1yhaoduo.com; HttpOnly; SameSite=Lax  
Cache-Control: max-age=6  
Expires: Sat, 25 Jan 2020 15:56:00 GMT  
X-Frame-Options: SAMEORIGIN  
Vary: Accept-Encoding  
Server: cloudflare  
CF-RAY: 55ab6f23aee09cbd-AMS
```

```
Port: 443
```

```
Banner: HTTP/1.1 301 Moved Permanently
Date: Sat, 25 Jan 2020 15:47:19 GMT
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: max-age=3600
Expires: Sat, 25 Jan 2020 16:47:19 GMT
Location: https://get.vitalsource.com/
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Vary: Accept-Encoding
Server: cloudflare
CF-RAY: 55ab628f3b05acca-OTP
```

Port: 53

Banner:

```
\x00\x00\x80\x83\x00\x01\x00\x00\x00\x01\x00\x00\t_services\x07_dns-
sd\x04_udp\x05local\x00\x00\x0c\x00\x01\x00\x00\x06\x00\x01\x00\x0
0(\xac\x00@\x01a\x0croot-servers\x03net\x00\x05ns1d\x0cverisign-
grs\x03com\x00f\x0f1\xd4\x00\x00\x07\x08\x00\x00\x03\x84\x00\t:\x8
0\x00\x01q\x80
```

Displaying stats:

The list of properties we want summary information on

```
FACETS = [
    ('org', 3),
    'domain',
    'port',
    'asn',
    ('country', 10),
]
```

```
FACET_TITLES = {
    'org': 'Top 3 Organizations',
    'domain': 'Top 5 Domains',
    'port': 'Top 5 Ports',
    'asn': 'Top 5 Autonomous Systems',
    'country': 'Top 10 Countries',
}

try:
    # Query
    query = 'apache 2.4'

    # Count results
    result = api.count(query, facets=FACETS)

    print('Shodan Summary Information')
    print('Query: %s' % query)
    print('Total Results: %s\n' % result['total'])

    # Print the summary info from the facets
    for facet in result['facets']:
        print(FACET_TITLES[facet])

        for term in result['facets'][facet]:
            print('%s: %s' % (term['value'], term['count']))

    # Print an empty line between summary info
    print('')

except shodan.APIError as e:
    print('Error: {}'.format(e))
```

Example of output:

Shodan Summary Information

Query: apache 2.4

Total Results: 64678

Top 3 Organizations

Liquid Web, L.L.C: 23199

Amazon.com: 7588

Hetzner Online GmbH: 1818

Top 5 Domains

amazonaws.com: 10679

telecom.net.ar: 1661

your-server.de: 1243

t-ipconnect.de: 664

vultr.com: 443

Top 5 Ports

80: 21212

443: 19890

8080: 3024

10000: 1723

8081: 1366

Top 5 Autonomous Systems

as53824: 13848

as32244: 9351

as16509: 6294

as24940: 1759

as7303: 1453

Top 10 Countries

```
US: 31090
DE: 5833
CN: 4554
BR: 3010
AR: 1809
JP: 1475
GB: 1168
IN: 1009
FR: 756
CA: 613
```

Note: this examples comes from the [official documentation](#) but were adapted for Python 3 and updated to better suit this article.

Ruby – shodanz

Installation

In a virtual ruby environment like [rbenv](#):

```
$ gem install shodanz
```

Then the API key will always be initialized like that in our code:

```
require 'shodanz'
```

```
api = Shodanz.client.new(key: 'YOUR_API_KEY')
```

For production projects you may prefer read the API key via the environment variable SHODAN_API_KEY.

Examples

Basic search:

```
# Search Shodan
results = api.host_search('apache')
```

```
# Show results
puts "Results found: #{results['total']}"
results['matches'].each do |result|
  puts "IP: #{result['ip_str']}"
  puts result['data'] + "\n"
end
```

Example of output:

```
IP: 154.218.139.58
HTTP/1.1 200 OK
Date: Tue, 28 Jan 2020 22:13:53 GMT
Server: Apache
Upgrade: h2
Connection: Upgrade, close
Last-Modified: wed, 26 Apr 2017 08:03:47 GMT
ETag: "52e-54e0d47a39ec0"
Accept-Ranges: bytes
Content-Length: 1326
Vary: Accept-Encoding
Content-Type: text/html
```

```
IP: 132.148.235.102
HTTP/1.1 200 OK
Date: Tue, 28 Jan 2020 22:13:53 GMT
Server: Apache
Upgrade: h2,h2c
Connection: Upgrade
Last-Modified: Fri, 10 May 2019 09:10:49 GMT
ETag: "a4edb-7ab-58884f152c219"
Accept-Ranges: bytes
```

```
Content-Length: 1963
Vary: Accept-Encoding,User-Agent
Content-Type: text/html
```

```
IP: 112.126.140.94
HTTP/1.1 404 Not Found
Date: Tue, 28 Jan 2020 22:13:34 GMT
Server: Apache
X-Powered-By: PHP/5.2.17
X-UA-Compatible: IE=EmulateIE7
Transfer-Encoding: chunked
Content-Type: text/html
```

Available ports of a host:

```
# Lookup the host
host = api.host('1.1.1.1')

# Print general info
puts "
  IP: #{host['ip_str']}
  Organization: #{host['org'] || 'n/a'}
  Operating System: #{host['os'] || 'n/a'}
"

# Print all banners
host['data'].each do |item|
  puts "
    Port: #{item['port'] || 'n/a'}
    Banner: #{item['data'] || 'n/a'}
  "
```

"

end

Example of output:

IP: 1.1.1.1

Organization: Mountain View Communications

Operating System: n/a

Port: 443

Banner: HTTP/1.1 403 Forbidden

Server: cloudflare

Date: Tue, 28 Jan 2020 18:34:35 GMT

Content-Type: text/html

Content-Length: 553

Connection: keep-alive

CF-RAY: 55c50fb4e8149d5a-AMS

Port: 80

Banner: HTTP/1.1 409 Conflict

Date: Tue, 28 Jan 2020 17:26:54 GMT

Content-Type: text/html; charset=UTF-8

Transfer-Encoding: chunked

Connection: close

Set-Cookie: __cfduid=d189a930262f96d94a707a90d853a56bd1580232414;

expires=Thu, 27-Feb-20 17:26:54 GMT; path=/;

domain=.www.1yhaoduo.com; HttpOnly; SameSite=Lax

Cache-Control: max-age=6

Expires: Tue, 28 Jan 2020 17:27:00 GMT

X-Frame-Options: SAMEORIGIN

Vary: Accept-Encoding

Server: cloudflare

CF-RAY: 55c4ac8fba63801a-SAN

Port: 53

Banner:

Recursion: enabled

Resolver ID: AMS

Displaying stats:

```
# The list of properties we want summary information on
```

```
FACETS = {
```

```
'org': 3,
```

```
'domain': 5,
```

```
'port': 5,
```

```
'asn': 5,
```

```
'country': 10,
```

```
}
```

```
FACET_TITLES = {
```

```
'org': 'Top 3 Organizations',
```

```
'domain': 'Top 5 Domains',
```

```
'port': 'Top 5 Ports',
```

```
'asn': 'Top 5 Autonomous Systems',
```

```
'country': 'Top 10 Countries',
```

```
}
```

```
# Query
```

```
query = 'apache 2.4'
```

```
# Count results
```

```
result = api.host_count(query, facets: FACETS)
```

```
puts 'Shodan Summary Information'
```

```
puts "Query: #{query}"
puts "Total Results: #{result['total']}\n"

# Print the summary info from the facets
result['facets'].each do |facet, _v|
  puts FACET_TITLES[facet]

  result['facets'][facet].each do |term|
    puts "#{term['value']}: #{term['count']}"
  end

  # Print an empty line between summary info
  puts ''
end
```

Example of output:

Shodan Summary Information

Query: apache 2.4

Total Results: 63939

Liquid Web, L.L.C: 23126

Amazon.com: 7843

Hetzner Online GmbH: 1798

amazonaws.com: 10398

telecom.net.ar: 1609

your-server.de: 1232

t-ipconnect.de: 629

vultr.com: 450

```
80: 21131
443: 19772
8080: 3023
10000: 1672
8081: 1372
```

```
as53824: 13810
as32244: 9316
as16509: 6138
as24940: 1740
as7303: 1410
```

```
US: 30877
DE: 5781
CN: 4432
BR: 2949
AR: 1757
JP: 1472
GB: 1168
IN: 1030
FR: 720
CA: 613
```

Async support with the stream API:

```
require 'async'
require 'shodanz'
```

```
api = Shodanz.client.new(key: 'YOUR_API_KEY')
```

```
# Asynchronously stream banner info from shodan and check any
```

```
# IP addresses against the experimental honeypot scoring service.
api.streaming_api.banners do |banner|
  if ip = banner['ip_str']
    Async do
      score = api.rest_api.honeypot_score(ip).wait
      puts "#{ip} has a #{score * 100}% chance of being a
honeypot"
      rescue Shodanz::Errors::RateLimited
      sleep rand
      retry
      rescue # any other errors
      next
    end
  end
end
end
```

Warning: Freelancer API plan or better required for using the stream API, developer or free plan won't work.

Note: this async example comes from the shodanz documentation.

Useful methods

```
# Returns all the protocols that can be used when launching an
Internet scan
api.protocols

# Returns a list of port numbers that the Shodan crawlers are
looking for
api.ports
```

```
# Returns information about the Shodan account linked to this API key
```

```
api.profile
```

```
# Look up the IP address for the provided list of hostnames
```

```
api.resolve('archlinux.org', 'blackarch.org')
```

```
# Look up the hostnames that have been defined for the given list of IP addresses
```

```
api.reverse_lookup('138.201.81.199', '176.31.253.211')
```

```
# Get your current IP address as seen from the Internet
```

```
api.my_ip
```

```
# Calculates a honeypot probability score ranging from 0 (not a honeypot) to 1.0 (is a honeypot)
```

```
api.honeypot_score('1.1.1.1')
```

```
# API Plan Information
```

```
api.info
```

Exploits API

```
puts client.exploit_count(port: 22, page: 1)
```

```
puts client.exploit_search('rce couchdb', type: 'remote',
```

```
platform: 'linux', author: 'Metasploit')
```

You can find more examples [here](#) or read the shodanz [API documentation](#).

Node.js – shodan-client

Installation

In a virtual nodejs environment like [nodenv](#):

```
$ npm i shodan-client
```

Then the API key will always be initialized like that in our code:

```
const util = require('util');  
const api = require('shodan-client');
```

```
const key = 'API key here';
```

Examples

Basic search

```
const searchOpts = {};
```

```
const searchQuery = 'apache';
```

api

```
.search(searchQuery, key, searchOpts) // Search Shodan  
  .then(results => {  
    console.log('Results found: ' + results['total'] + "\n"); //
```

Show results

```
    for (const result of results['matches']) {  
      console.log(`IP: ${result['ip_str']}`);  
      console.log(result['data'] + "\n");  
    }  
  })  
  .catch(err => {  
    console.log('Error:');  
    console.log(err);  
  });
```

Example of output:

Results found: 25855805

IP: 210.143.102.156

HTTP/1.1 302 Found

Date: Sat, 01 Feb 2020 18:45:43 GMT

Server: Apache/2.2.15 (Scientific Linux)

Location: https://210.143.102.156/

Content-Length: 299

Connection: close

Content-Type: text/html; charset=iso-8859-1

IP: 52.168.162.242

HTTP/1.1 200 OK

Date: Sat, 01 Feb 2020 18:44:49 GMT

Server: Apache

X-Frame-Options: SAMEORIGIN

Last-Modified: Tue, 13 Aug 2019 14:51:43 GMT

ETag: "f11-59000c7615dc0"

Accept-Ranges: bytes

Content-Length: 3857

X-XSS-Protection: 1; mode=block

Cache-Control: no-cache, no-store, must-revalidate

Pragma: no-cache

Expires: 0

Content-Type: text/html; charset=UTF-8

Set-Cookie: pwcount=2;Secure;Path=/

Cache-Control: no-cache

```
IP: 217.160.91.209
HTTP/1.1 403 Forbidden
Date: Sat, 01 Feb 2020 18:45:18 GMT
Server: Apache
Content-Length: 1364
X-Frame-Options: deny
Content-Type: text/html
```

Available ports of a host

```
const searchOpts = {};
```

```
const ip = '1.1.1.1';
```

```
api
```

```
.host(ip, key, searchOpts) // Lookup the host
  .then(host => {
    // Print general info
    console.log(`
      IP: ${host['ip_str']}
      Organization: ${host['org']} || 'n/a'}
      Operating System: ${host['os']} || 'n/a'}
    `);
    // Print all banners
    for (const item of host['data']) {
      console.log(`
        Port: ${item['port']} || 'n/a'}
        Banner: ${item['data']} || 'n/a'}
      `);
    }
  })
  .catch(err => {
    console.log('Error:');
  });
```

```
        console.log(err);  
    });
```

Example of output:

```
IP: 1.1.1.1  
Organization: Mountain View Communications  
Operating System: n/a
```

```
Port: 443  
Banner: HTTP/1.1 403 Forbidden  
Server: cloudflare  
Date: Sat, 01 Feb 2020 19:26:14 GMT  
Content-Type: text/html  
Content-Length: 553  
Connection: keep-alive  
CF-RAY: 55e650de89868020-SAN
```

```
Port: 80  
Banner: HTTP/1.1 409 Conflict  
Date: Sat, 01 Feb 2020 19:16:16 GMT  
Content-Type: text/html; charset=UTF-8  
Transfer-Encoding: chunked  
Connection: close  
Set-Cookie: __cfduid=dd6d38c961c18135646e1681bd1f809ad1580584576;  
expires=Mon, 02-Mar-20 19:16:16 GMT; path=/;  
domain=.www.1yhaoduo.com; HttpOnly; SameSite=Lax  
Cache-Control: max-age=6  
Expires: Sat, 01 Feb 2020 19:16:22 GMT  
X-Frame-Options: SAMEORIGIN
```

```
Vary: Accept-Encoding
Server: cloudflare
CF-RAY: 55e64240bb5a801a-SAN
```

Displaying stats

```
const FACETS = {
  'org': 3,
  'domain': 5,
  'port': 5,
  'asn': 5,
  'country': 10,
};

const FACET_TITLES = {
  'org': 'Top 3 Organizations',
  'domain': 'Top 5 Domains',
  'port': 'Top 5 Ports',
  'asn': 'Top 5 Autonomous Systems',
  'country': 'Top 10 Countries',
};

// https://github.com/jesusprubio/shodan-client/issues/34
// const opts = { facets: FACETS };
const opts = { facets: JSON.stringify(FACETS).replace(/["{}]/g,
  '') };

// Query
const query = 'apache 2.4';

api
  .count(query, key, opts) // Count results
  .then(result => {
```

```
console.log('Shodan Summary Information');
console.log(`Query: ${query}`);
console.log(`Total Results: ${result['total']}\n`);

// Print the summary info from the facets
for (const facet in result['facets']) {
  console.log(FACET_TITLES[facet]);

  for (const term of result['facets'][facet]) {
    console.log(`${term['value']}: ${term['count']}`);
  }

  // Print an empty line between summary info
  console.log('');
}
})
.catch(err => {
  console.log('Error:');
  console.log(err);
});
```

Example of output:

Shodan Summary Information

Query: apache 2.4

Total Results: 63112

Top 3 Organizations

Liquid Web, L.L.C: 22985

Amazon.com: 8614

Hetzner online GmbH: 1797

Top 5 Domains

amazonaws.com: 10051
telecom.net.ar: 1600
your-server.de: 1220
t-ipconnect.de: 603
vultr.com: 429

Top 5 Ports

80: 21098
443: 19669
8080: 3040
10000: 1669
8081: 1411

Top 5 Autonomous Systems

as53824: 13725
as32244: 9260
as16509: 5941
as24940: 1750
as7303: 1383

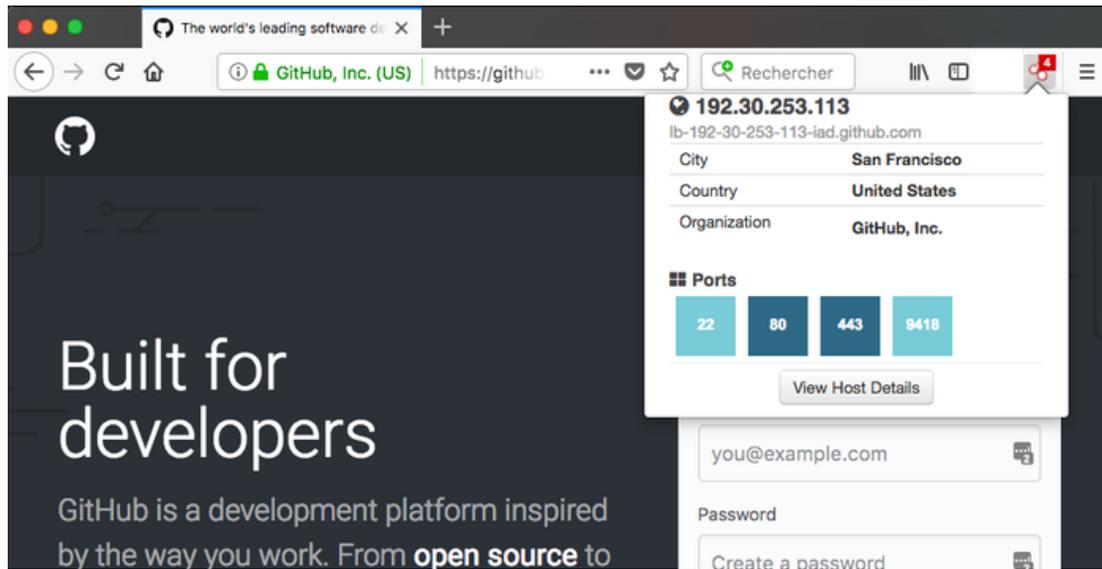
Top 10 Countries

US: 30672
DE: 5780
CN: 4072
BR: 2931
AR: 1745
JP: 1415
GB: 1147
IN: 939
FR: 738
CA: 675

Plugins

Firefox

Shodan.io

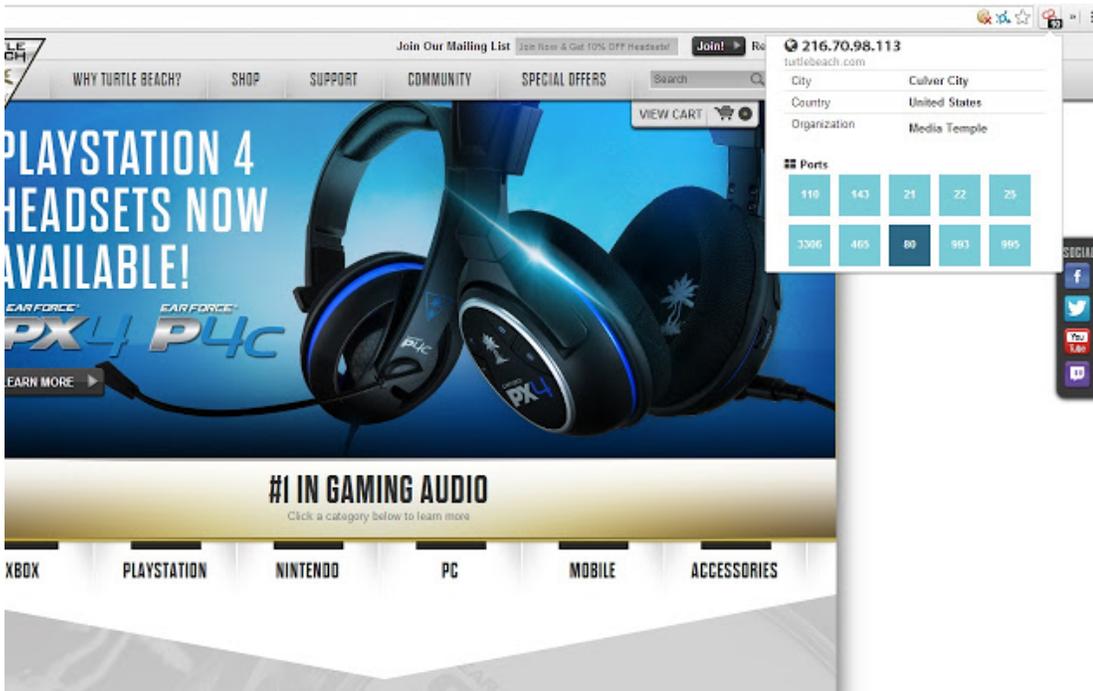


This add-on retrieves data gathered by Shodan.io of the current website you're browsing. It displays you general information such as the Organisation but also open ports.

Source

Chromium

Shodan



The Shodan plugin tells you where the website is hosted (country, city), who owns the IP and what other services/ports are open.

The Shodan plugin for Chrome automatically checks whether Shodan has any information for the current website. Is the website also running FTP, DNS, SSH or some unusual service? With this plugin you can see all the info that Shodan has collected on a given website/domain.

Shodan Search Query Syntax

Banner and properties

To get the most out of Shodan it's important to understand the search query syntax.

In Shodan's vocabulary a banner is an object containing the information of a service.

In the [official documentation](#) the below example of a simplified banner is given:

```
{  
  "data": "Moxa Nport Device  
    Status: Authentication disabled  
    Name: NP5232I_4728  
    MAC: 00:90:e8:47:10:2d",  
  "ip_str": "46.252.132.235",  
  "port": 4800,  
  "org": "Starhub Mobile",  
  "location": {  
    "country_code": "SG"  
  }  
}
```

Each key of the dictionary is called a property (data, ip_str, port, etc.). Each property stores a different type of information about the service.

By default Shodan is looking only into the data property, if no filter is provided.

Search filters

You could have found the previous example banner by searching Moxa Nport Device, but if you would have searched for devices from Starhub Mobile it wouldn't have returned the expected result. That's because, as I said earlier, by default, Shodan only searches the data property!

To search data using other properties we must use filters.

Search filters are special keywords to tell Shodan that you wish to search specific properties.

Filters are formatted as key:value.

Some examples:

- To search for devices located in the Starhub Mobile Network:
 - org:”Starhub Mobile”
- To search for devices located in Singapore:
 - country:SG
- And of course they can be combined:
 - org:”Starhub Mobile” country:SG

Properties/filters specification

Here is the complete list of properties for banners (Source: [Official documentation](#)).

General properties

Property	[Type] Description
asn	[String] The autonomous system number (ex. “AS4837”).
data	[String] Contains the banner information for the service.
ip	[Integer] The IP address of the host as an integer.
ip_str	[String] The IP address of the host as a string.
ipv6	[String] The IPv6 address of the host as a string. If this is present then the “ip” and “ip_str” fields wont be.
port	[Integer] The port number that the service is operating on.
timestamp	[String] The timestamp for when the banner was fetched from the device in the UTC timezone. Example: “2014-01-15T05:49:56.283713”
hostnames	[String[]] An array of strings containing all of the

hostnames that have been assigned to the IP address for this device.

domains	[String[]] An array of strings containing the top-level domains for the hostnames of the device. This is a utility property in case you want to filter by TLD instead of subdomain. It is smart enough to handle global TLDs with several dots in the domain (ex. “co.uk”)
location	[Object] An object containing all of the location information for the device.
location.area_code	[Integer] The area code for the device’s location. Only available for the US.
location.city	[String] The name of the city where the device is located.
location.country_code	[String] The 2-letter country code for the device location.
location.country_code3	[String] The 3-letter country code for the device location.
location.country_name	[String] The name of the country where the device is located.
location.dma_code	[Integer] The designated market area code for the area where the device is located. Only available for the US.
location.latitude	[Double] The latitude for the geolocation of the device.
location.longitude	[Double] The longitude for the geolocation of the device.
location.postal_code	[String] The postal code for the device’s location.

location.region_code	[String]	The name of the region where the device is located.
opts	[Object]	Contains experimental and supplemental data for the service. This can include the SSL certificate, robots.txt and other raw information that hasn't yet been formalized into the Banner Specification.
org	[String]	The name of the organization that is assigned the IP space for this device.
isp	[String]	The ISP that is providing the organization with the IP space for this device. Consider this the "parent" of the organization in terms of IP ownership.
os	[String]	The operating system that powers the device.
transport	[String]	Either "udp" or "tcp" to indicate which IP transport protocol was used to fetch the information

Optional Properties

Property	[Type]	Description
uptime	[Integer]	The number of minutes that the device has been online.
link	[String]	The network link type. Possible values are: "Ethernet or modem", "generic tunnel or VPN", "DSL", "IPIP or SIT", "SLIP", "IPSec or GRE", "VLAN", "jumbo Ethernet", "Google", "GIF", "PPTP", "loopback", "AX.25 radio modem".
title	[String]	The title of the website as extracted from the HTML source.

html	[String] The raw HTML source for the website.
product	[String] The name of the product that generated the banner.
version	[String] The version of the product that generated the banner.
devicetype	[String] The type of device (webcam, router, etc.).
info	[String] Miscellaneous information that was extracted about the product.
cpe	[String] The relevant Common Platform Enumeration for the product or known vulnerabilities if available. For more information on CPE and the official dictionary of values visit the CPE Dictionary.

SSL Properties

Property	[Type] Description
ssl.cert	[Object] The parsed certificate properties that includes information such as when it was issued, the SSL extensions, the issuer, subject etc.
ssl.cipher	[Object] Preferred cipher for the SSL connection
ssl.chain	[Array] An array of certificates, where each string is a PEM-encoded SSL certificate. This includes the user SSL certificate up to its root certificate.
ssl.dhparams	[Object] The Diffie-Hellman parameters if available: “prime”, “public_key”, “bits”, “generator” and an optional

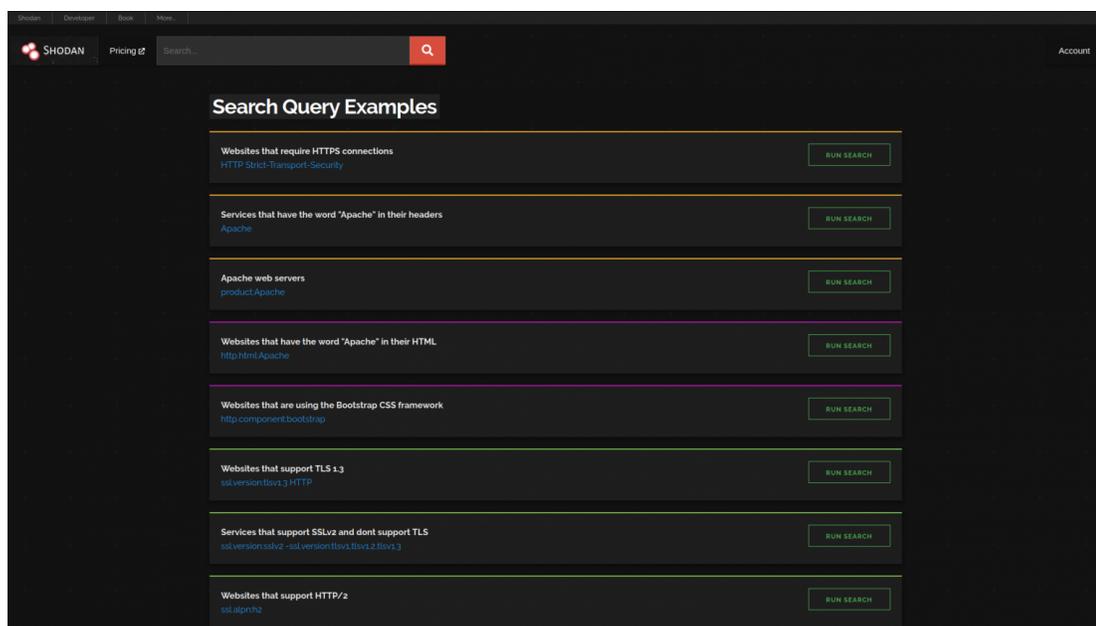
“fingerprint” if we know which program generated these parameters.

ssl.versions [Array] A list of SSL versions that are supported by the server. If a version isnt supported the value is prefixed with a “-”. Example: [“TLsv1”, “-SSLv2”] means that the server supports TLsv1 but doesnt support SSLv2.

Miscellaneous

The beta version of the website offers two useful pages:

- [Filters](#) – A filter/property cheat sheet list
- [Examples](#) – A list of search query examples



Shodan dorks & use cases

I'll start by showcasing some simple [snippets from shodan](#).

Examples are given for the CLI:

Number of devices vulnerable to Heartbleed

```
$ shodan count vuln:cve-2014-0160
80467
```

Get a list of subdomains for a domain

```
$ shodan domain cnn.com
CNN.COM
```

```
A 151.101.193.67
A 151.101.129.67
A 151.101.65.67
A 151.101.1.67
...
newsroom.blogs CNAME cnnnewsroom.wordpress.com
newsstream.blogs CNAME cnnnewsstream.wordpress.com
now CNAME www.cnn.com
ntm.blogs CNAME ntm.blogs.cnn.com.edgesuite.net
olympics.blogs CNAME olympics.blogs.cnn.com.edgesuite.net
olympics.edition CNAME cnn.site.scribblelive.com
on A 157.166.224.172
outfront.blogs CNAME cnnoutfront.wordpress.com
pagingdrgupta.blogs CNAME cnpagingdrgupta.wordpress.com
parkerspitzer.blogs CNAME cnparkerspitzer.wordpress.com
...
```

Create a private firehose for your network and subscribe to it

```
$ shodan alert create mynetwork 198.20.58.0/24 && shodan stream --
alerts=all
```

Find the top 10 most common vulnerabilities in Switzerland

```
$ shodan stats --facets vuln country:CH
Top 10 Results for Facet: vuln
```

```
cve-2018-1312 36,562
cve-2017-7679 31,109
cve-2019-0220 28,882
cve-2016-8612 27,638
cve-2018-17199 26,706
cve-2016-4975 26,560
cve-2018-1283 25,477
cve-2017-15715 25,477
cve-2017-15710 25,477
cve-2017-7668 23,261
```

I will continue with some publicly [shared queries](#):

D-Link Internet Camera DCS-5300 series, without authentication

```
$ shodan search 'd-Link Internet Camera, 200 OK'
```

FTP server with anonymous authentication enabled

```
$ shodan search '230 login successful port:21'
```

Databases

```
# MySQL
```

```
$ shodan search 'product:MySQL'
```

```
# MongoDB
```

```
$ shodan search 'product:MongoDB'
```

```
# elastic
```

```
$ shodan search 'port:9200 json'
```

```
# Memcached
```

```
$ shodan search 'product:Memcached'
```

```
# CouchDB
```

```
$ shodan search 'product:CouchDB'
```

```
# PostgreSQL
```

```
$ shodan search 'port:5432 PostgreSQL'
```

```
# Riak
```

```
$ shodan search 'port:8087 Riak'
```

```
# Redis
```

```
$ shodan search 'product:Redis'
```

```
# Cassandra
```

```
$ shodan search 'product:Cassandra'
```

Games

```
# Minecraft
```

```
$ shodan search 'Minecraft Server port:25565'
```

```
# Counter-Strike: Global Offensive
```

```
$ shodan search 'product:"Counter-Strike Global Offensive"'
```

```
# Starbound
```

```
$ shodan search 'product:Starbound'
```

```
# ARK: Survival Evolved
```

```
$ shodan search 'product:"ARK Survival Evolved"'
```

Industrial Control Systems

```
# XZERES Wind Turbine
```

```
$ shodan search 'title:"xzeres wind"'
```

```
# PIPS Automated License Plate Reader
$ shodan search 'html:"PIPS Technology ALPR Processors"'

# Modbus
$ shodan search 'port:502'

# Niagara Fox
$ shodan search 'port:1911,4911 product:Niagara'

# GE-SRTP
$ shodan search 'port:18245,18246 product:"general electric"'

# MELSEC-Q
$ shodan search 'port:5006,5007 product:mitsubishi'

# CODESYS
$ shodan search 'port:2455 operating system'

# S7
$ shodan search 'port:102'

# BACnet
$ shodan search 'port:47808'

# HART-IP
$ shodan search 'port:5094 hart-ip'

# Omron FINS
$ shodan search 'port:9600 response code'

# IEC 60870-5-104
$ shodan search 'port:2404 asdu address'
```

```
# DNP3
$ shodan search 'port:20000 source address'

# EtherNet/IP
$ shodan search 'port:44818'

# PCWorx
$ shodan search 'port:1962 PLC'

# Crimson v3.0
$ shodan search 'port:789 product:"Red Lion Controls"'

# ProConOS
$ shodan search 'port:20547 PLC'
```

And now, some [dorks](#) from [dalmoz](#):

ASCII video examples

[Shodan on asciinema.org](#)

Hacked Ubiquiti Networks Device

```
$ shodan search 'hacked-router-help-sos'
```

Surveillance cameras, user: admin, no password

```
$ shodan search 'hacked-router-help-sos'
```

Home routers' storage/attached USB storage

```
$ shodan search 'IPC$ all storage devices'
```

PBX phone gateways without authentication

```
$ shodan search 'port:23 console gateway -password'
```

Lantronix ethernet adapter's admin interface without password

```
$ shodan search 'Press Enter for Setup Mode port:9999'
```

Polycom video-conference system no-auth shell

```
$ shodan search '"polycom command shell"'
```

VNC servers without authentication

```
$ shodan search '"authentication disabled" port:5900,5901'
```

NPort serial-to-eth / MoCA devices without password

```
$ shodan search 'nport -keyin port:23'
```

Some *PenTestIT* queries:

Default Jenkins installations

```
$ shodan search 'http.favicon.hash:81586312'
```

SonarQube installations

```
$ shodan search 'http.favicon.hash:1485257654'
```

IBM WebSphere version disclosure

```
$ shodan search 'WASRemoteRuntimeVersion'
```

And to finish, a collection of search queries: *Awesome Shodan Search Queries*

- [Website](#)
- [GitHub](#)

Tools using Shodan

- <https://developer.shodan.io/apps>
- <https://github.com/BullsEye0/shodan-eye>
- https://www.rapid7.com/db/modules/auxiliary/gather/shodan_search
- <https://github.com/s0md3v/Striker>
- <https://github.com/lanmaster53/recon-ng>
- <https://github.com/smicallef/spiderfoot>
- https://github.com/DefensePointSecurity/threat_note
- <https://github.com/OWASP/Amass>
- <https://github.com/woj-ciech/Kamerka-GUI>
- <https://github.com/random-robbie/My-Shodan-Scripts>
- <https://github.com/jakejarvis/awesome-shodan-queries>
- <https://github.com/pielco11/fav-up>

ShodanSploit

It allows you to use all Shodan calls on your terminal and making detailed queries.

Github repository: <https://github.com/shodansploit/shodansploit>

Install:

```
git clone https://github.com/ismailtasdelen/shodansploit.git
```

```
cd shodansploit
```

```
python shodansploit.py
```

```
docker run -t ismailtasdelen/shodansploit
```

Docker Run:

```
docker run -rm -it ismailtasdelen/shodansploit
```

Menu:

[1] GET > /shodan/host/{ip}

[2] GET > /shodan/host/count

[3] GET > /shodan/host/search

[4] GET > /shodan/host/search/tokens

[5] GET > /shodan/ports

[6] GET > /shodan/exploit/author

[7] GET > /shodan/exploit/cve

[8] GET > /shodan/exploit/msb

[9] GET > /shodan/exploit/bugtraq-id

[10] GET > /shodan/exploit/osvdb

[11] GET > /shodan/exploit/title

[12] GET > /shodan/exploit/description

[13] GET > /shodan/exploit/date

[14] GET > /shodan/exploit/code

[15] GET > /shodan/exploit/platform

[16] GET > /shodan/exploit/port

```
[17] GET > /dns/resolve
```

```
[18] GET > /dns/reverse
```

```
[19] GET > /labs/honeyscore/{ip}
```

```
[20] GET > /account/profile
```

```
[21] GET > /tools/myip
```

```
[22] GET > /tools/httpheaders
```

```
[23] GET > /api-info
```

```
[24] Exit
```

Fav-Up

Description:

Lookups for real IP starting from the favicon icon and using Shodan.

Install:

At least python3.6 is required due to spicy syntax.

```
git clone https://github.com/pielco11/fav-up.git
```

```
pip3 install -r requirements.txt
```

Command overview:

```
usage: python3 favup [options]

optional arguments:
  -h, --help            show this help message and exit
  -kf KEY_FILE, --key-file KEY_FILE
                        Specify the file which contains the API key.
  -k KEY, --key KEY     Specify the API key.
  -sc, --shodan-cli    Load the API key from Shodan CLI.
  -ff FAVICON_FILE, --favicon-file FAVICON_FILE
                        Load the favicon icon from a local file.
  -fu FAVICON_URL, --favicon-url FAVICON_URL
                        Load the favicon icon from an URL.
  -w WEB, --web WEB     Extracts the favicon location from the page.
  -fh FAVICON_HASH, --favicon-hash FAVICON_HASH
                        Running from direct favicon hash number
  -fl FAVICON_LIST, --favicon-list FAVICON_LIST
                        Iterate over a file that contains the full path of all
                        the icons which you want to lookup.
  -ul URL_LIST, --url-list URL_LIST
                        Iterate over a file that contains the full URL of all
                        the icons which you want to lookup.
  -wl WEB_LIST, --web-list WEB_LIST
                        Iterate over a file that contains all the domains
                        which you want to lookup.
  -o OUTPUT, --output OUTPUT
                        Specify output file, currently supported formats are
                        CSV and JSON.
```

Examples

Favicon-file:

```
python3 favUp.py --favicon-file favicon.ico -sc
```

Favicon-url

```
python3 favUp.py --favicon-url
https://domain.behind.cloudflare/assets/favicon.ico -sc
```

Web

```
python3 favUp.py --web domain.behind.cloudflare -sc
```

Module

```
from favUp import FavUp

f = FavUp()

f.shodanCLI = True

f.web = "domain.behind.cloudflare"

f.show = True

f.run()

for result in f.faviconsList:

    print(f"Real-IP: {result['found_ips']}")

    print(f"Hash: {result['favhash']}")
```

Related info:

<https://pielco11.ovh/posts/cloud-hunting/>

Articles of advanced uses

- [Pivoting with Property Hashes](#)
- [Working with Shodan Data Files](#)
- [Create a GIF from an IP Image History](#)

Shodan alternatives

Web commercial alternatives

- [Onyphe](#) – pretty like Shodan but in addition of scanning it also crawls data from passive DNS lookup, threatlist lookup and paste sites lookup. However the free version is more limited than Shodan.
- [ZoomEye](#) – is also very similar to Shodan, has a great set of advanced filters that are more documented than Shodan's and a ton of pre-set queries. There is also a great free API tier.
- [Censys](#) – like Shodan, it also has the ability to track changes, send alerts, etc. It seems there is no free API plan, the only free option is to use the website.
- [thingful](#) – a search engine that is targeting only the Internet of Things
- [FOFA](#) – is like Shodan, it also has a CLI tool and a Java, Go, C and Python library.
- [Greynoise](#) – is like Shodan, but there is no free API plan, only web visualizer access. Also has a python library and a CLI tool.
- [BinaryEdge](#) – like Shodan there are the search engine, honeypots/sensors detector but also an uncommon feature: Torrents/DHT Monitoring. There is a free Web & API plan.

Open source self hosted alternatives

- [IVRE](#) – [Source](#) > is a network recon framework, including tools for passive recon (flow analytics relying on Bro, Argus, Nfdump, fingerprint analytics based on Bro and p0f) and active recon (IVRE uses Nmap to run scans, can use ZMap as a pre-scanner; IVRE can also import XML output from Nmap and Masscan).
- It has a WebUI and a CLI tool.
- [purplepee](#) – [Source](#) > it allows you to view general relations about a websites HTTP header, websites DNS records, websites SSL certificates and open TCP ports as well as ASN whois information.
- In addition of the open-source project, there is also a public instance hosted online.

Thanks

First I want to thanks John C. Matherly a.k.a. @achillean to have created Shodan and maintained it for 10 years.

Then I also want to thanks Porter Adams (Co-Founder of Disappear Digital) and Ismael Gonzalez (<http://osint.team/> member).

Finally I want to thanks Nathaniel Fried and Peter James Hansen for their amazing work at TurgenSec.

References

- [SHODAN for Penetration testers](#)
- [Shodan – CLI Snippets](#)
- [Shodan Command-Line Interface](#)
- [Shodan library documentation](#)
- [Shodan help center](#)
- More references directly quoted during the article

About the author

My name is *Alexandre ZANNI* aka noraj. I'm a pentester and ethical hacker. Also I'm a staff member of the [RTFM association](#) and a developer of [BlackArch Linux](#).

My hacker page: pwn.by/noraj

C O M M E N T S

[OUR SITE](#)  [FACEBOOK](#)
